Self-Stabilizing Leader Election for Single-Hop Wireless Networks despite Jamming

Andrea Richa, Jin Zhang

Arizona State University

Stefan Schmid
Deutsche Telekom Labs &
TU Berlin

Christian Scheideler

University of Paderborn

Motivation

Channel availability hard to model:

- Background noise (microwave etc.)
- Temporary Obstacles (cars etc.)
- Mobility
- Co-existing networks
- Jammers
- Etc.

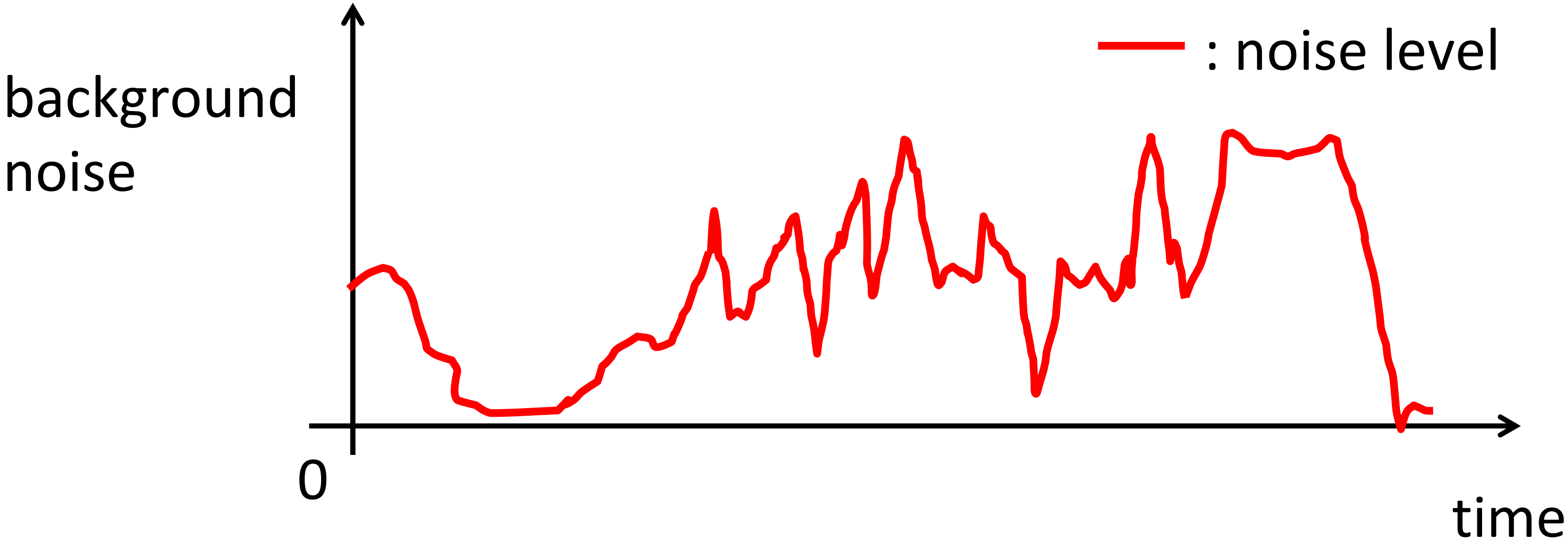
Motivation

Ideal world:

Usual approach adopted in theory.

Motivation

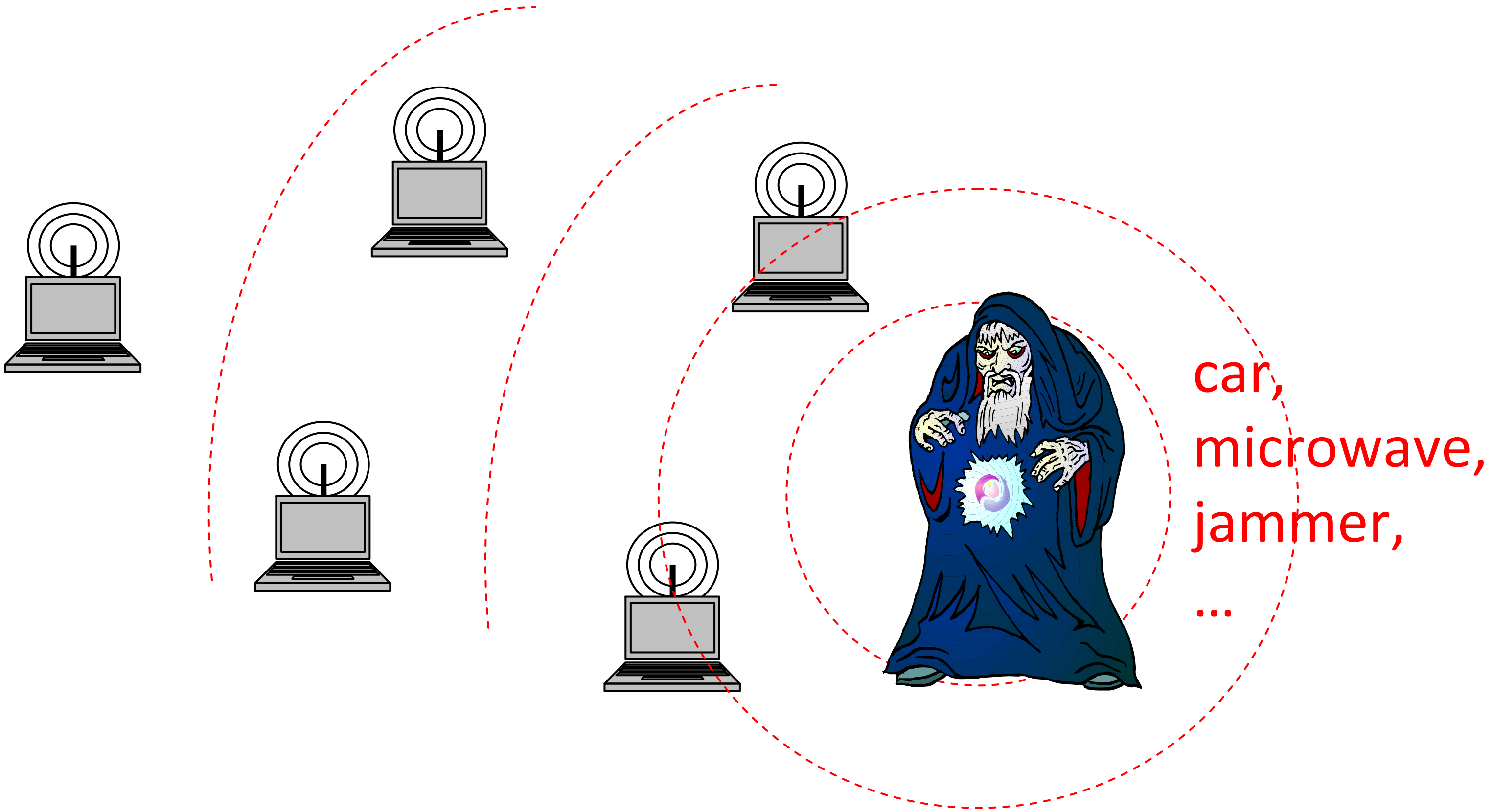
Real world:



How to model that???

Our Approach: Adversarial Jammer

Idea: model unpredictable behavior via adversary!



