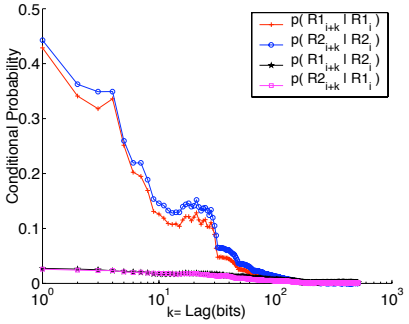
Errors are bursty and independent



[slide: A. Miu, MobiCom '05]

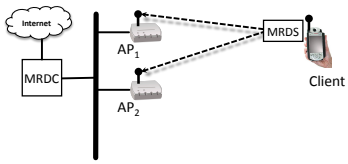
Challenges in developing MRD

- How to correct simultaneous frame errors?
  - Frame combining
- How to handle retransmissions in MRD?
  - Request-for-acknowledgment protocol
- How to adapt bit rates in MRD?
  - MRD-aware rate adaptation

[slide elements: A. Miu, MobiCom '05]

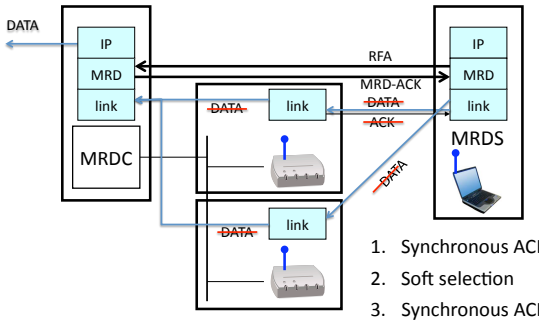
Flawed retransmission schemes

- Conventional link-layer ACKs do not work
  - Final status known only to MRDC
- Two levels of ACKs are high overhead
- Cannot disable link-layer ACKs



[slide elements: A. Miu, MobiCom '05]

MRD acknowledgement protocol



1. Synchronous ACK
2. Soft selection
3. Synchronous ACK loss
4. Frame combining

[slide: A. Miu, MobiCom '05]

Bit-by-bit frame combining

Combine failure

TX: 1100 1010

1100 0000 R1
⊕ 1101 1010 R2
0001 1010

1. Locate bits with unmatched value

Patterns	CRC Ok	
1100 0000	X	
1100 0010	X	
1100 1000	X	
1100 1010	O	→ Corrected frame

2. Select bit combination at unmatched bit locations, check CRC

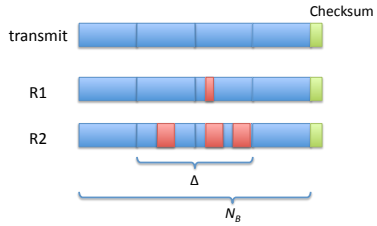
Problem: Exponential # of CRC checks in # of mismatched bits.

[based on slide: A. Miu, MobiCom '05]

Block-based frame combining

- Recall: bit errors occur in bursts
- Divide frame into N_B blocks (e.g., $N_B = 6$)
- Attempt recombination with all possible block patterns until CRC passes
  - ✓ # of checks upper bounded by 2^{NB}
  - ✗ Failure rate increases with N_B

Block based frame combining

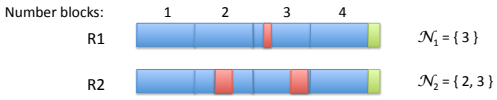


2^{Δ} combinations of Δ differing blocks

$\Delta < N_B$

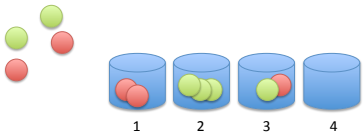
Main issue: what is probability of failure: p_f?

