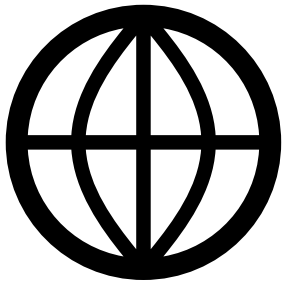
Learning-driven Wireless Networking for Remote Monitoring of Wildlife

Paul Patras - @paulpatras

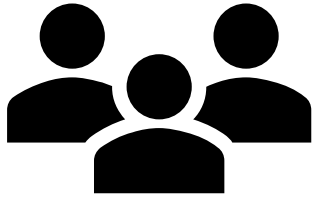
Work with N. Facchi (Adant Technologies),
F. Gringoli (U. Brescia) and D. Malone (NUI)

Image: https://www.flickr.com/people/rayinmanila/

Almost 50% of World's Population Offline



4,223,190,647
Internet users is the world



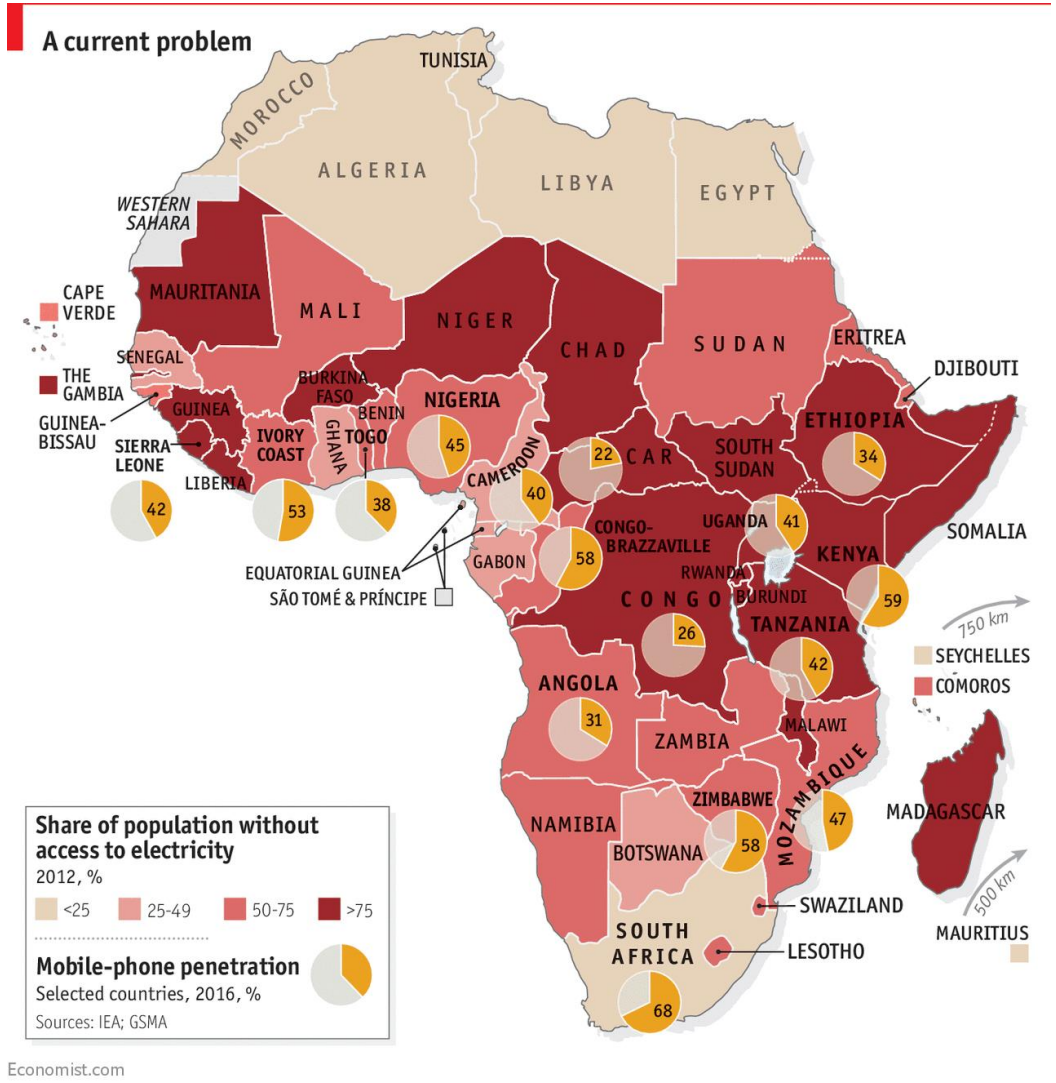
7,703,012,060
World population

Sources: Internet Live Stats, Worldometers

*estimates on 10 May, 2019.

**Individuals who can access the Internet at home, via any device type & connection.

Particularly Problematic in the Global South



Consequences on Wildlife Conservation

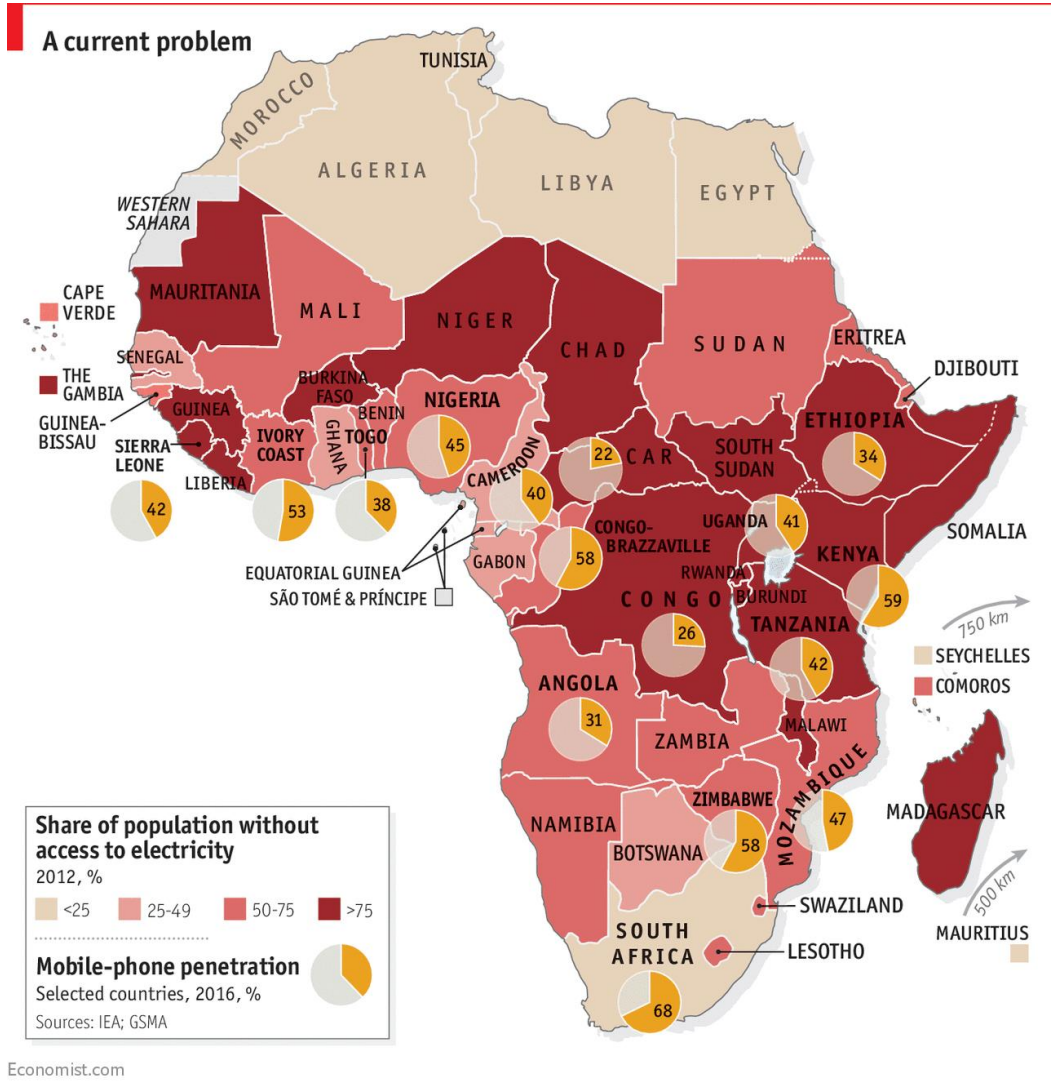


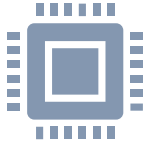
Image: UNICEF Ethiopia

Poverty and Limited Access to Education

Deployment Costs and Skills Shortage

Tackling the Connectivity Problem

Goals:



Inexpensive equipment, ideally powered by renewable energy

Long range, multi-hop comms for sparsely populated regions

Minimal/no management requirements

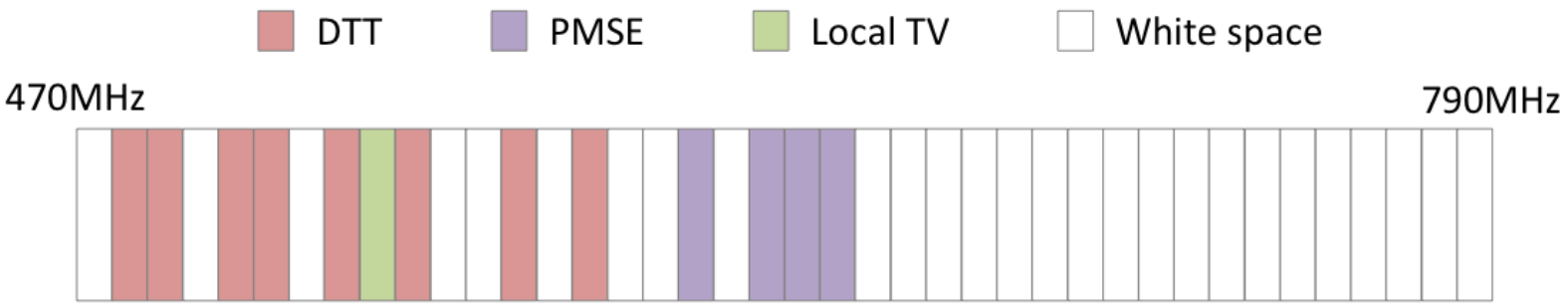
Hardware No Longer a Barrier

Example

€20 embedded module
Ultra-low power
3x networking transceivers
Programmable in micro-python

The TV White Spaces Potential

Sub-GHz bands could bridge digital divide in underdeveloped regions



(+) Long communication ranges, obstacle penetration

(-) Contention for channel time more problematic

Minimal/No Management Requirements

Ideally:

- Medium-high wireless throughput
- Scheduled medium access, without precise synchronisation
- Easy to incorporate new wireless nodes

Minimal/No Management Requirements

Ideally:

- Medium-high wireless throughput
- Scheduled medium access, without precise synchronisation
- Easy to incorporate new wireless nodes

Challenge:

- How to schedule multi-hop transmissions to a wired gateway effectively, in a decentralised way?

Why Multi-hop? Why Decentralised?

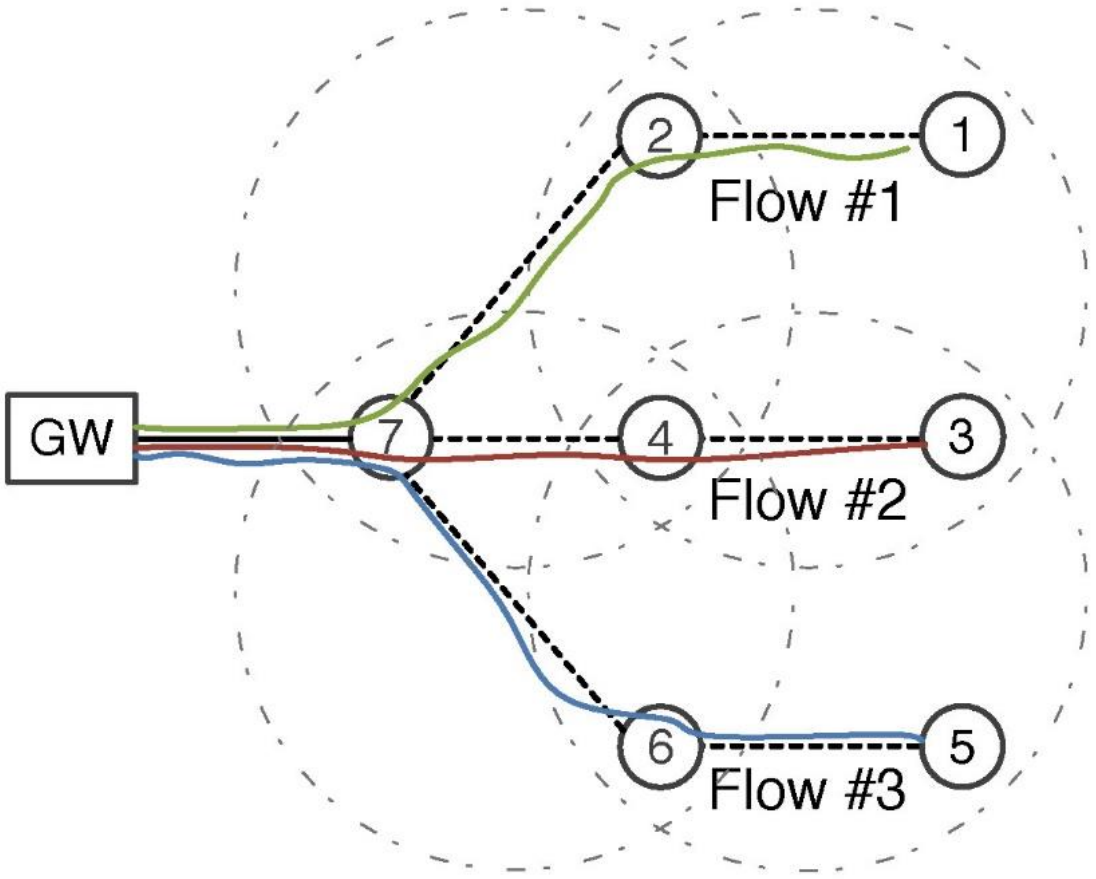
1) Cut network deployment costs
(fewer base stations/relays, wires);

2) Circumvent the need for skilled
network engineers.