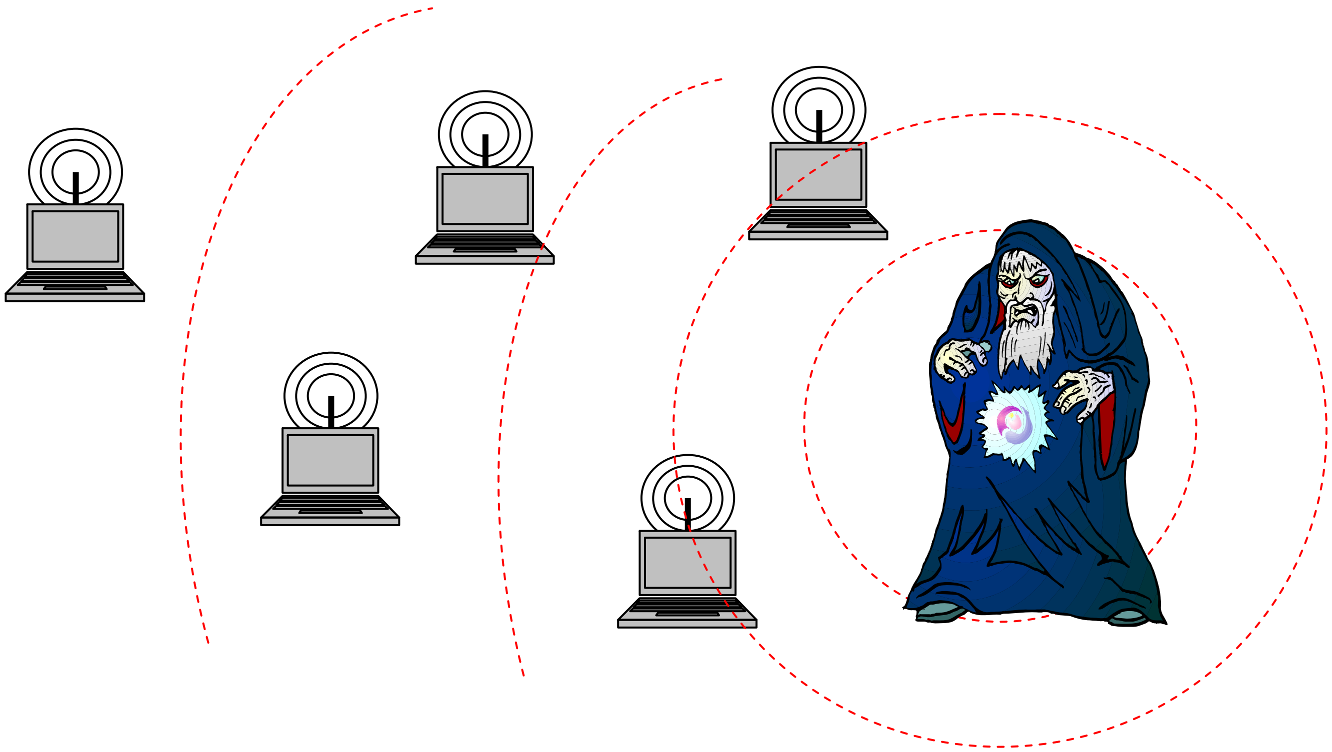
Our Approach: Adversarial Jammer

(T,λ)-bounded jammer, $0 < \lambda < 1$:

- jammer knows all protocols used by nodes and entire history of communication activity (at PHY layer) BUT does not know internal states, cannot read messages
- in any time window of size $w \geq T$, the adversary may jam up to λw time steps

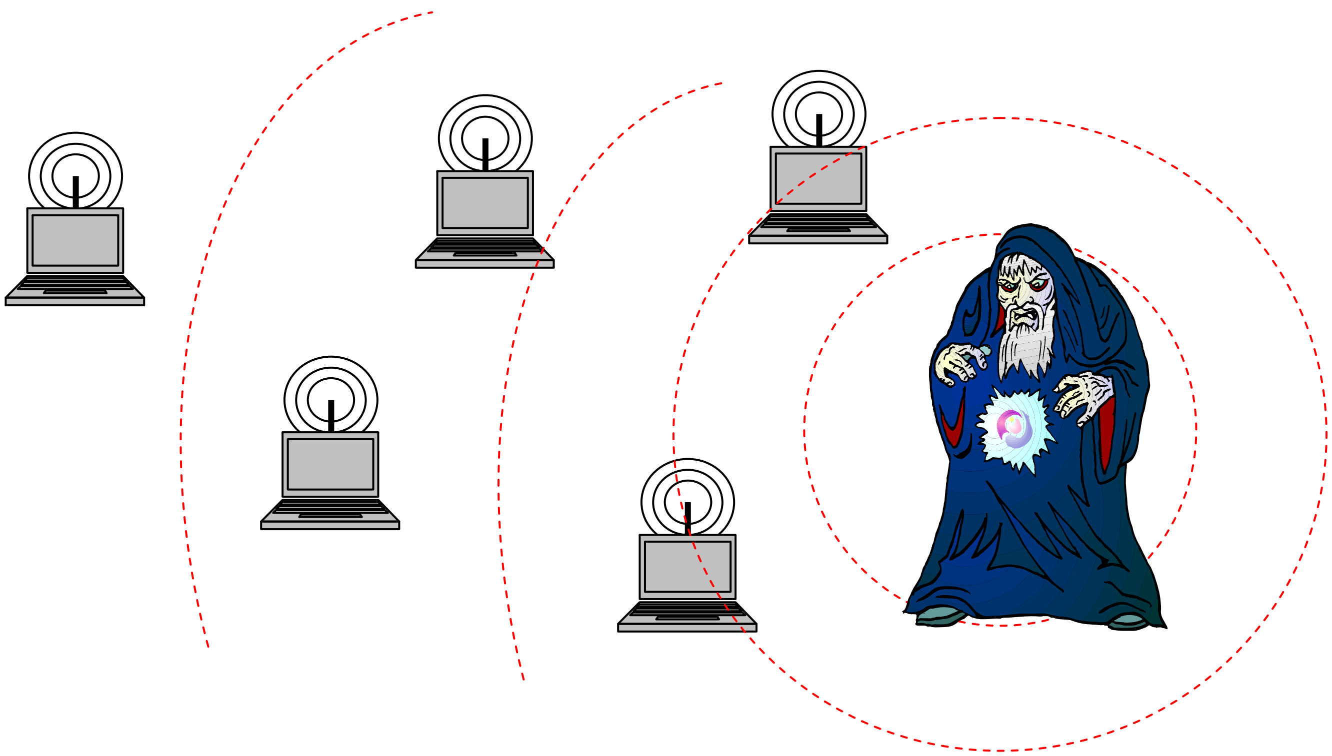
Our Approach: Adversarial Jammer

- *n* reliable nodes and a (*T*,*λ*)-bounded jammer
- the nodes do not know *n*, *T* or *λ*



Our Approach: Adversarial Jammer

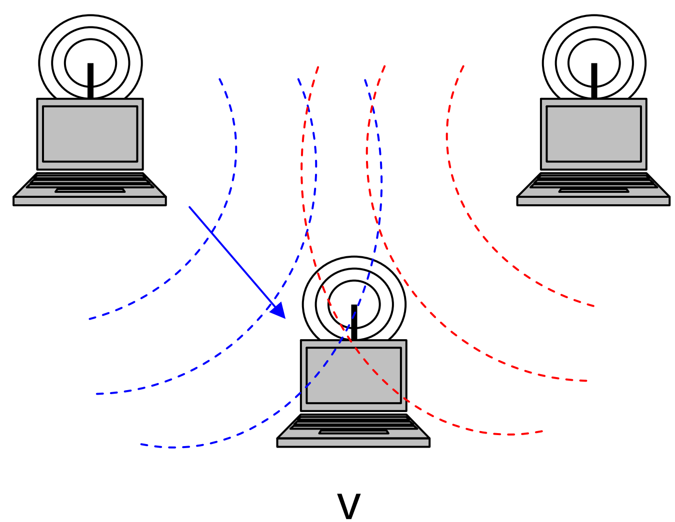
- all nodes within transmission range of each other and of the jammer (single-hop network)



Our Approach: Adversarial Jammer

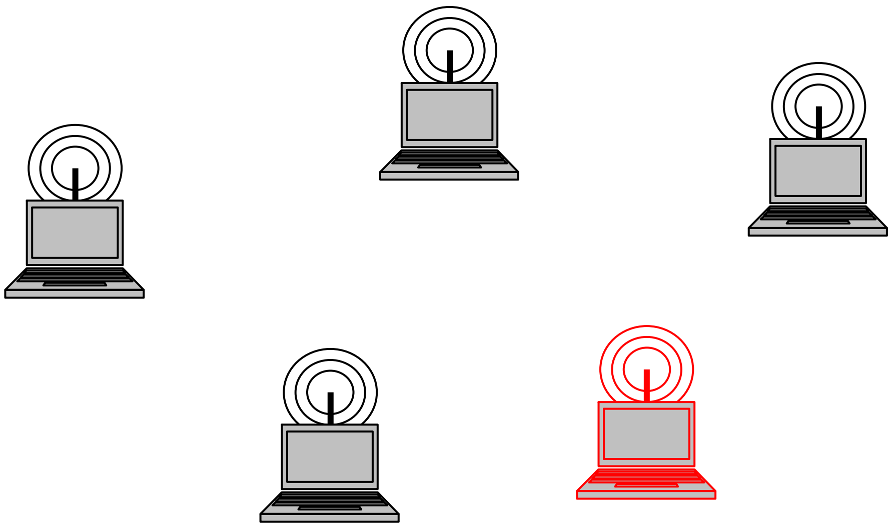
- at each time step, a node may decide to transmit a packet
- a node may transmit or sense the channel at a time step but not both (half-duplex)
- when sensing the channel a node *v* may
  - sense an idle channel
  - receive a packet
  - sense a busy channel