Frame combining failure event



$$\begin{cases} |\mathcal{N}_1 \cap \mathcal{N}_2| > 0 & \Rightarrow \textbf{failure} \\ |\mathcal{N}_1 \cap \mathcal{N}_2| = 0 & \Rightarrow \text{success} \end{cases}$$

Balls and bins analogy



Throw d_1 identical red, d_2 identical green balls in N_B (distinguishable) bins at random

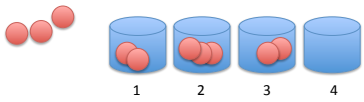
place red ball in bin b :: bit error in block b of AP_1

place green ball in bin b :: bit error in block b of AP_2

frame combining error :: ???

Stars and bars analogy

How many ways to throw d identical balls into N_B (distinguishable) bins?



|*|**|

As many as there are star-bar strings!

$$\binom{d + N_B - 1}{N_B - 1} = \binom{N_B + d - 1}{d}$$

Conditional probability of combining error

Suppose we have d_1 errors in received transmission at AP_1, d_2 errors in received transmission at AP_2.

$$p_f(d_1,d_2) \le N_B \frac{\binom{N_B + (d_1 - 1) - 1}{d_1 - 1}\binom{N_B + (d_2 - 1) - 1}{d_2 - 1}}{\binom{N_B + d_1 - 1}{d_1}\binom{N_B + d_2 - 1}{d_2}}$$

choose a bin to have one green, one red

Place remaining d_1-1 red balls

Place remaining d_2-1 green balls

Probability of failure given d_1 errors at AP_1, d_2 errors at AP_2

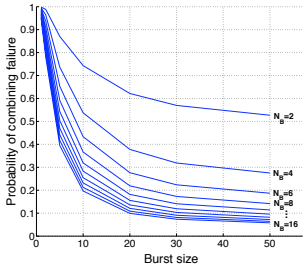
Place d_1 red balls

Place d_2 green balls

[graph: A. Miu, MRD '05]

Probability of frame combining failure

$$p_f \le \sum_{d_1=1}^{N_B} \sum_{d_2=1}^{N_B} p_f(d_1,d_2) \Pr(d_1 \text{ errors at } AP_1, d_2 \text{ errors at } AP_2)$$



[slide: A. Miu, MobiCom '05]

MRD-aware rate adaptation

- Standard rate adaptation does not work
  - Reacts only to link-layer losses from 1 receiver

  ➔Uses sub-optimal bit-rates
- MRD-aware rate adaptation
  - Reacts to losses at the MRD-layer

Implication: First use multiple paths, *then* adapt bit rates.

[Graphic: Bicket, MS thesis '05, algorithm: http://madwifi-project.org/browser/madwifi/trunk/ath_rate/onoe/onoe.c?rev=3902]

ONOE (madwifi) + MRD rate adaptation

Algorithm (Atsushi Onoe)

repeat every second, per-destination:
if num_tx > 0 and num_tx_ok = 0
  then *current_bitrate* ↓; init(); **return**;
if *num_tx* > 10 **and** *num_tx_ok / num_tx* < 0.5
  then *current_bitrate* ↓; init(); **return**;
if *num_tx_ok / num_tx* < 0.1 **then** *credits*−−; **return**;
if *num_tx_ok / num_tx* > 0.9 **then**
  credits++;
  if *credits* > 10 **then** *current_bitrate* ↑; init();

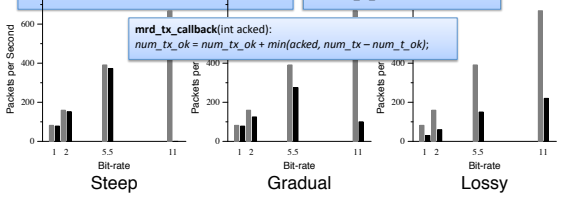
init():
num_tx = num_tx_ok = 0;
credits = 0;

State (per-dest)

credits
current_bitrate
num_tx
num_tx_ok

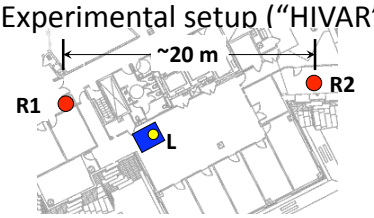
tx_callback(bool *success*, int *txcount*):
num_tx = *num_tx* + *txcount*;
if *success* **then**
  num_tx_ok++;

mrd_tx_callback(int acked):
num_tx_ok = *num_tx_ok* + *min(acked, num_tx – num_t_ok)*;