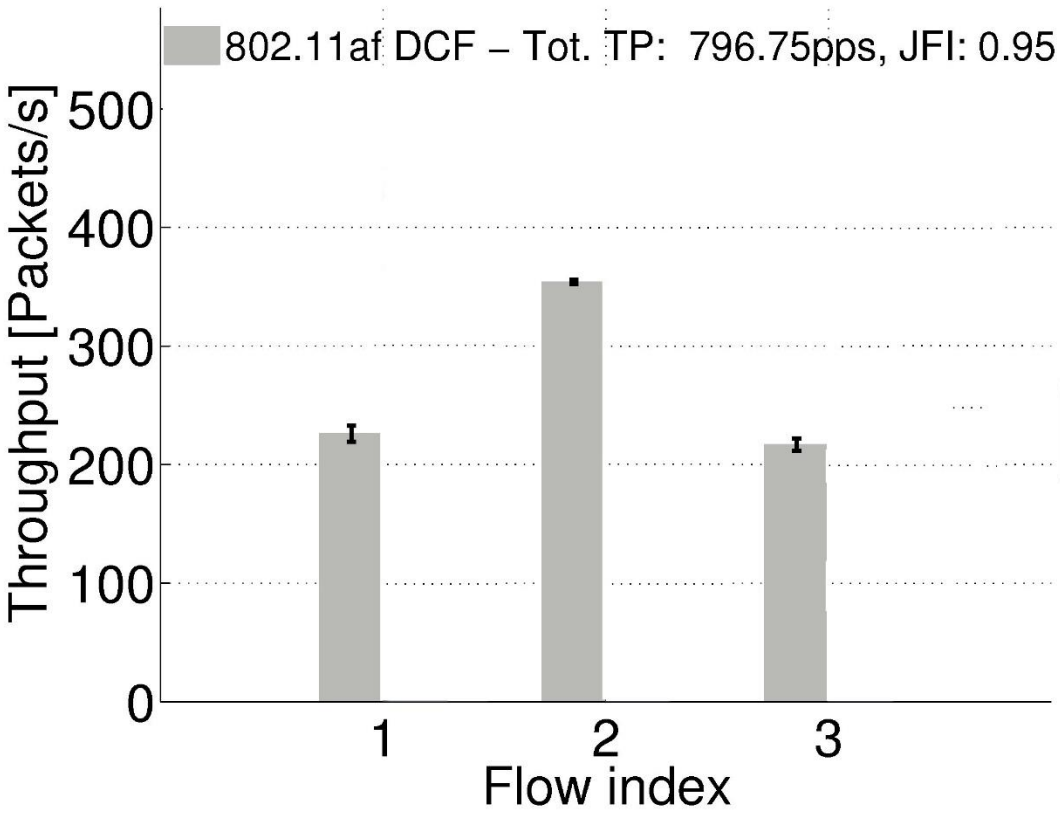
Why Multi-hop? Why Decentralised?

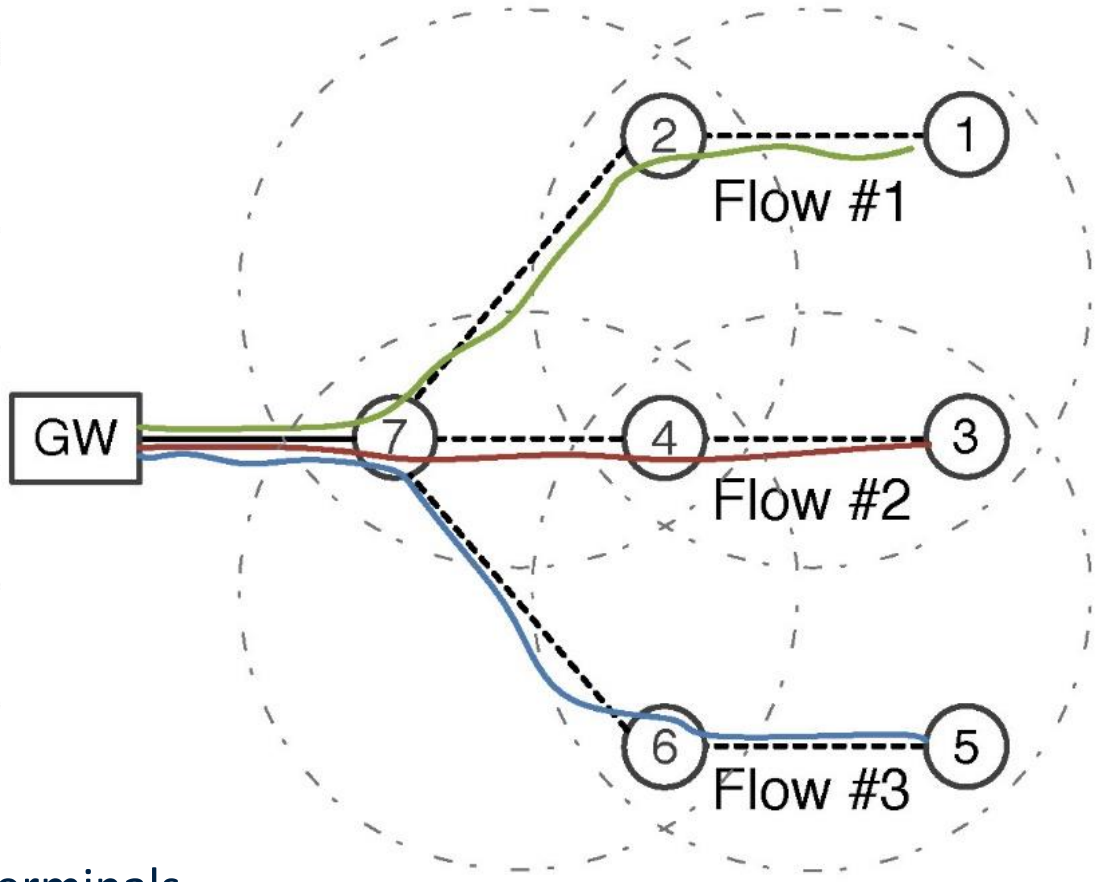
1) Cut network deployment costs (fewer base stations/relays, wires);

2) Circumvent the need for skilled network engineers.