collision → busy ← jammer



Leader Election

Our goal: select a leader among the nodes



Challenges: we may start in any state, there is wireless jammer

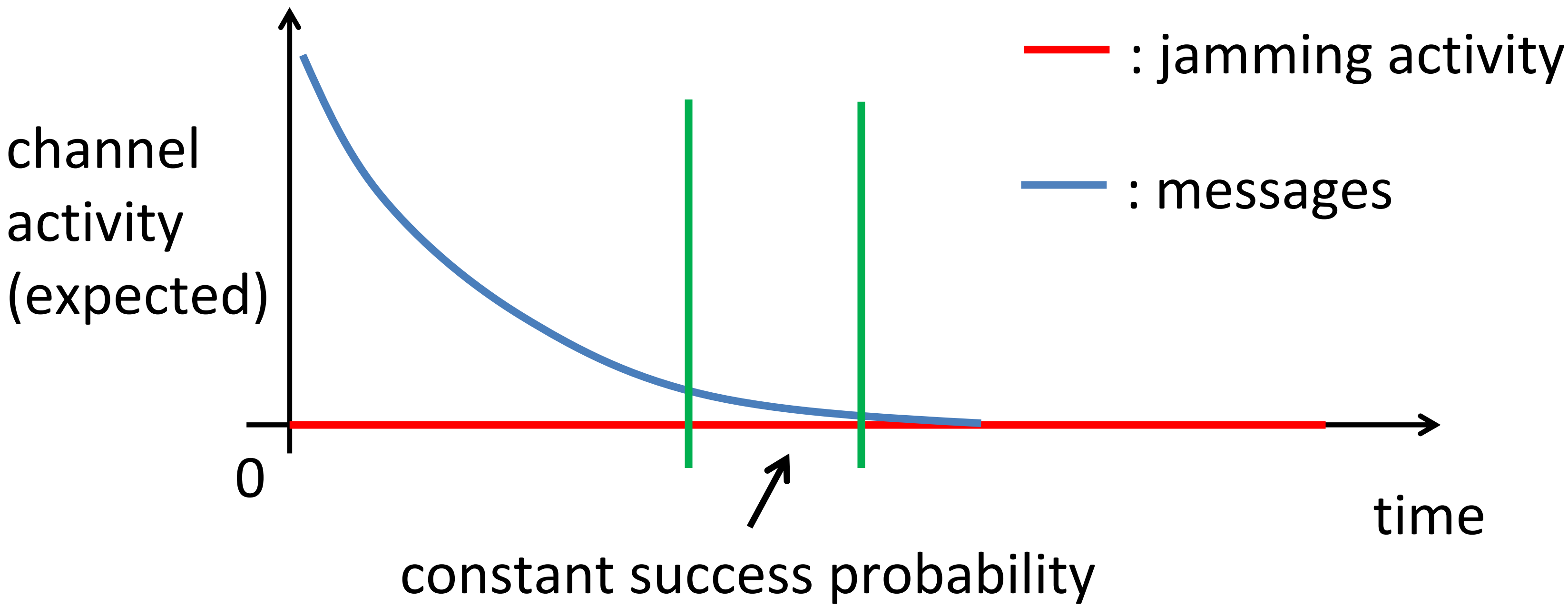
Leader Election

- Goal: design a self-stabilizing protocol that elects a single node as the leader, irrespective of the jamming activity
- No leader election protocol proposed so far can achieve that within our model
- Challenges:
  - a leader node should let the others (*followers* or other leaders) know that he is still around
  - the followers should be able to notice when there is no leader in the network

Leader Election

Why is leader election difficult under jamming?

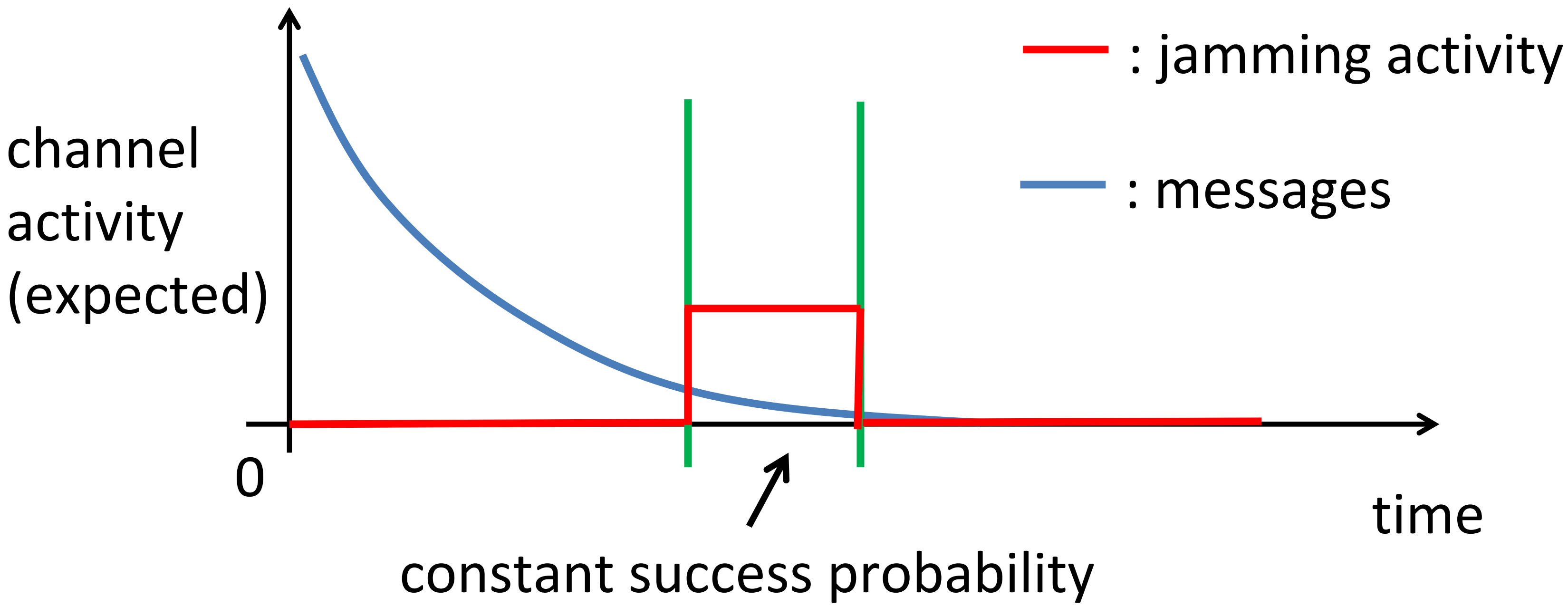
Example: exponential/polynomial backoff



Leader Election

Why is leader election difficult under jamming?

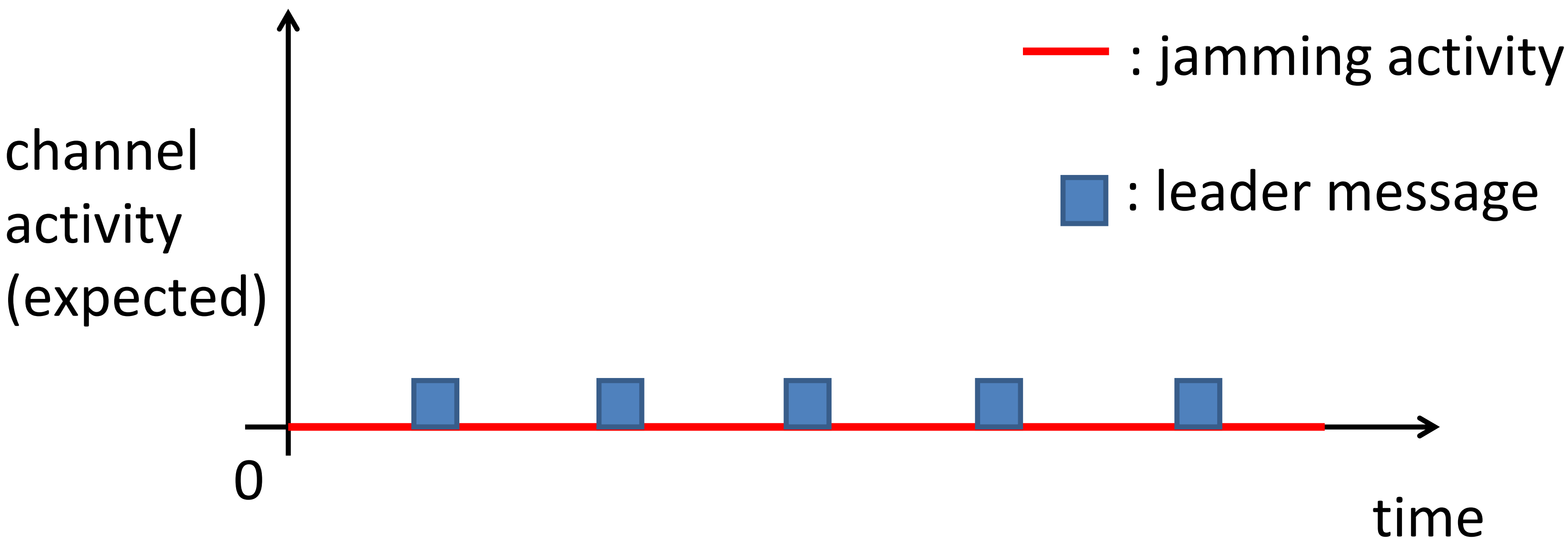
Example: exponential/polynomial backoff



Leader Election

Why is leader election difficult under jamming?

