

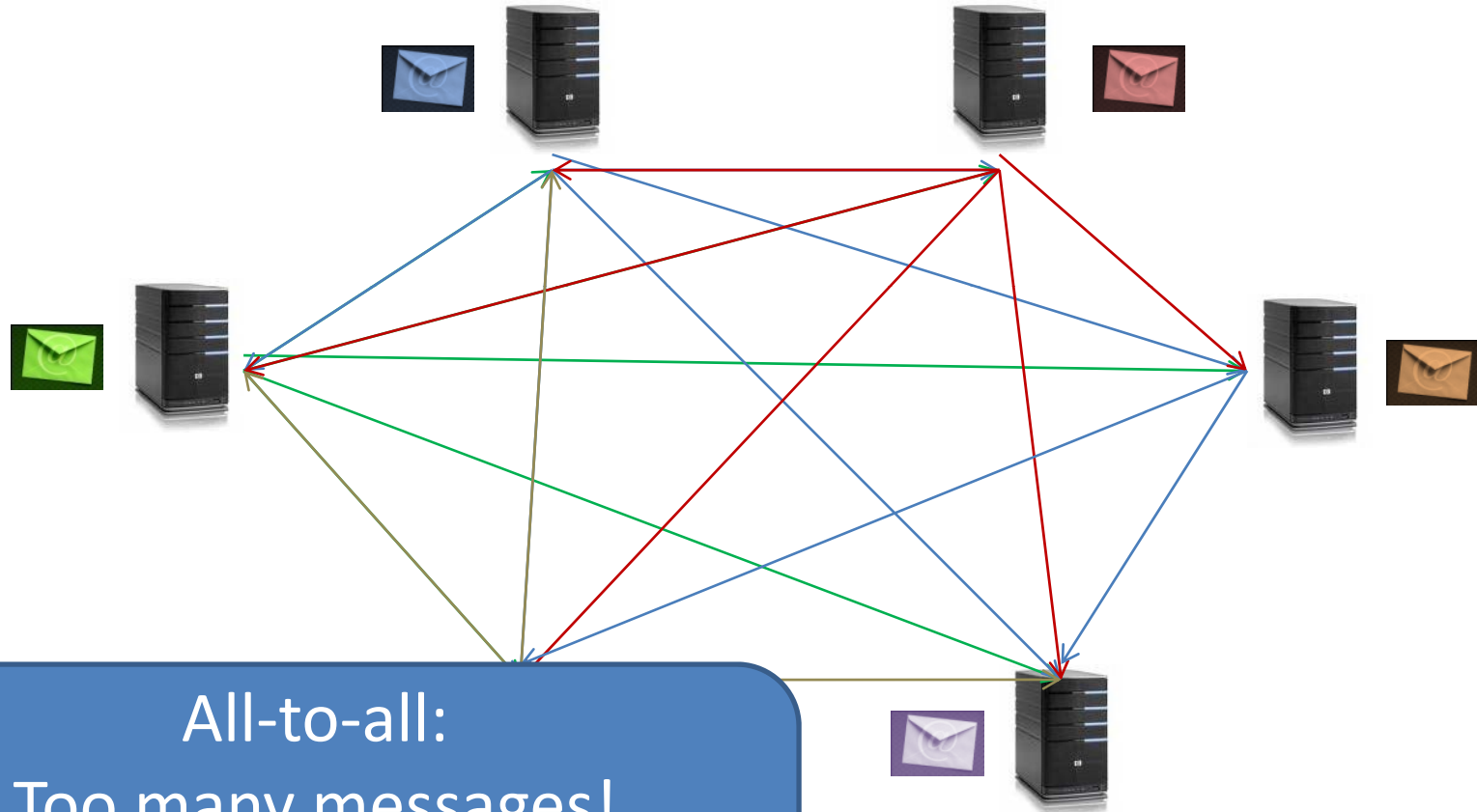
How Efficient is Robust Gossip?