Carrier Sensing Inappropriate

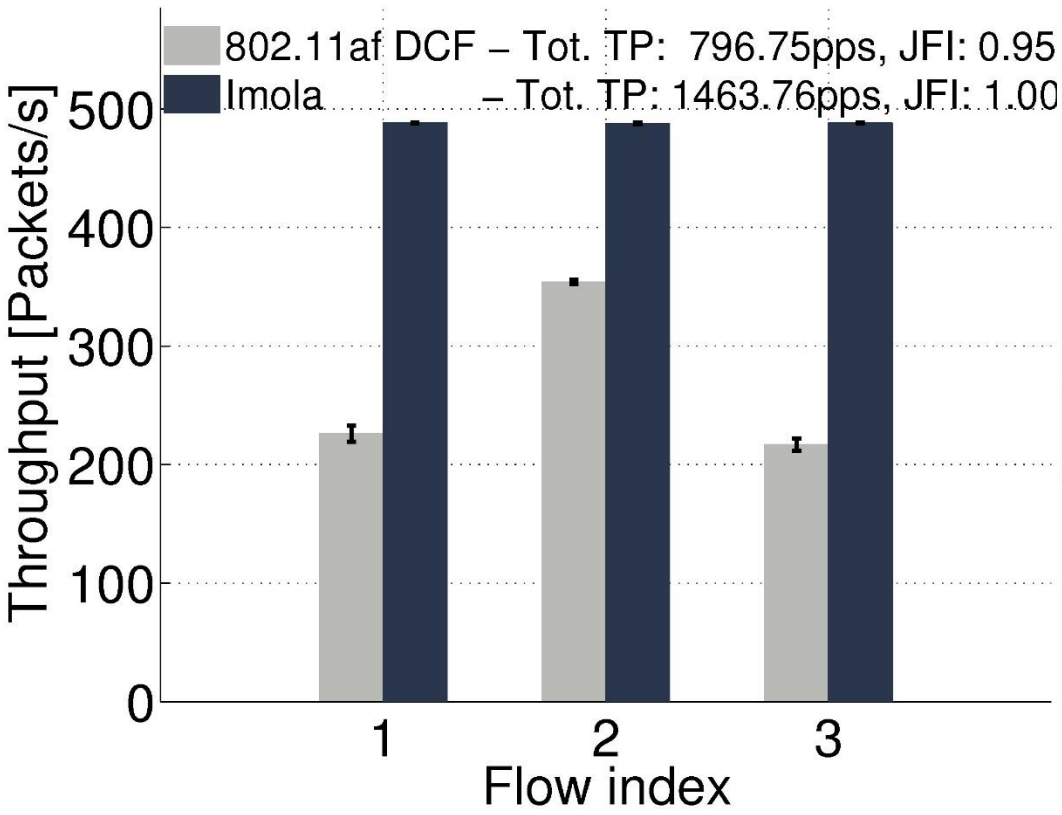


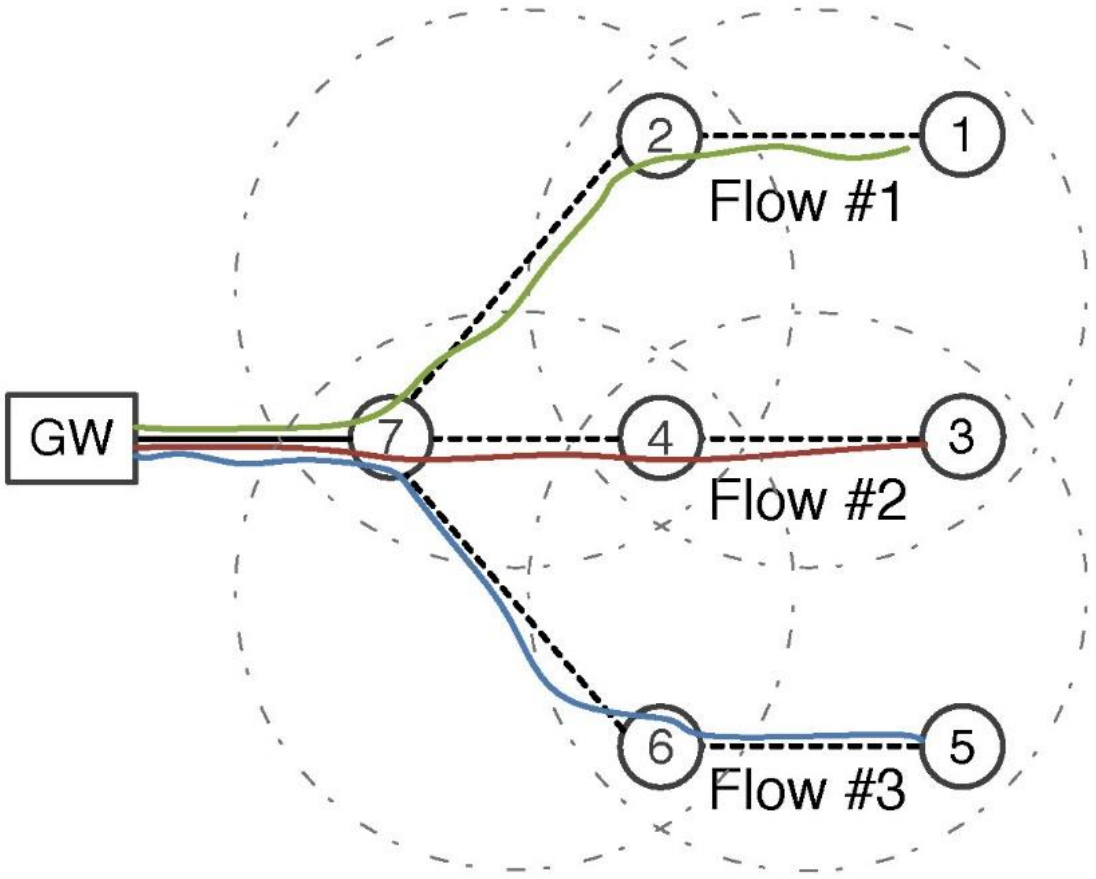


Hidden (e.g. 2 and 6) and exposed (e.g. 1 an 3) terminals

-> uneven flow throughput distribution

Decentralised Learning-driven Scheduling



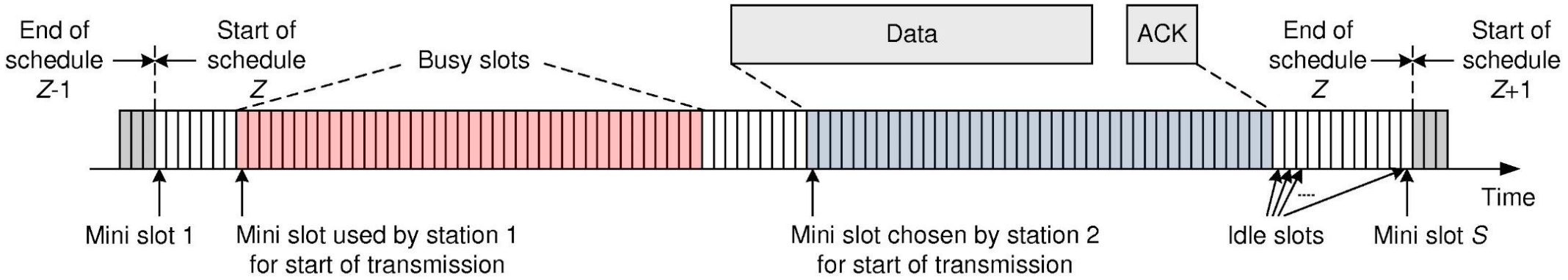


The Imola Solution in a Nutshell

- Disable carrier sensing, acquire local neighbourhood info
- Simple learning to infer when is the right time to transmit and avoid collisions (pseudo-scheduled operation)
- Rapid convergence
- Implementable with commodity hardware
- Fair distribution of resources and high total throughput

*N. Facchi, F. Gringoli, D. Malone, P. Patras, Computer Communications, 2016

System Design



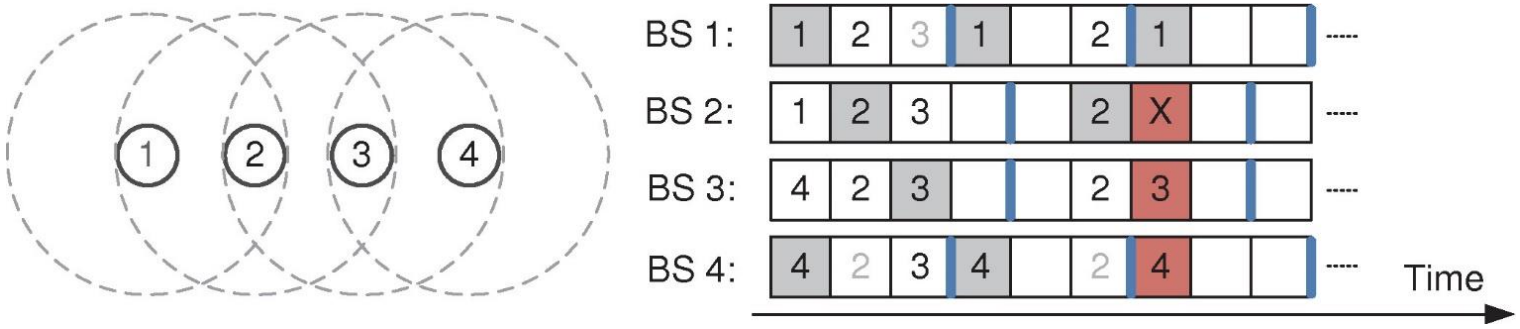
- Channel divided into *mini-slots* of short duration
- Stations observe schedules of *S* mini-slots, proportional to their (one- and two-hop) neighbours count *n*
- A transmission (including ACK) spans *T* slots

Schedule Length Computation

- Node joining listens for T_{scan} and estimates the number neighbours, n_i, by source/destination addresses of overheard frames.

Schedule Length Computation

- Node joining listens for T_{scan} and estimates the number neighbours, n_i, by source/destination addresses of overheard frames.



- Single hop is straightforward -> set $S = N \times T$
- Things difficult in multi-hop, as stations may observe different numbers of neighbours and periodically drift into collision
- 1 and 4 observe 3 one-and two-hop neighbours; 2 and 3 observe 4.