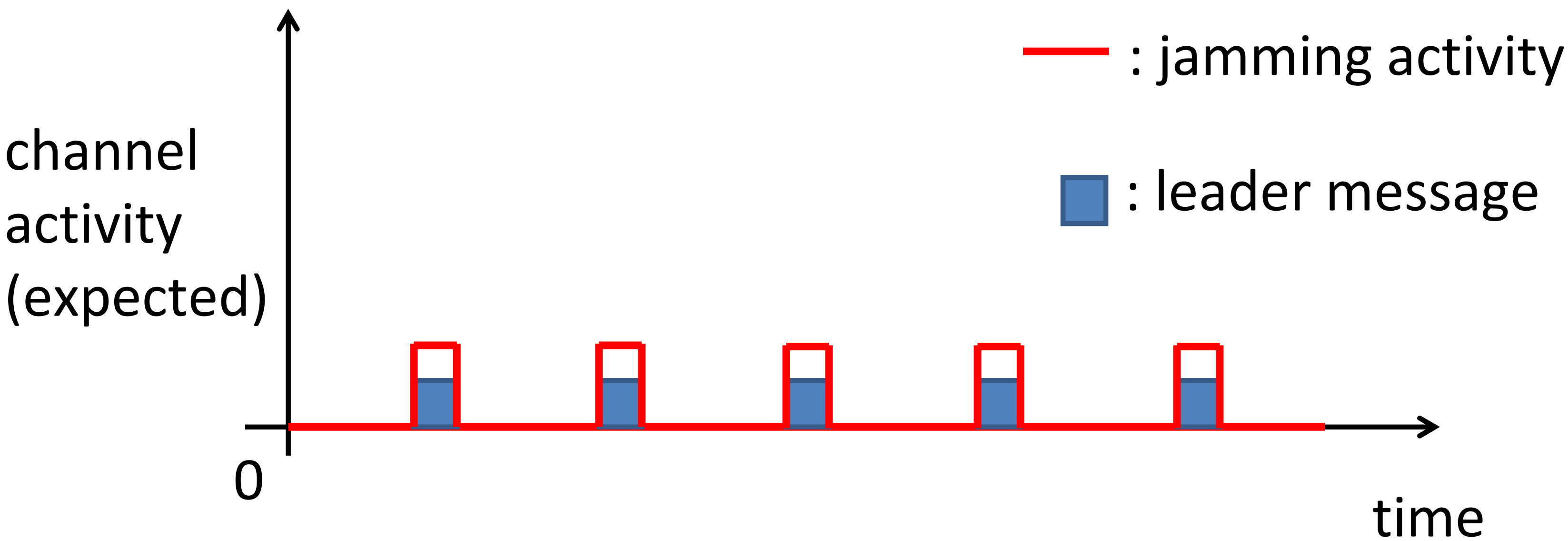
Example: reserved leader slot to notify nodes about leader



Leader Election

Why is leader election difficult under jamming?

Example: reserved leader slot to notify nodes about leader



Overview

- Related work
- Jamming-resistant MAC protocol
- Ideas leading to robust leader election
- Conclusion

Related Work

Leader election:

- Classical problem in theory with MANY publications
- CSMA-based MAC protocols in wireless networks implicitly use leader election to transmit messages
- Self-stabilizing leader election:
  - Antonoiu, Srimani 1996
  - Cai, Izumi, Wada 2009
  - Ghosh, Gupta 1996
  - Itkis, Lin, Simon 1995
  - …

Related Work

Defenses against jamming:

- PHY layer: spread spectrum and frequency hopping, BUT ISM band too narrow
- MAC layer:
  - IEEE 802.11 not even robust against simple jammers [Bayraktaroglu, King, Liu, … 2008]
  - Coding strategies [Chiang, Hu 2007]
  - Channel monitoring and retreat [Alnifie, Simon 2007], [Xu, Wood, Zhang 2004]
  - Hide messages from jammer [Wood, Stankovic, Zhou 2007]

Related Work

Our jamming model:

- Awerbuch, Richa, S (PODC 2008):
  MAC protocol for single-hop network +
  simple leader election protocol
- Richa, S, Schmid, Zhang (DISC 2010): multi-hop networks
- Richa, S, Schmid, Zhang (ICDCS 2011): reactive jammer

Other jamming models in theory:

- Pelc, Peleg 2005 (random jamming)
- Koo, Bhandari, Katz, Vaidya 2006 (bounded budget)
- Gilbert, Rachid Guerraoui, Kowalski, Newport 2009
  (multi-channel, adversary can disrupt any t of c channels)

Leader Election Protocol

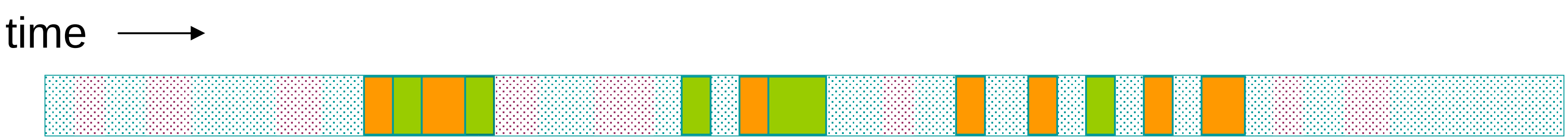
Our leader election protocol is based on jamming-resistant MAC protocol from PODC 08.

Basic idea: only adapt access probabilities based on idle and successful time steps

Leader Election Protocol

Our leader election protocol is based on jamming-resistant MAC protocol from PODC 08.

Basic idea: only adapt access probabilities based on idle and successful time steps



- idle steps
- successful transmissions
- steps jammed by adversary
- steps where collision occurred but no jamming

Jamming-resistant MAC protocol

[PODC 08]

In each step:

- node v sends a message with probability p_v . If v decides not to send a message then
  - if v senses an idle channel, then $p_v = \min\{(1+\gamma)p_v, p_{max}\}$
  - if v successfully receives a message, then $p_v = p_v/(1+\gamma)$ and $T_v = \max\{T_v - 1, 1\}$
- $c_v = c_v + 1$. If $c_v > T_v$ then
  - $c_v = 1$
  - if v did not receive a message successfully in the last T_v steps then $p_v = p_v/(1+\gamma)$ and $T_v = T_v + 1$

Algorithm 1 Leader Election: Follower

$mc := c_v \mod b$
if $mc = 0$ **then**
    $ls_1 := ls'_0, ls_2 := ls'_1, ls_3 := ls'_2, ls_4 := ls'_3$
    $s_v := s'_v$
end if
if $(ls_3 =$ undefined) **or** $(mc \neq ls_1$ **and** $mc \neq ls_2$ **and** $mc \neq ls_3$ **and** $mc \neq ls_4)$ **then**
    v decides with p_v to send a follower message
    if v sends a follower message **then**
        the message contains:
        $cc_1 := ls'_0, cc_2 := ls'_1, cc_3 := ls'_2, cc_4 := ls'_3$, $c_{new} := c_v, T_{new} := T_v, p_{new} := p_v$
    end if
end if
if v does not send a follower message **then**
    v senses the channel
    if channel is idle **then**
        if $mc = ls_3$ **then**
            $s'_v := 1$
            $p_v := \hat{p}$
        else
            $p_v := \min\{(1+\gamma)p_v, \hat{p}\}$
        end if
    else if v receives 'LEADER' **then**
        $s'_v := 0$
        $ls_3 := undefined$
        $ls'_2 := undefined$
    else if v receives a tuple of $\{cc_1, cc_2, cc_3, cc_4, c_{new}, T_{new}, p_{new}\}$ **then**
        $T_v := T_{new}$
        $p_v := (1+\gamma)^{-1}p_{new}$
        $c_v := c_{new}$
        $ls'_0 := random(0, b-1)$
        $ls'_1 := cc_1, ls'_2 := cc_2, ls'_3 := cc_3, ls'_4 := cc_4$
    end if
end if
$c_v := c_v + 1$
if $c_v \geq b \cdot T_v$ **then**
    $c_v := 0$
    if (**not** CONDITION) **then**
        $p_v := (1+\gamma)^{-1}p_v, T_v := T_v + 1$
        $ls'_0 := undefined, ls'_1 := undefined, ls'_2 := undefined, ls'_3 := undefined, ls'_4 := undefined$
    else
        $T_v := \max\{T_v - 1, 4\}$
    end if
end if

