

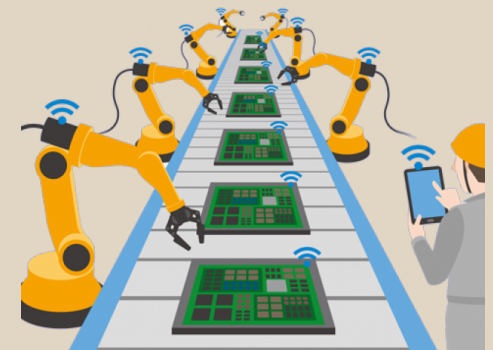
# A Decentralized Medium Access Protocol for Real-Time Wireless Ad Hoc Networks With Unreliable Transmissions

I-Hong Hou

Joint work with Ping-Chun Hsieh

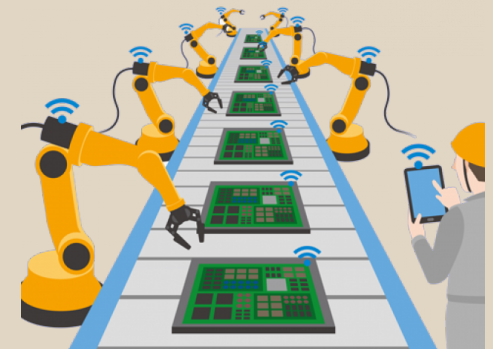
# Background: Factory IoT

- Factory IoT: Connect **sensors** and **actuators** through **wireless**



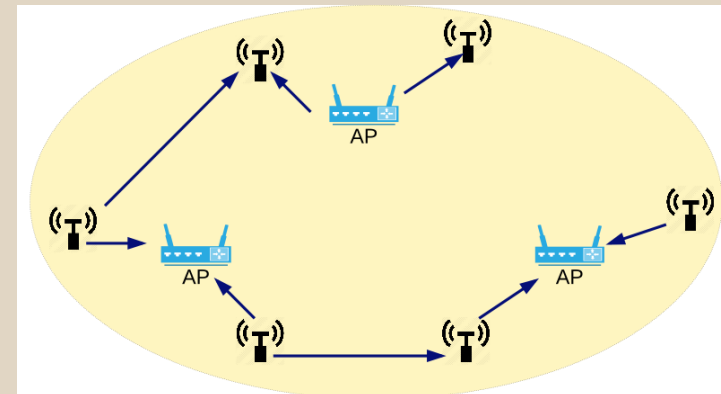
# Background: Factory IoT

- Factory IoT: Connect **sensors** and **actuators** through **wireless**
- **Sensors**: Generate packets from sensed data
  - Unpredictable traffic patterns
- **Actuators**: Make decisions based on received packets
  - Safety relies on strict delay/reliability guarantees
- **Wireless**:
  - Transmissions are unreliable
  - Wireless links interfere with each other