Schedule Length Computation

Solution: choose schedule lengths that are multiple of each other

Schedule Length Computation

Solution: choose schedule lengths that are multiple of each other

- Establishing this precisely requires global topology knowledge
  -> **expensive!**

Schedule Length Computation

Solution: choose schedule lengths that are multiple of each other

- Establishing this precisely requires global topology knowledge

  -> **expensive!**

- Choose length as closest power of 2 that can accommodate neighbours, i.e.

$$S_i = 2^{\lceil \log_2 n_i \rceil}(T + \varepsilon)$$

- If station does not settle into collision-free operation, double S_i e.g. up to $S_{max} \leq 30$ms (can accommodate up to N = 31 two-hop neighbours transmitting MTU size packets at 16Mb/s).

Channel Access Protocol

- Recall carrier sensing disabled, channel divided into mini slots;
- $p_k(t)$ probably of choosing slot k for a new transmission attempt t.

- Intialise all probabilities with

$$p_k(0) = \frac{1}{S}, \forall\, k \in \{1, \dots, S\}$$

- Upon unsuccessful transmission at slot j, update all slot probabilities

$$p_k(t+1) = \alpha p_k(t) + (1-\alpha)\frac{2^{\|k-j\|}}{3(2^{S/2}-1)}$$

Channel Access Protocol

- Rationale

$$p_k(t+1) = \alpha p_k(t) + (1-\alpha)\frac{2^{\|k-j\|}}{3(2^{S/2}-1)}$$

Channel Access Protocol

- Rationale

$$p_k(t+1) = \alpha p_k(t) + (1-\alpha)\frac{2^{\|k-j\|}}{3(2^{S/2}-1)}$$

In (0,1) - control
stickiness / learning rate

Channel Access Protocol

- Rationale

$$p_k(t+1) = \alpha p_k(t) + (1-\alpha)\frac{2^{\|k-j\|}}{3(2^{S/2}-1)}$$

In (0,1) - control stickiness / learning rate

Distance between *j* and *k* mod *S* – give more weight to slots farther from *j*

Channel Access Protocol

- Rationale

$$p_k(t+1) = \alpha p_k(t) + (1-\alpha)\frac{2^{\|k-j\|}}{3(2^{S/2}-1)}$$

In (0,1) - control stickiness / learning rate

Distance between *j* and *k* mod *S* – give more weight to slots farther from *j*

Term used to ensure probabilities add up to one and never go below a minimum value $\gamma = \frac{1}{3(2^{S/2}-1)}$

Channel Access Protocol

- Finally, if node transmits successfully in slot *j*

$$p_j(t+1) = 1;$$
$$p_k(t+1) = 0, \forall k \neq j.$$

- Maintain these until node experiences a collision (deterministic behaviour); then restart learning phase.

Schedule Length Adaptation

- What about maintaining efficiency when some nodes are switched off?

- Imola periodically seeks to reduce the schedule length by half to explore more throughput efficient cycles.

Schedule Length Adaptation

- What about maintaining efficiency when some nodes are switched off?
- Imola periodically seeks to reduce the schedule length by half to explore more throughput efficient cycles.
- How frequently should this exploration be performed?
- If attempting too frequently when there is no scope for shorter schedules, throughput performance could be harmed due to repeated collisions.

Schedule Length Adaptation

- What about maintaining efficiency when some nodes are switched off?
- Imola periodically seeks to reduce the schedule length by half to explore more throughput efficient cycles.
- How frequently should this exploration be performed?
- If attempting too frequently when there is no scope for shorter schedules, throughput performance could be harmed due to repeated collisions.
- We will work with a frequency f_i that ensures the opportunity cost of trying $S_i' = S_i/2$ is below 5%.

Prototype Implementation

Implemented Imola with Broadcom BCM4318 wireless cards.

1) MAC CPU controls real-time operation through firmware -> essential to meet fine timing requirements
2) Highly programmable -> OpenFWWF firmware developed at Brescia

Prototype Implementation

Implemented Imola with Broadcom BCM4318 wireless cards.

1)MAC CPU controls real-time operation through firmware -> essential to meet fine timing requirements

2)Highly programmable -> OpenFWWF firmware developed at Brescia

(-) Do not work in sub-gigahertz bands, but sufficient for proof-of-concept

(+) We can interrupt ongoing receptions and start transmission at precise time instances, independent of channel state.

Source at: http://netweb.ing.unibs.it/openfwwf/IMOLA/

Updating Slot Probabilities

- Simple 88MHz CPU with fixed-point ALU capable only of addition, subtraction and shift operations.
- Work only with powers of two for the parameters -> obtain current slot and TX time indicator from real time clock register only with shift and mask operations.

Updating Slot Probabilities

- Simple 88MHz CPU with fixed-point ALU capable only of addition, subtraction and shift operations.
- Work only with powers of two for the parameters -> obtain current slot and TX time indicator from real time clock register only with shift and mask operations.
- For updating probabilities, we transform the division by 3 operation and use

$$\frac{2^{\|k-j\|}}{3} = 2^{\|k-j\|-16} \cdot 2^{16}/3 \approx 2^{\|k-j\|-16} \cdot 0\text{x}5555$$

Updating Slot Probabilities

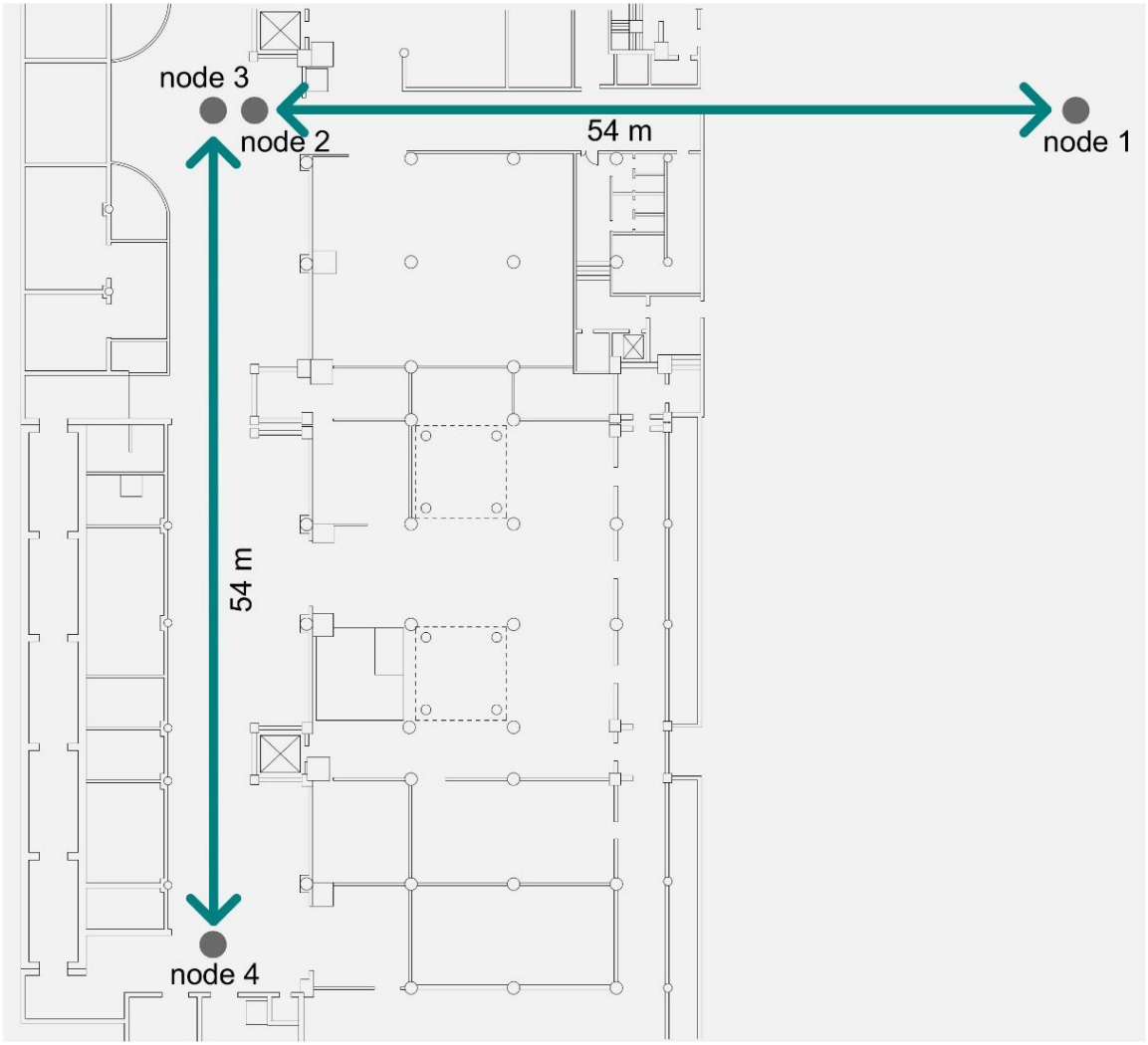
- Simple 88MHz CPU with fixed-point ALU capable only of addition, subtraction and shift operations.
- Work only with powers of two for the parameters -> obtain current slot and TX time indicator from real time clock register only with shift and mask operations.
- For updating probabilities, we transform the division by 3 operation and use