[slide: A. Miu, MobiCom '05]

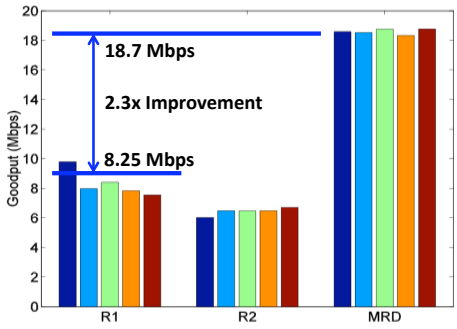
Experimental setup ("HIVAR")



- 802.11a/b/g implementation in Linux (MADWiFi)
- L transmits 100,000 1,500 byte UDP packets w/7 retries
- 802.11a @ auto bit rate (6, 9, 12, 18, 24, 36, 48, 54)
- L is in motion at walking speed, > 1 minute per trial
- Variants: R1, R2, MRD (5 trials each)

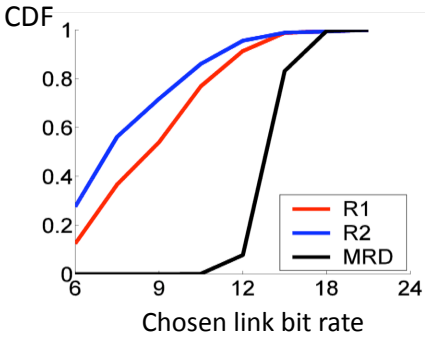
[slide: A. Miu, MobiCom '05]

MRD throughput improvement

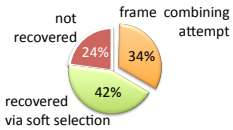


[graphic: A. Miu, MobiCom '05]

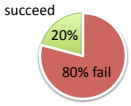
MRD maintains high bit rate



Closer look at frame recovery method



Breakdown of lost frames

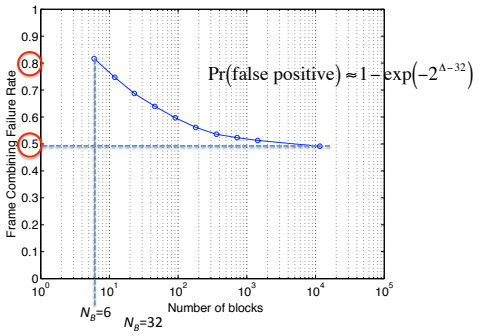


Breakdown of frame combining attempts

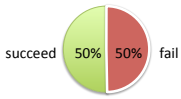
Frame recovery method
(out of recovered losses at R1)
soft selection via R2 85%
frame combining 15%

Frame recovery rate ≈ 49%

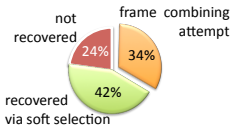
Would increasing N_B help?



Best case for frame combining



Breakdown of frame combining attempts



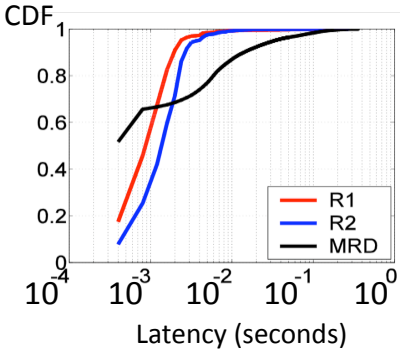
Breakdown of lost frames

Frame recovery method (out of recovered losses at R1)	
soft selection via R2	85%
frame combining	15%