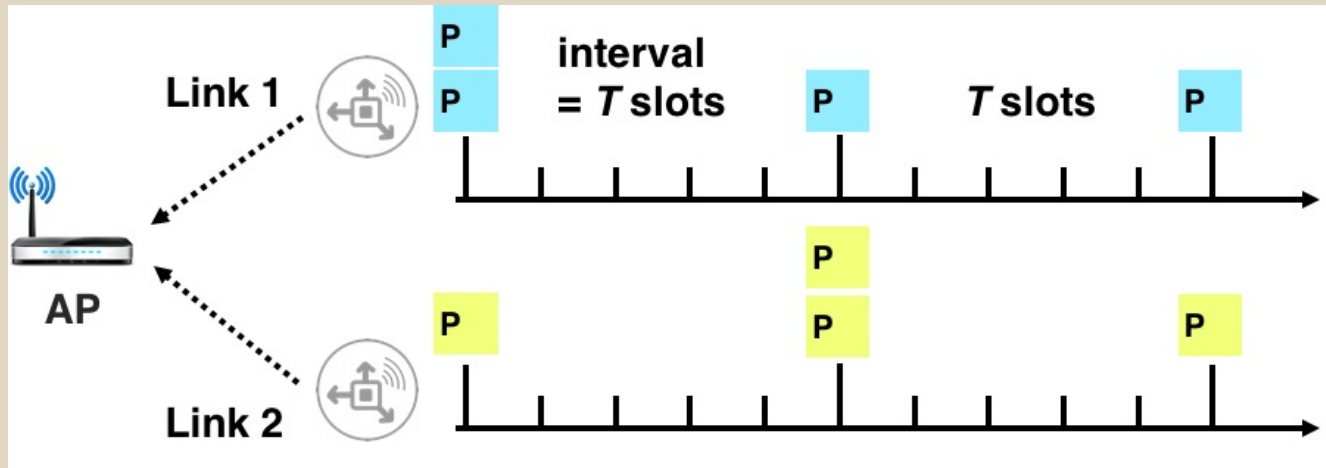


# Network Topology

- A system with many sensors and APs
- A total of  $N$  links (uplink/downlink/M2M)
- All links interfere with each other, but nodes cannot overhear others' transmissions
  - Interference range  $>$  transmission range
- Nodes have perfect carrier sensing capabilities
  - Avoiding hidden terminal by setting a low CS threshold
- Each link carries a real-time flow with hard delay/reliability requirements

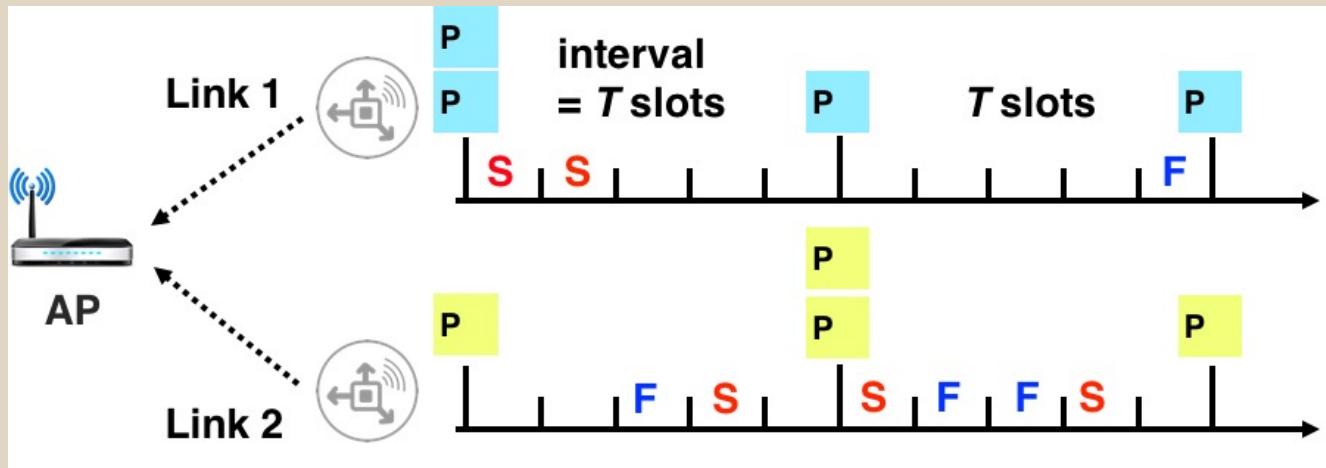


# Packet Arrivals and Departures



- Time is divided into intervals. One interval =  $T$  transmissions
- At the beginning of each interval, each link generates a random number of packets
- At the end of each interval, all undelivered packets are dropped
  - Delay of each delivered packets  $\leq T$

# Wireless Transmissions and Service Requirements



- Only one link can transmit at any time
  - Multiple simultaneous transmissions lead to collision
- Each transmission of link  $n$  is successful with probability  $p_n$ 
  - Transmissions can fail due to channel fading
- Each link  $n$  requires that its timely-throughput is at least  $q_n$  packets/interval