$$\frac{2^{\|k-j\|}}{3} = 2^{\|k-j\|-16} \cdot 2^{16}/3 \approx 2^{\|k-j\|-16} \cdot 0\text{x}5555$$

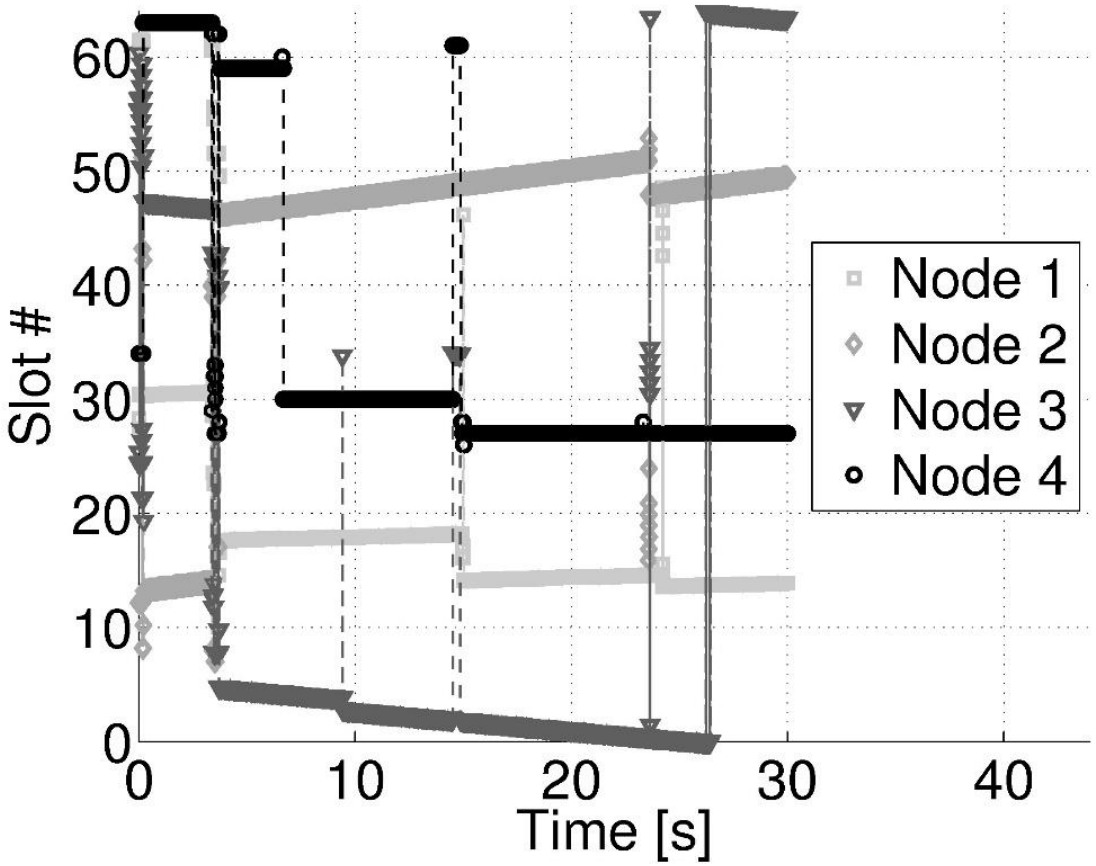
- Plus some changes in kernel to enable direct-link type operation and LLC based tracing and debugging.

Experiments

- 4 Linux embedded PC in basement at U. Brescia.
- Pathological hidden and exposed terminals, 4 flows.
- UDP traffic using iperf tool.
- tcpdump to monitor slot selection in LLC headers.
- Compare against DCF.

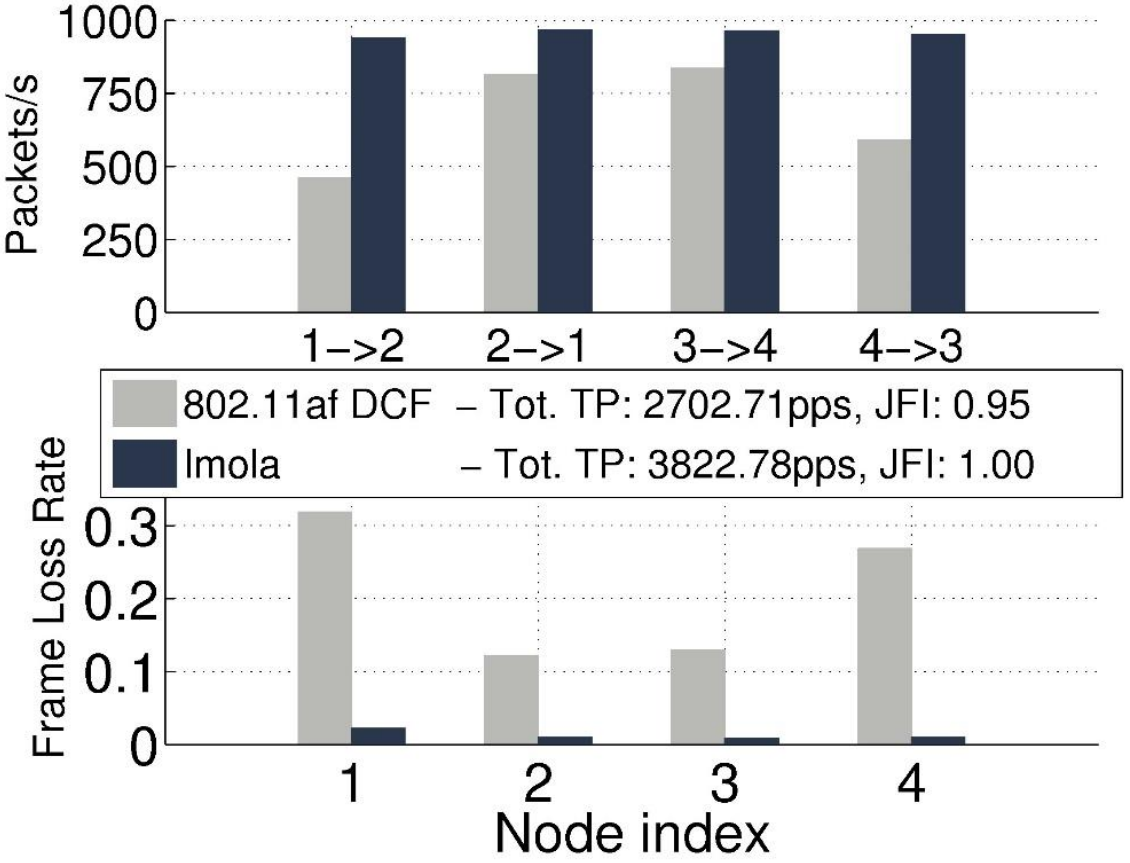


Convergence



- Despite clock drift and lack of precise synchronisation, Imola reaches convergence quickly.
- Maintains steady collision-free operation, only occasionally changing slots due to clock drift.