Algorithm 2 Leader Election: Leader

$mc := c_v \mod b$
if $mc = 0$ **then**
    $ls_1 := ls'_1, ls_2 := ls'_2, ls_3 := ls'_3, ls_4 := ls'_4$
end if
if $mc = ls_1$ **or** $mc = ls_2$ **or** $mc = ls_3$ **or** $mc = ls_4$ **then**
    v sends the leader message 'LEADER'
else
    v decides with p_v to send 'LEADER'
    if v does not send 'LEADER' **then**
        v senses the channel
        if channel is idle **then**
            $p_v := \min\{(1+\gamma)^2 p_v, \hat{p}\}$
        else if v receives a message **then**
            $p_v := (1+\gamma)^{-1}p_v$
            if message is 'LEADER' **then**
                $s_v := 0, s'_v := 0$
                $ls_3 := undefined, ls'_2 := undefined$
            else if message is a follower message, i.e., a tuple of $\{cc_1, cc_2, cc_3, cc_4, c_{new}, T_{new}, p_{new}\}$ **then**
                $c_v := c_{new}, T_v := T_{new}$
                $ls'_1 := cc_1, ls'_2 := cc_2, ls'_3 := cc_3, ls'_4 := cc_4$
            end if
        end if
    end if
end if
$c_v := c_v + 1$
if $c_v \geq b \cdot T_v$ **then**
    $c_v := 0$
    if (**not** CONDITION) **then**
        $p_v := (1+\gamma)^{-1}p_v, T_v := T_v + 1$
        $ls'_0 := undefined, ls'_1 := undefined, ls'_2 := undefined, ls'_3 := undefined, ls'_4 := undefined$
    else
        $T_v := \max\{T_v - 1, 4\}$
    end if
end if

Leader Election Protocol

- The protocol is executed in a round based manner. We define one round as a sequence of b time steps for some constant b.
- Each node v maintains
  - probability value p_v ,
  - time window threshold T_v , counter c_v ,
  - node states s_v (s_v=1: leader; s_v=0: follower) and s'_v
  - leader slots:
    current: $ls_1, ls_2, ls_3, ls_4, \in [0,b-1]$
    next round: $ls'_0, ls'_1, ls'_2, ls'_3, ls'_4 \in [0,b-1]$
- $p_{max} < 1/24$
- $\gamma = O(1/(\log T + \log\log n))$

Leader Election Protocol

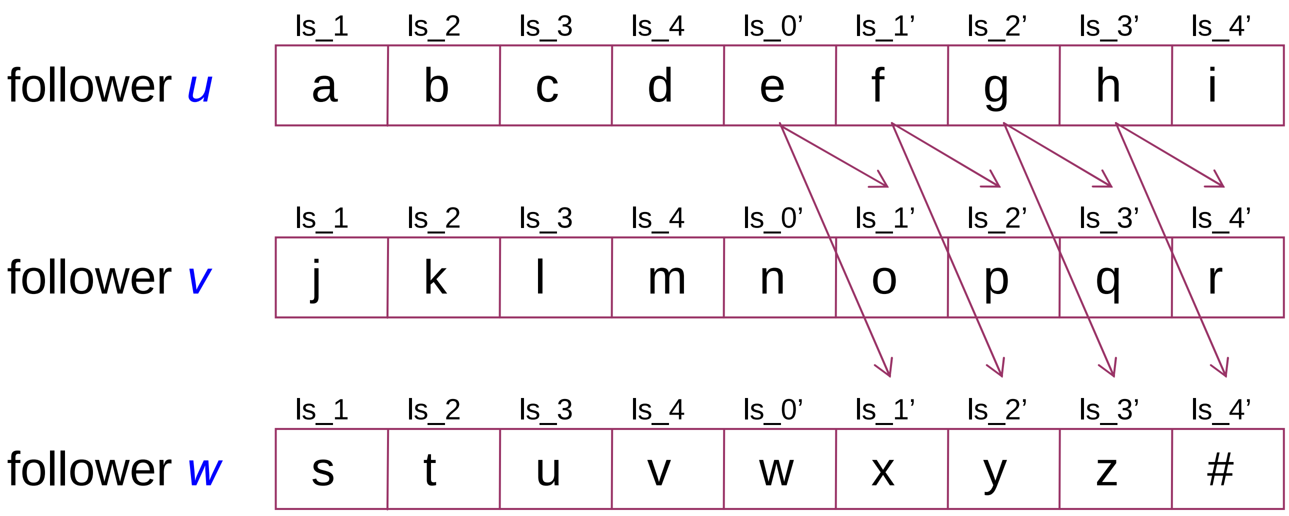
No leader in the network:

	ls_1	ls_2	ls_3	ls_4	ls_0’	ls_1’	ls_2’	ls_3’	ls_4’
follower *u*	a	b	c	d	e	f	g	h	i
follower *v*	j	k	l	m	n	o	p	q	r
follower *w*	s	t	u	v	w	x	y	z	#

- all the values in the leader slots are in [0,b-1]
- the value in ls'_0 is generated randomly and independently by each node.

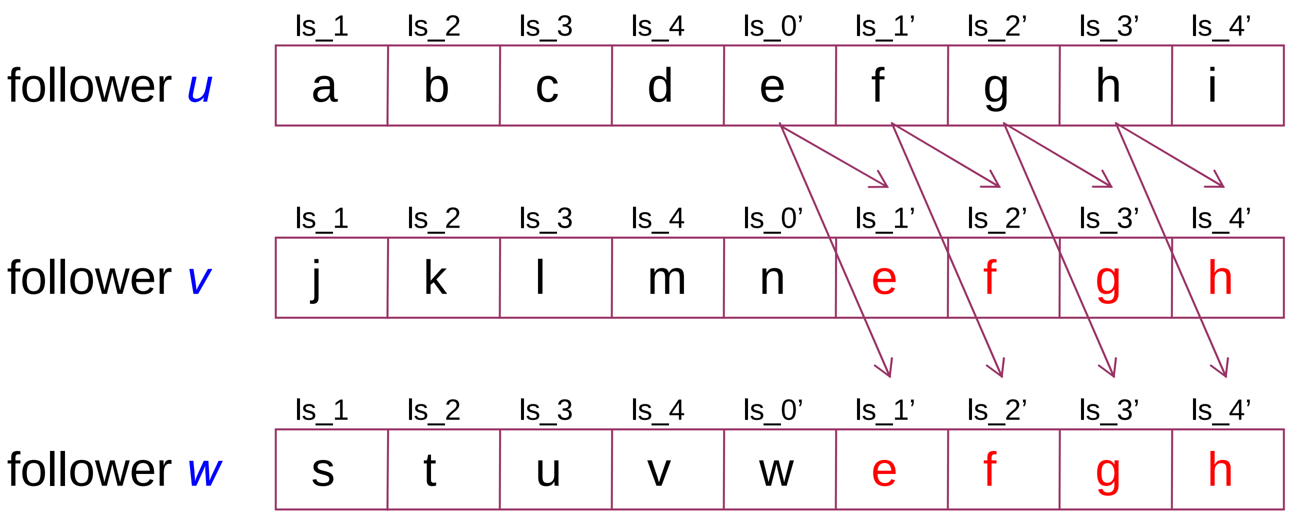
Leader Election Protocol

Suppose now *u* successfully sends a message.