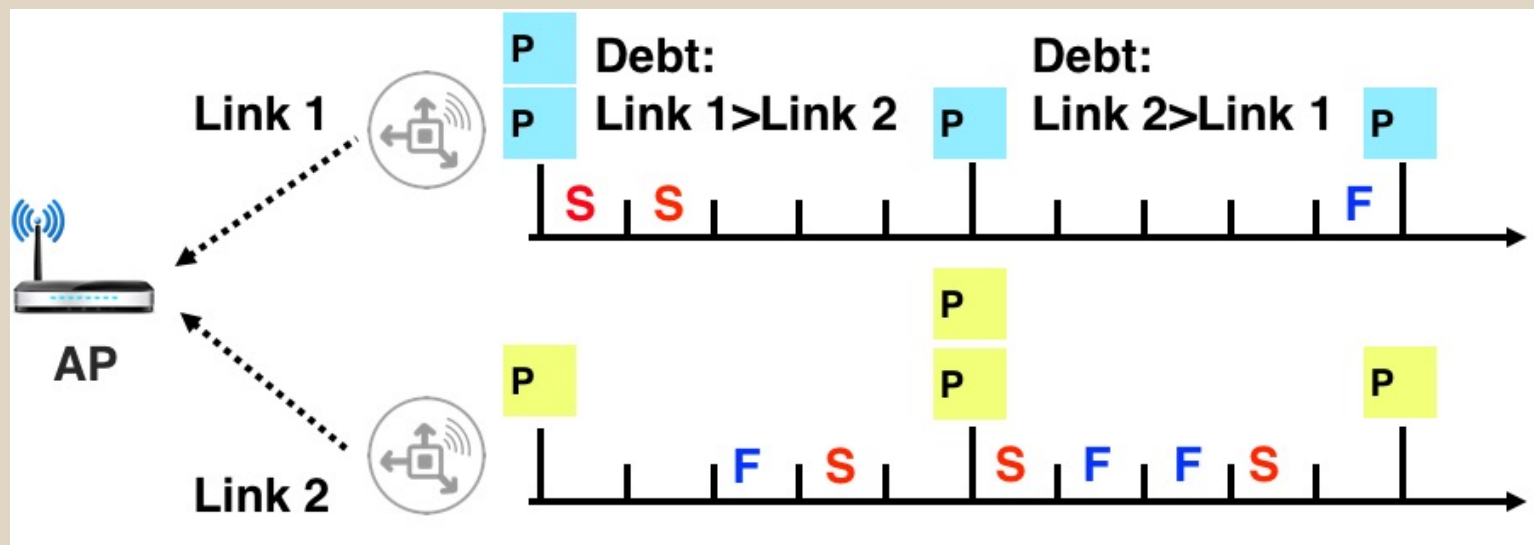
# Goal of This Work

- Design a decentralized protocol that delivers  $q_n$  timely-throughput for each link  $n$ , whenever it is theoretically feasible to do so
- Such a protocol is called feasibility-optimal

# Feasibility-Optimal Centralized Policy



- Def: The **debt** of link  $n$  in the  $k^{\text{th}}$  interval is  $p_n [kq_n - (\text{Actual deliveries for } n \text{ up to interval } k)]$
- Largest Debt First Policy (LDF): In each slot, transmit the packet with the highest debt
- LDF has been proved to be feasibility-optimal





# Challenges of Decentralized Protocols



- Packets arrive randomly, and links have no knowledge about others' packet arrivals
- Transmissions are unreliable, and links cannot overhear others' transmissions

# Challenges of Decentralized Protocols

- Packets arrive randomly, and links have no knowledge about others' packet arrivals
- Transmissions are unreliable, and links cannot overhear others' transmissions
- → A link cannot know the debts of other links
- → It cannot even know whether a transmission is successful

# Our Contributions

---

- Propose a protocol that maintains an ordering among links to avoid transmission collisions
  - Each link only knows its own position in the ordering

# Our Contributions

---

- Propose a protocol that maintains an ordering among links to avoid transmission collisions
  - Each link only knows its own position in the ordering
  - Links can cooperatively change their positions without knowing the states of other links

# Our Contributions

- Propose a protocol that maintains an ordering among links to avoid transmission collisions
  - Each link only knows its own position in the ordering
  - Links can cooperatively change their positions without knowing the states of other links
  - Centralized coordination is only needed at system startup

# Our Contributions

- Propose a protocol that maintains an ordering among links to avoid transmission collisions
  - Each link only knows its own position in the ordering
  - Links can cooperatively change their positions without knowing the states of other links
  - Centralized coordination is only needed at system startup
  - Links coordinate among themselves only through carrier sensing → No explicit message exchange