Practical Performance Gains

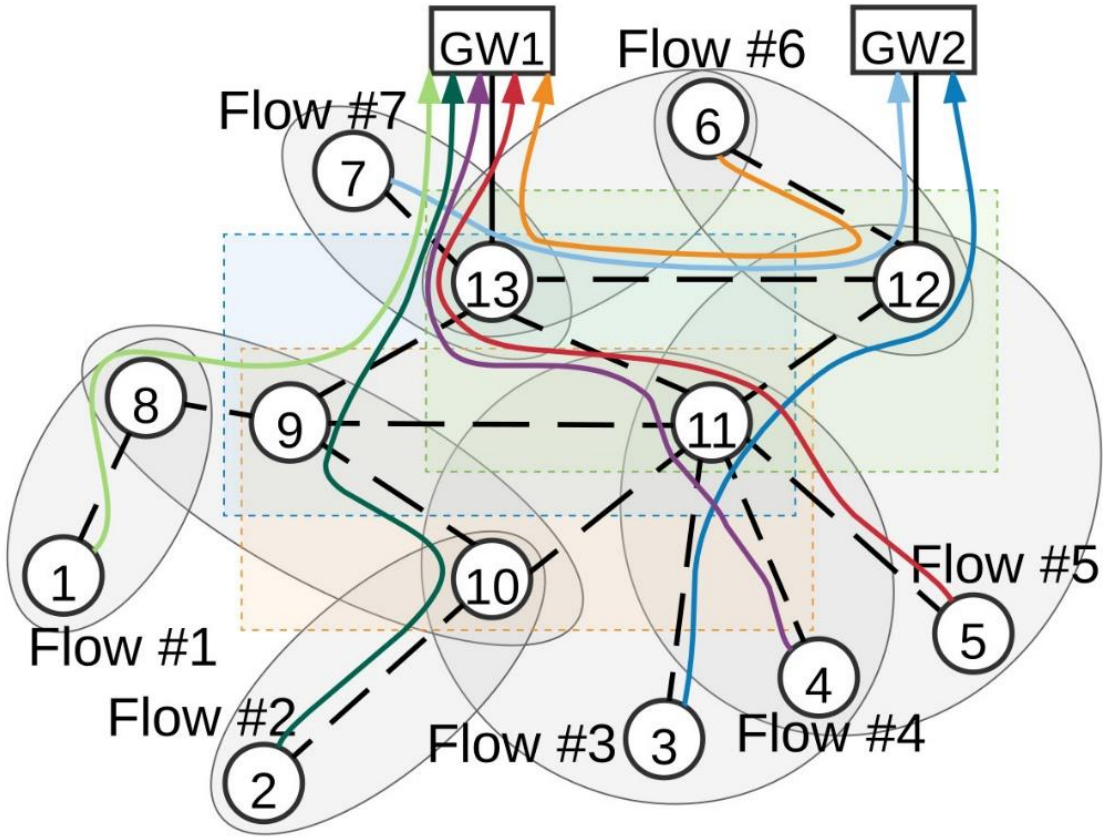


- Imola overcomes hidden and exposed terminal problems; can double individual throughputs; preserves fairness (JFI=1).
- Eliminates almost completely frame loss rate specific to CSMA.

Performance at Scale

- NS-3 implementation to evaluate for more complex network topologies.
- Mini-slot duration 16μs.
- Transmission budget T = 240 μs per schedule.
- Stations always have 1000-byte packets to transmit at 54 Mb/s.
- Losses only due to collisions; negligible clock skews.