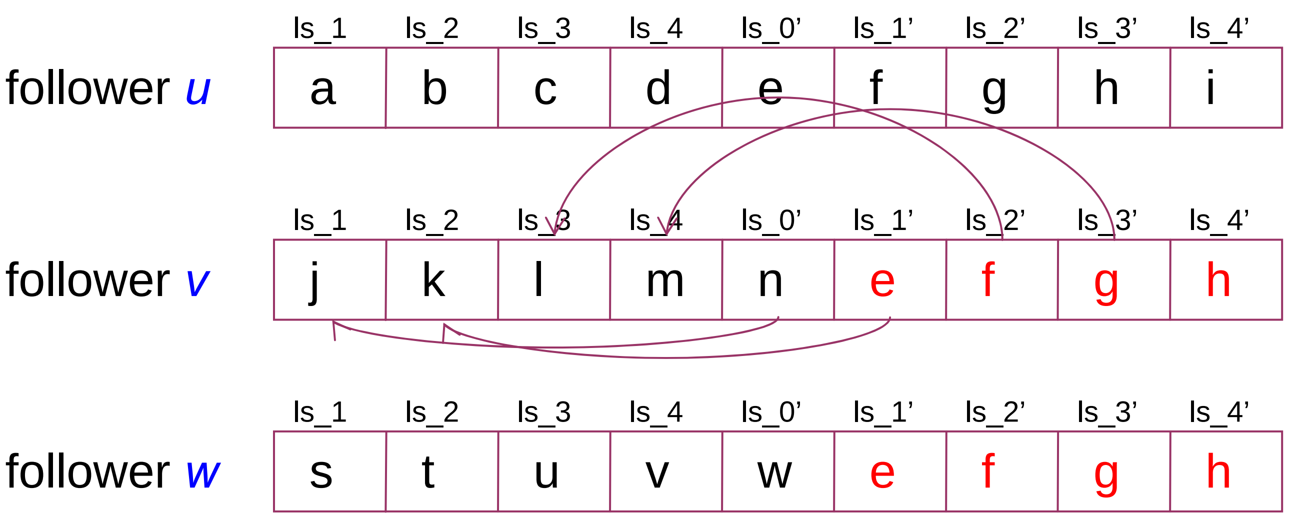
Leader Election Protocol

Suppose now *u* successfully sends a message.



Leader Election Protocol

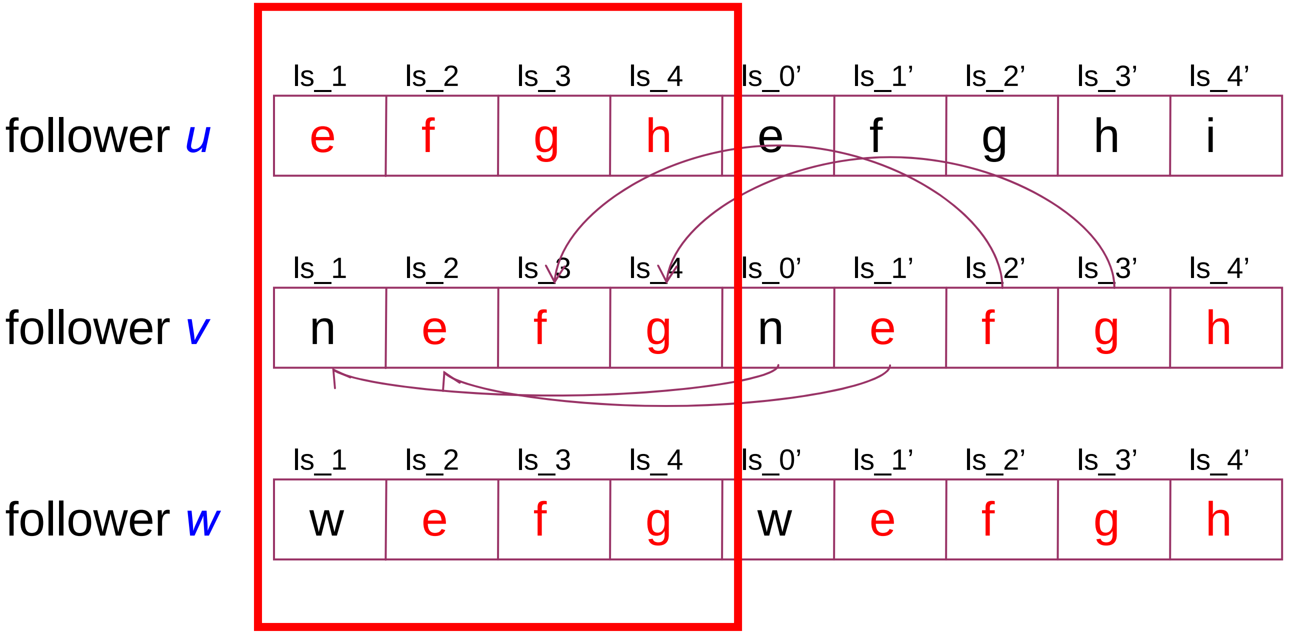
Suppose now *u* successfully sends a message.



Slot values are transferred in the next round.

Leader Election Protocol

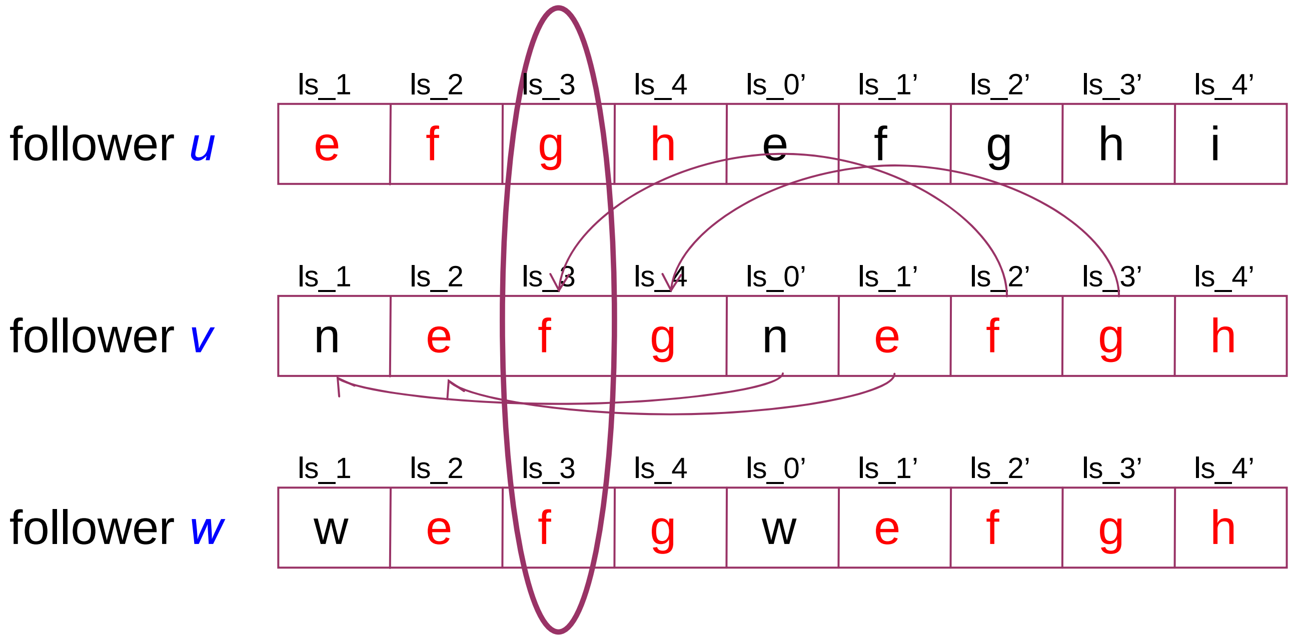
Suppose now *u* successfully sends a message.



Slot values are transferred in the next round.