# Our Contributions

- Propose a protocol that maintains an ordering among links to avoid transmission collisions
  - Each link only knows its own position in the ordering
  - Links can cooperatively change their positions without knowing the states of other links
  - Centralized coordination is only needed at system startup
  - Links coordinate among themselves only through carrier sensing → No explicit message exchange
  - Small and bounded overhead

# Our Contributions

- Propose a protocol that maintains an ordering among links to avoid transmission collisions
  - Each link only knows its own position in the ordering
  - Links can cooperatively change their positions without knowing the states of other links
  - Centralized coordination is only needed at system startup
  - Links coordinate among themselves only through carrier sensing → No explicit message exchange
  - Small and bounded overhead
- Propose an algorithm that is feasibility-optimal

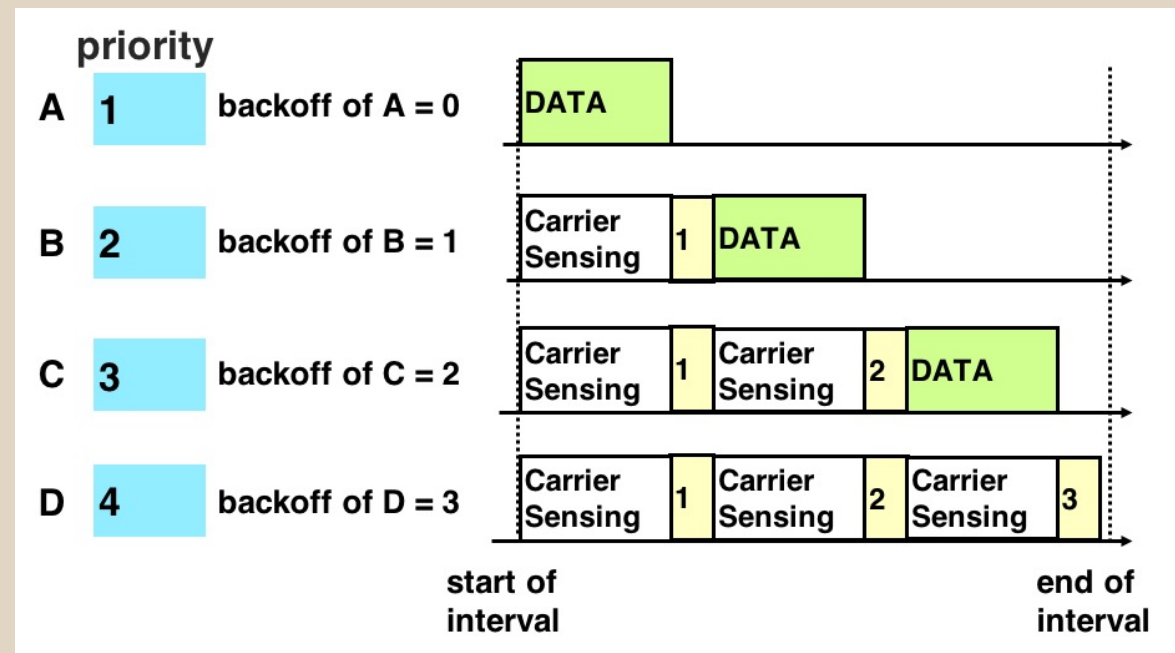


# Protocol Overview: System Startup

- At startup, a centralized controller determines a priority ordering among links randomly
- The controller informs each link of its position
  - No two links have the same position
- The controller also notifies all links of a common random seed
  - All links can generate a common sequence of pseudo-random numbers in the range  $[0, N - 1]$