Frame recovery rate ≈ 59%

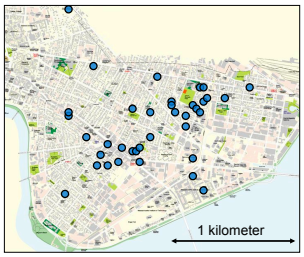
[slide: A. Miu, MobiCom '05]

Latency distribution



[Slides: adapted from Biswas, SIGCOMM '05]

ExOR: a new approach to routing in multi-hop wireless networks



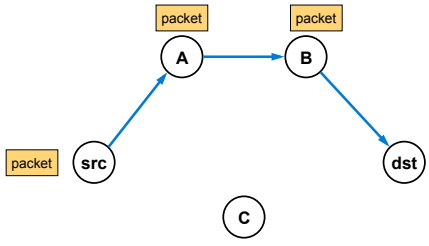
- Dense 802.11-based mesh
- Goal is high-throughput and capacity

The big wireless picture

Adaptation	**Opportunism**	**Diversity**
SampleRate	ExOR	MRD

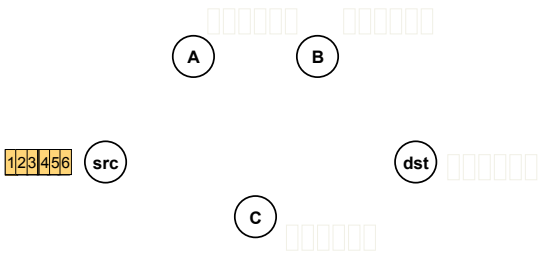
- Influence on later work
- Real implementation
- Evaluation

Initial approach: Traditional routing



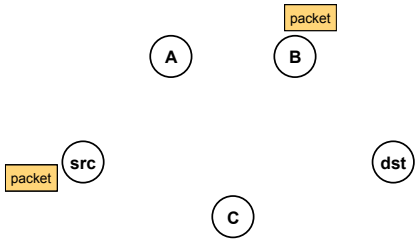
- Identify a route, forward over links
- Abstract radio to look like a wired link

Radios aren't wires



- Every packet is broadcast
- Reception is probabilistic

ExOR: exploiting probabilistic broadcast

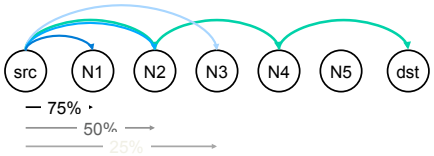


- Decide who forwards <u>after</u> reception
- Goal: only closest receiver should forward
- Challenge: agree efficiently and avoid duplicate xmits

Outline

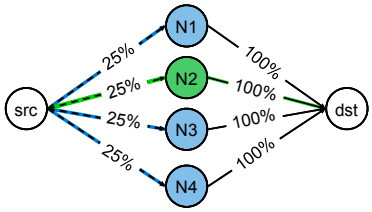
- Introduction
- Why ExOR might increase throughput
- ExOR protocol
- Measurements
- Related Work

Why ExOR might increase throughput (1)



- Best traditional route over 50% hops: $3(1/0.5) = 6$ tx
- Throughput $\cong 1/\#\text{ transmissions}$
- ExOR exploits lucky long receptions
- ExOR recovers unlucky short receptions

Why ExOR might increase throughput (2)

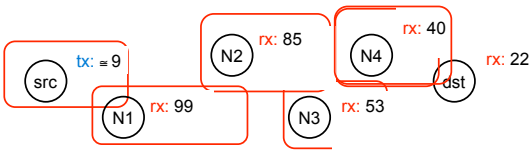


- Traditional routing: $1/0.25 + 1 = 5$ tx
- ExOR: $1/(1-(1-0.25)^4) + 1 = 2.5$ transmissions
- Assumes independent losses

Outline

- Introduction
- Why ExOR might increase throughput
- ExOR protocol
- Measurements
- Related Work

ExOR batching



- Challenge: finding the closest node to have rx'd
- Send batches of packets for efficiency
- Node closest to the dst sends first
  - Other nodes listen, send remaining packets in turn
- Repeat schedule until dst has whole batch