Leader Election Protocol

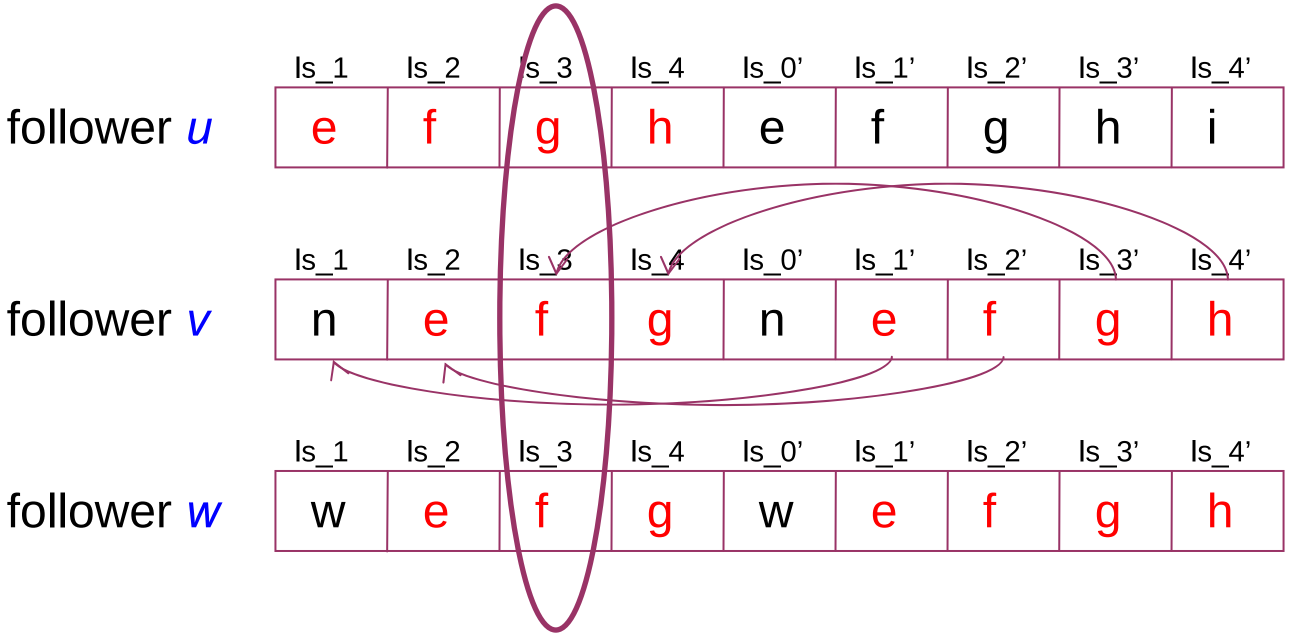
Followers do not transmit in ls slots (if ls_3 defined)



Leaders do transmit in all ls slots

Leader Election Protocol

- Suppose now node *v* and *w* sense an idle channel at time step f. Then *v* and *w* conclude there is no leader in the network, hence they will become a leader at the beginning of next round.



Leader Election Protocol

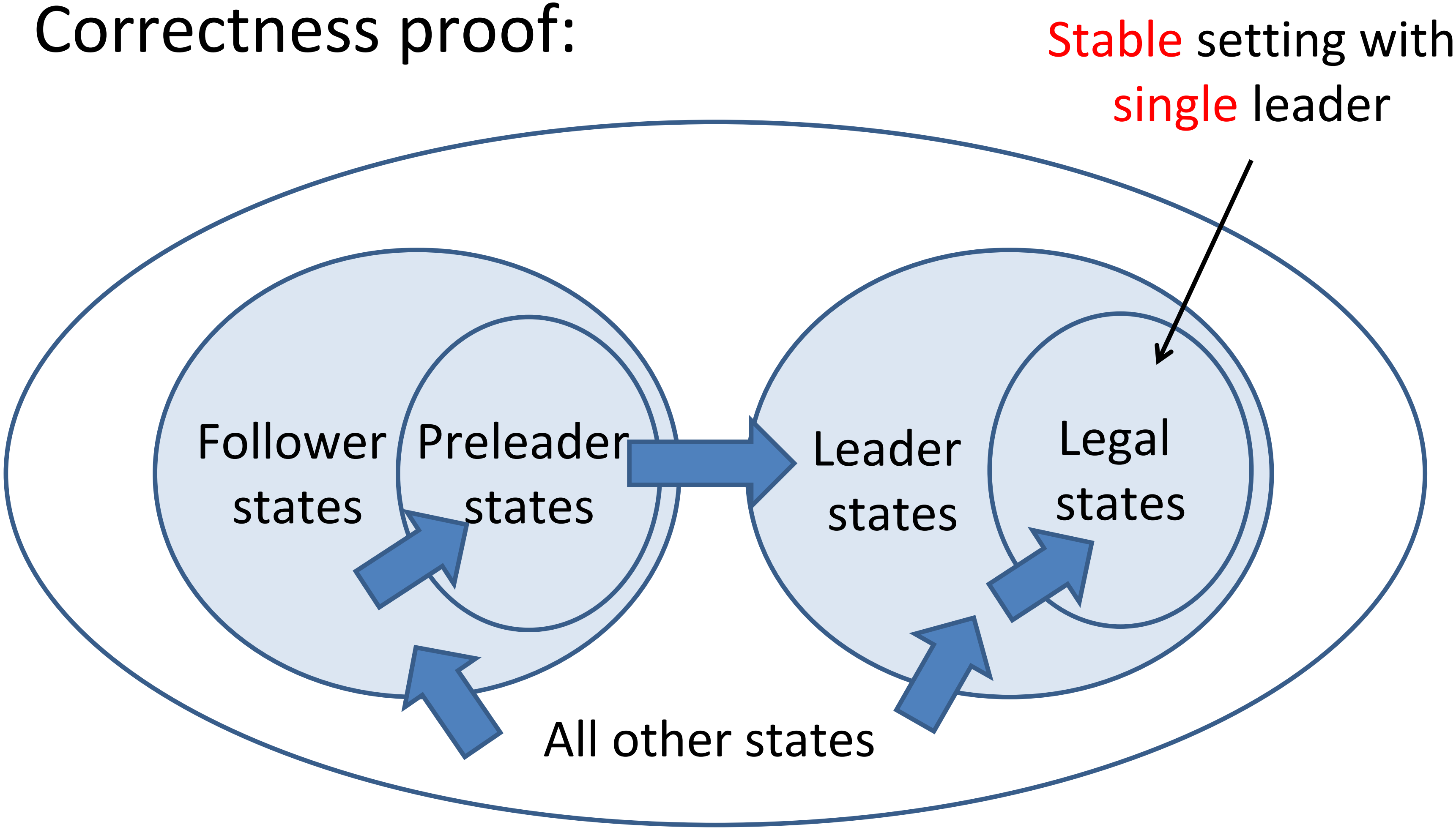
- Soon enough a leader message will get through and make all the remaining nodes followers.
- E.g., leader *v* successfully sends a leader message.

	ls_1	ls_2	ls_3	ls_4	ls_0’	ls_1’	ls_2’	ls_3’	ls_4’
follower *u*	e	f	N/A	h	e	f	N/A	h	i
leader *v*	e	f	g	h	n	e	f	g	h
follower *w*	e	f	N/A	h	w	e	N/A	g	h

- Other nodes invalidate ls_3 and set s'_v to 0 so that they do not become leaders in the next round