# Protocol Overview: Collision-Free Transmission

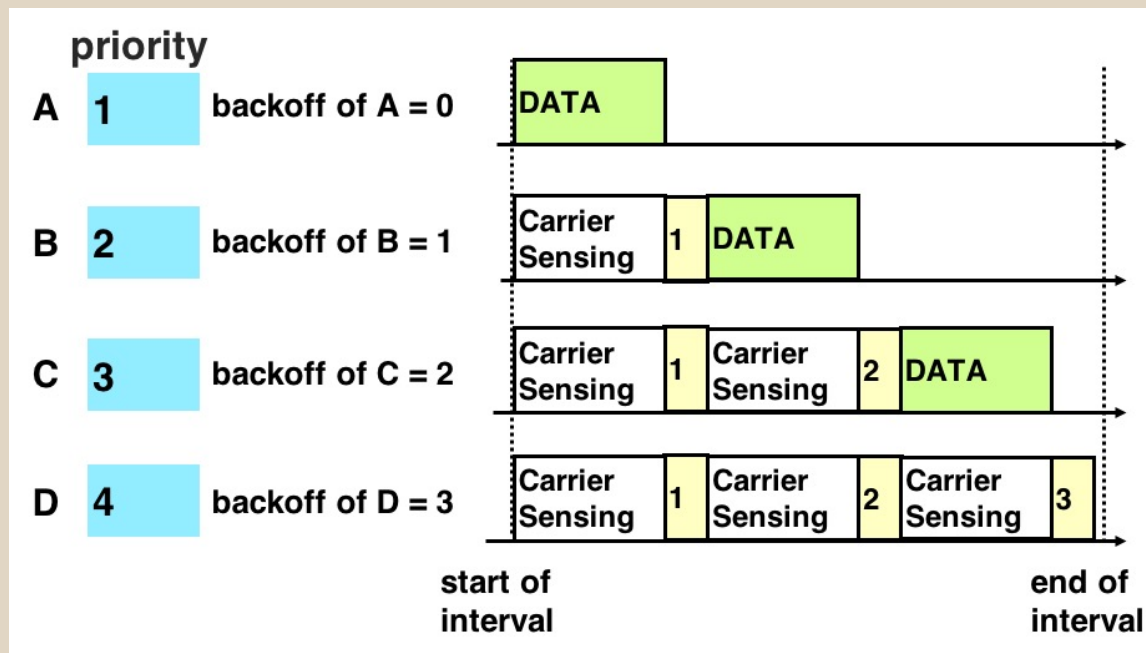
- The link of priority  $n$  only begins to transmit after it senses  $(n - 1)$  units of channel idle time



# Protocol Overview: Collision-Free Transmission



- The link of priority  $n$  only begins to transmit after it senses  $(n - 1)$  units of channel idle time
- Similar to the backoff policy in WiFi
- Total idle time  $\leq (N - 1)$  units
- WiFi: tx time  $\sim 100$  time units



# Protocol Overview: Exchanging Priority Positions



- In interval  $k$ , links in positions  $R_k$  and  $R_k + 1$  may change positions
  - $R_k$  is the  $k^{\text{th}}$  random number in the common sequence

# Protocol Overview: Exchanging Priority Positions



- In interval  $k$ , links in positions  $R_k$  and  $R_k + 1$  may change positions
  - $R_k$  is the  $k^{\text{th}}$  random number in the common sequence
- Position  $n, n < R_k$ : Backoff =  $n - 1$
- Position  $R_k$ : Choose backoff from  $\{R_k - 1, R_k + 1\}$
- Position  $R_k + 1$ : Choose backoff from  $\{R_k, R_k + 2\}$
- Position  $m, m > R_k$ : Backoff =  $m + 1$