Performance at Scale

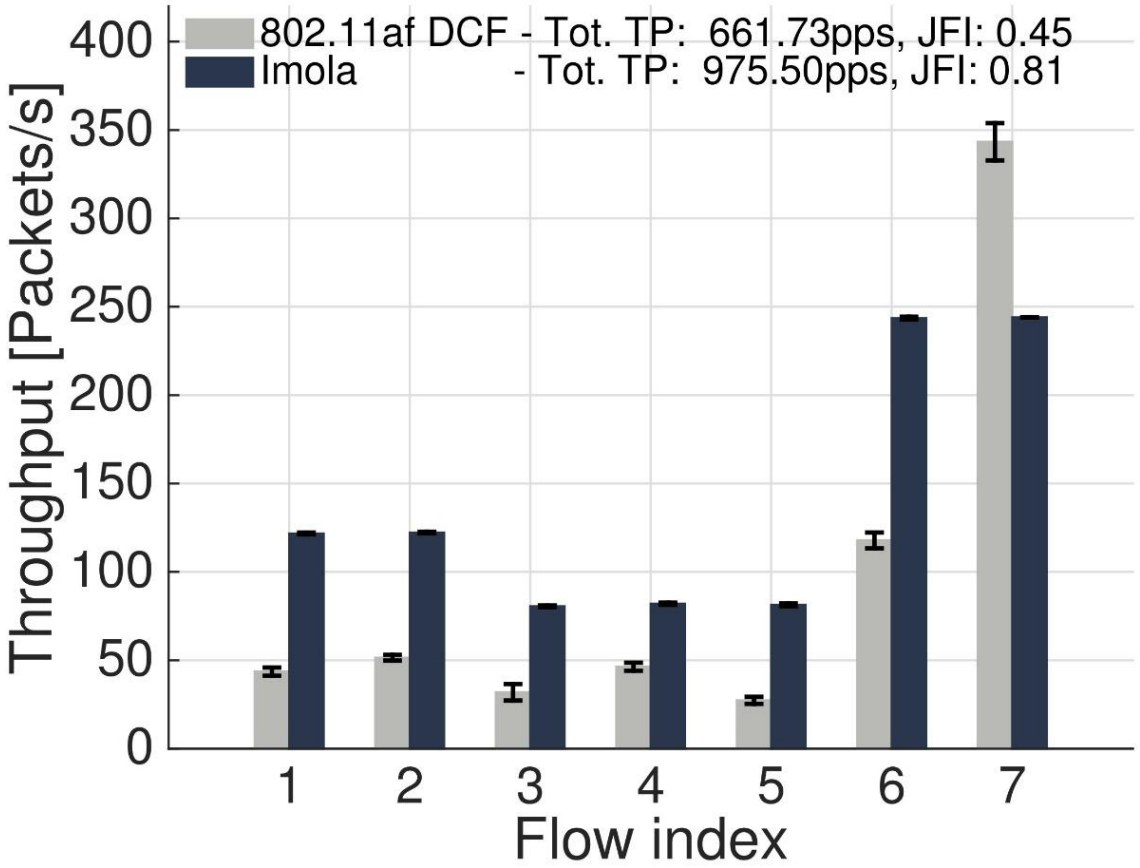


13 nodes, two gateways, 7 flows

Derived from community based networks

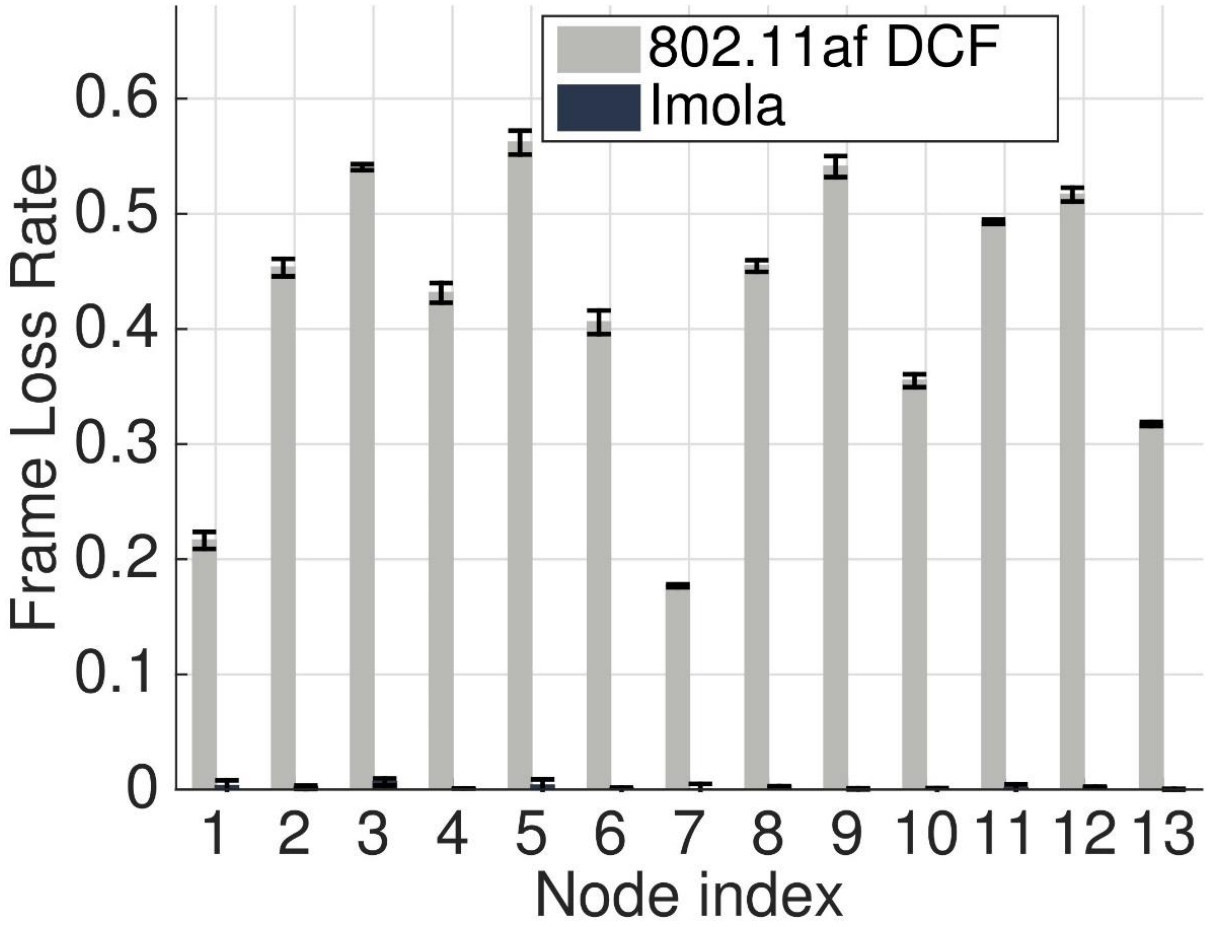
Complex overlap of carrier sensing ranges

Throughput Performance



- Imola doubles throughput performance of all flows, except for that of Flow #7.
- Flow #7 experiences 25% throughput reduction, but still superior, and fair with #6.

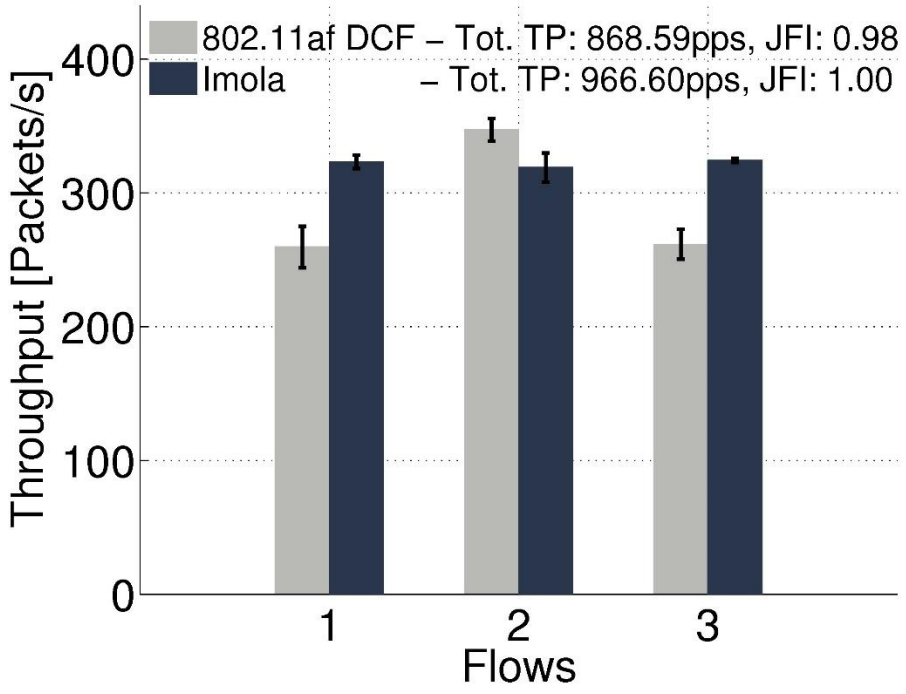
Frame Loss

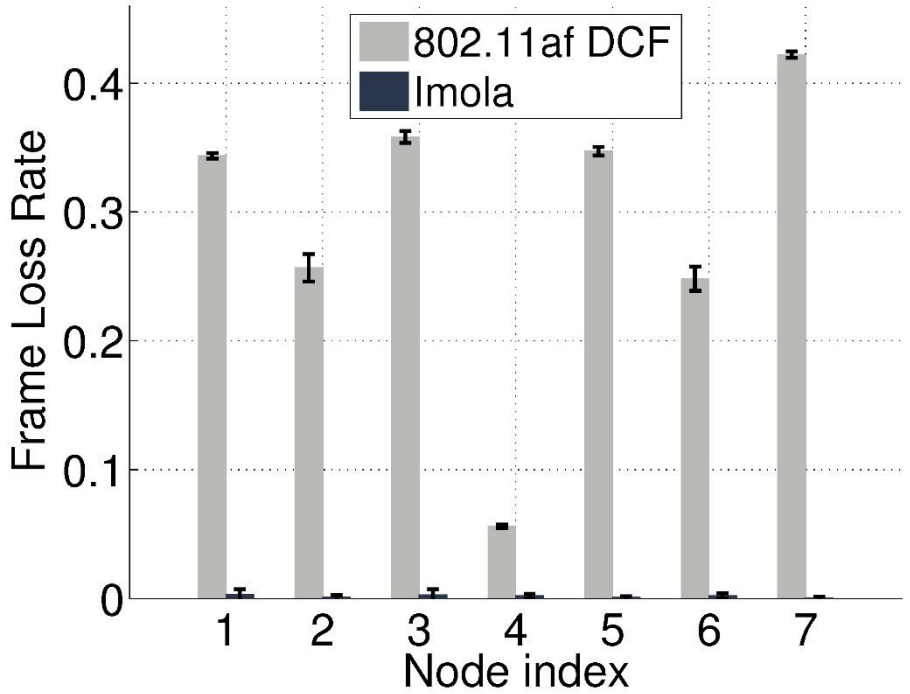


50% total throughput gain due to dramatic frame loss rate reduction at all nodes.

TCP Traffic

3-branch tree topology





≈12% more throughput, equalises flows (JFI=1), reduces frame loss rate dramatically

Further improvements possible, by e.g. piggybacking TCP ACKS on data link ACKs.

Conclusions

- Introduced decentralised protocol for multi-hop White-Fi networks that circumvents the performance conundrums specific to CSMA.
- Use reinforcement learning to achieve scheduled-like operation, without synchronisation and periodic exchange of topology info.
- Proof-of-concept implementation on commodity hardware and demonstrated practical feasibility through experiments.
- Up to 4x more throughput than the standard 802.11af DCF in simulation, reducing frame error rate by up to 100%.
- Works well with TCP but room for improvements.