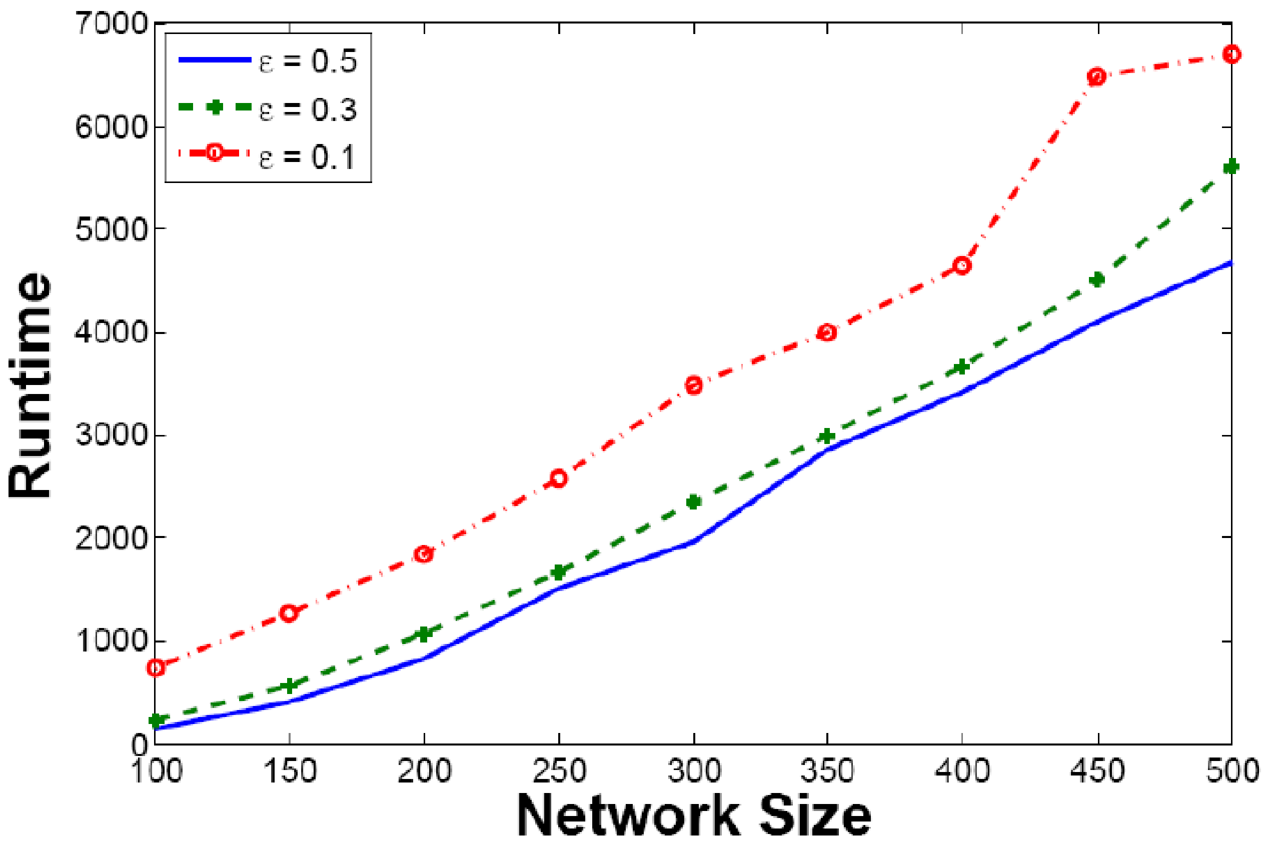
Leader Election Protocol

Correctness proof:



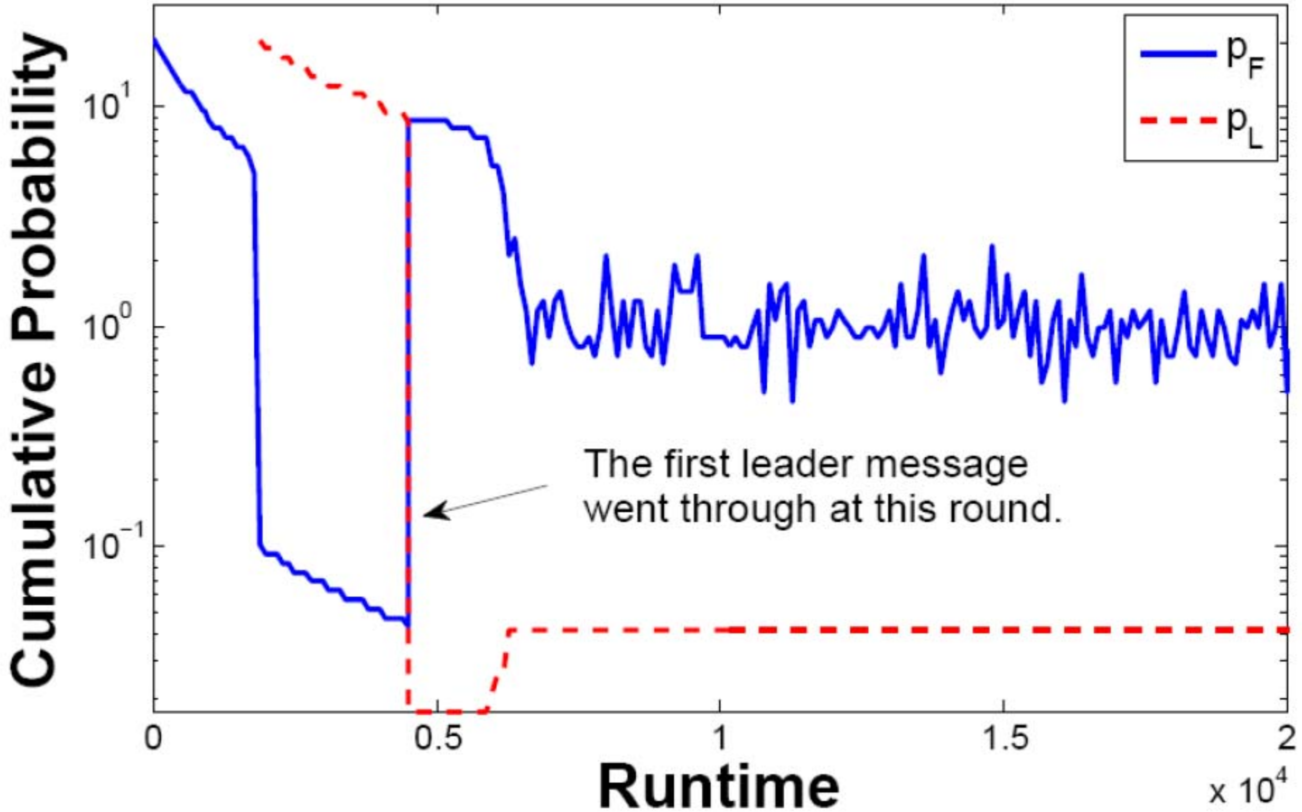
Leader Election

Experiment 1: Runtime



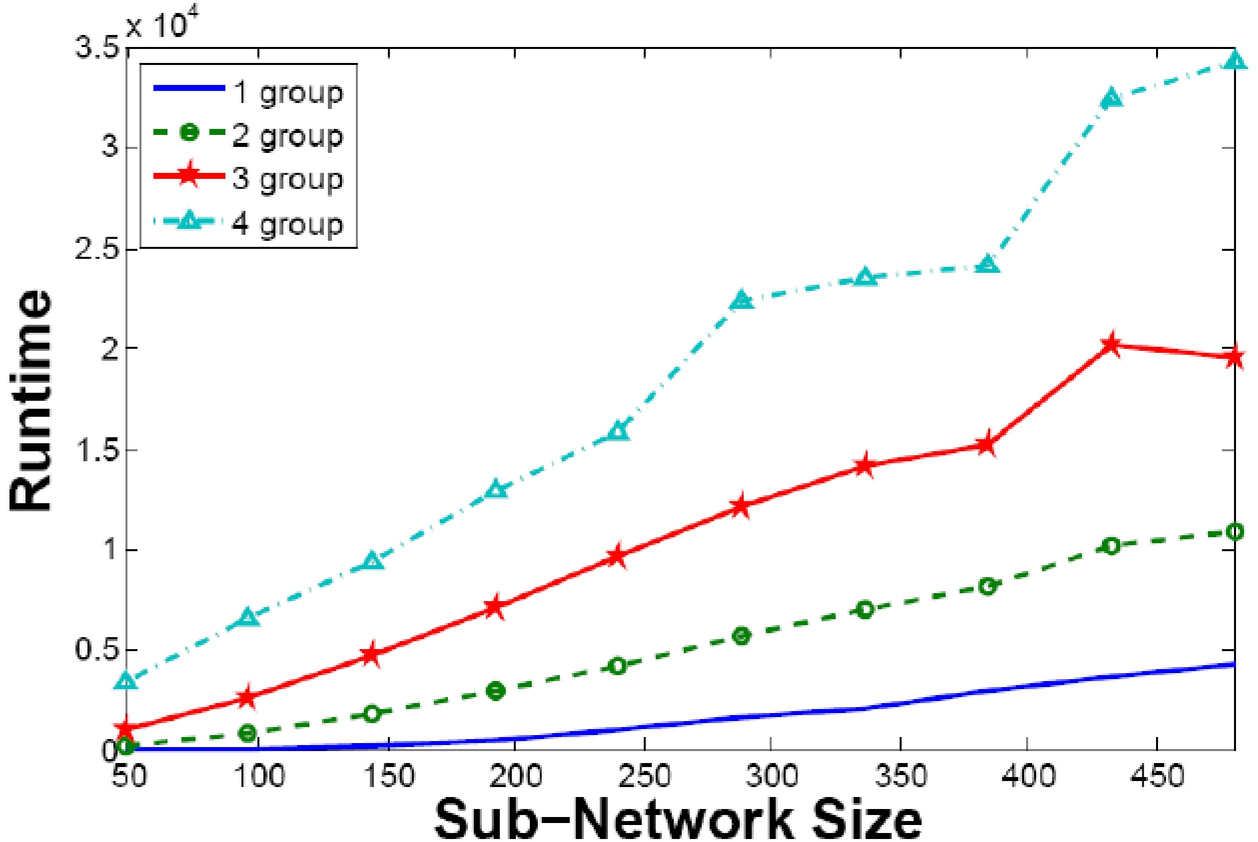
Leader Election

Experiment 2: Cumulative probabilities



Leader Election

Experiment 3: Co-existing networks



Conclusion

- First self-stabilizing leader election protocol that is robust to massive, adaptive jamming
- Experiments show that protocol works

Future work:

- Formal proof of runtime bound
- More efficient protocol
- More realistic communication model
- Other applications