initial priority	1	2	3	4
backoff timer	0	1 or 3	2 or 4	5

# Protocol Overview: Exchanging Priority Positions



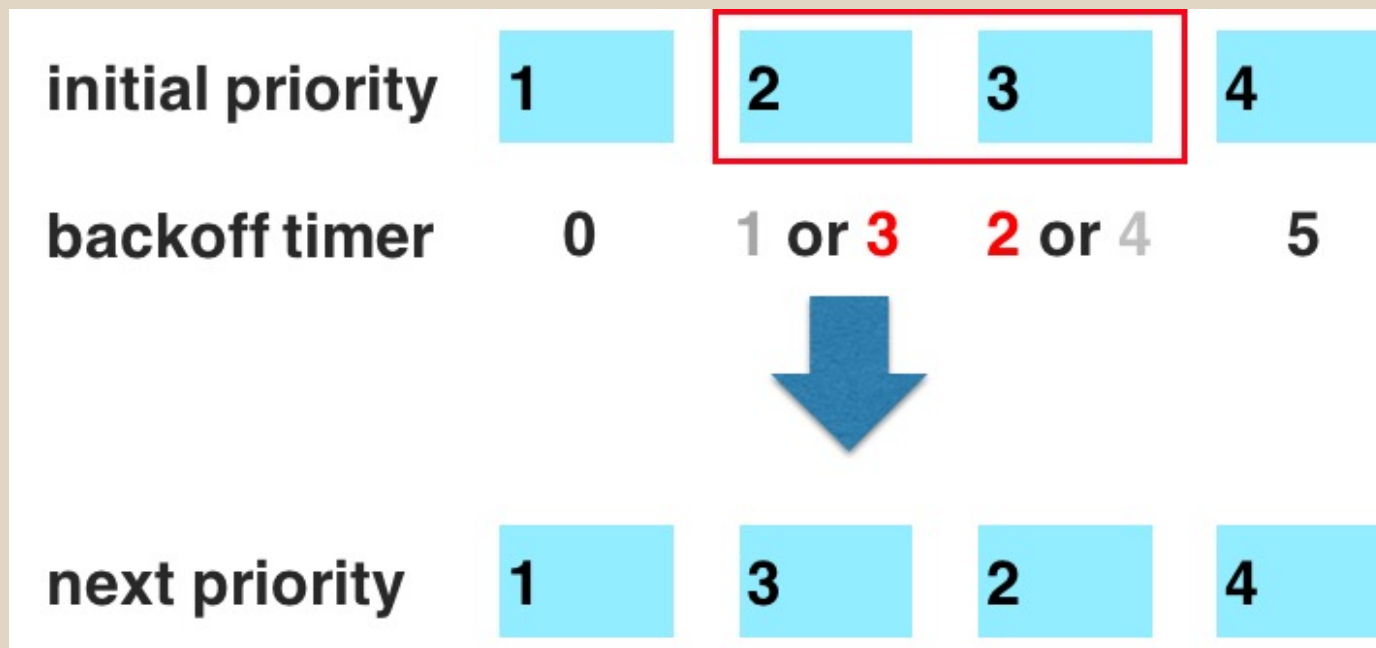
- In interval  $k$ , links in positions  $R_k$  and  $R_k + 1$  may change positions
  - $R_k$  is the  $k^{\text{th}}$  random number in the common sequence
- Position  $n, n < R_k$ : Backoff =  $n - 1$
- Position  $R_k$ : Choose backoff from  $\{R_k - 1, R_k + 1\}$
- Position  $R_k + 1$ : Choose backoff from  $\{R_k, R_k + 2\}$
- Position  $m, m > R_k$ : Backoff =  $m + 1$
- All backoff timers are different  $\rightarrow$  No collision!

initial priority	1	2	3	4
backoff timer	0	1 or 3	2 or 4	5

# Protocol Overview: Exchanging Priority Positions (II)



- Positions  $R_k$  and  $R_k + 1$  change positions only when position  $R_k$  chooses backoff =  $R_k + 1$ , and position  $R_k + 1$  chooses backoff =  $R_k$
- This event can be detected by carrier-sensing



# Protocol Summary

---

- The centralized controller is only involved in determining initial priority ordering and common random seed at startup
  - Collision-free transmissions are achieved through carrier sensing and backoff timer
  - The change of priority ordering is also done through carrier sensing and backoff timer
-



# How to Choose Backoff Timer?

- In interval  $k$ , links in positions  $R_k$  and  $R_k + 1$  may change positions

# How to Choose Backoff Timer?

- In interval  $k$ , links in positions  $R_k$  and  $R_k + 1$  may change positions
- Let:  $d_1 =$  debt of  $R_k$ ,  $d_2 =$  debt of  $R_k + 1$
- Position  $R_k$ : Backoff = 
$$\begin{cases} R_k - 1, w.p. \frac{e^{d_1}}{B + e^{d_1}} \\ R_k + 1, else \end{cases}$$
- Position  $R_k + 1$ : Backoff = 
$$\begin{cases} R_k, w.p. \frac{e^{d_2}}{B + e^{d_2}} \\ R_k + 2, else \end{cases}$$

# How to Choose Backoff Timer?

- In interval  $k$ , links in positions  $R_k$  and  $R_k + 1$  may change positions
- Let:  $d_1 =$  debt of  $R_k$ ,  $d_2 =$  debt of  $R_k + 1$
- Position  $R_k$ : Backoff = 
$$\begin{cases} R_k - 1, w.p. \frac{e^{d_1}}{B + e^{d_1}} \\ R_k + 1, else \end{cases}$$
- Position  $R_k + 1$ : Backoff = 
$$\begin{cases} R_k, w.p. \frac{e^{d_2}}{B + e^{d_2}} \\ R_k + 2, else \end{cases}$$
- Intuition: Larger debt leads to smaller backoff

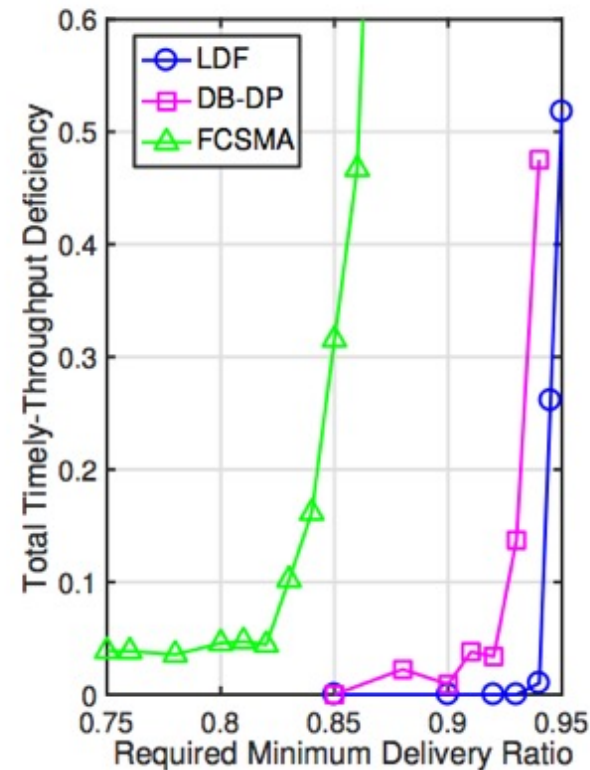
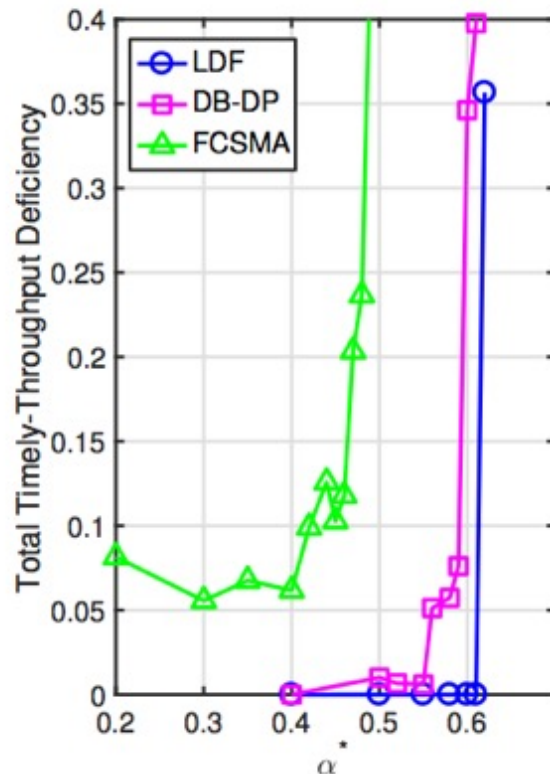
# Optimality Result

- **Theorem: Our policy is feasibility-optimal**
- Proof idea: We show that the priority ordering under our policy is similar to the LDF policy, which is the feasibility-optimal centralized policy
- Note: Each link picks the backoff timer solely based on its own debt

# NS3 Simulation Settings

- Simulations under 802.11a standard
  - Data rate = 54 Mb/s; Unit time for backoff = 9 us
- Policies:
  - Our debt-based decentralized priority algorithm (DB-DP)
  - LDF: Centralized policy. No overhead at all
  - FCSMA (by Li and Eryilmaz): Provably optimal without guarantees on overhead
- Metric: total timely-throughput deficit  
 $\sum (q_n - \text{Actual timely-throughput of } n)^+$

# Simulation Results



**DB-DP achieves almost the same throughput as LDF**

# Conclusions

- Consider the problem of scheduling wireless transmissions for real-time traffic
- Propose a decentralized protocol
- All coordination is carried out through carrier sensing and backoff timer
- Propose a policy for choosing backoff timer
- The policy is provably optimal with bounded overhead