Dan Alistarh (EPFL), Seth Gilbert (NUS),
Rachid Guerraoui (EPFL),
Morteza Zadimoghaddam (MIT)

Paper appeared at ICALP 2010,
EPFL TechReport online

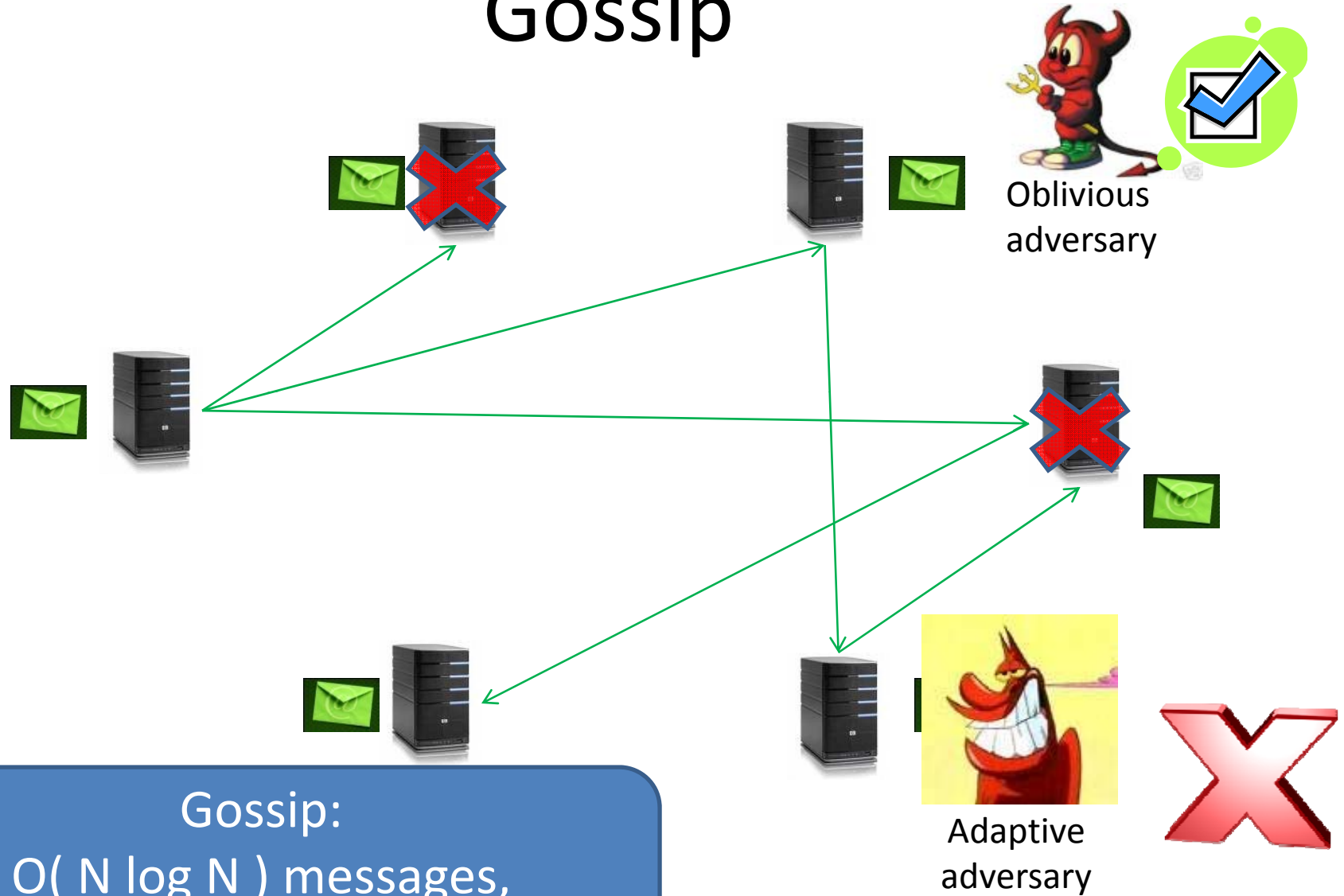


Information Exchange



All-to-all:
Too many messages!
Complexity $\Theta(N^2)$

Gossip



Gossip:
 $O(N \log N)$ messages,
 $O(\log N)$ rounds

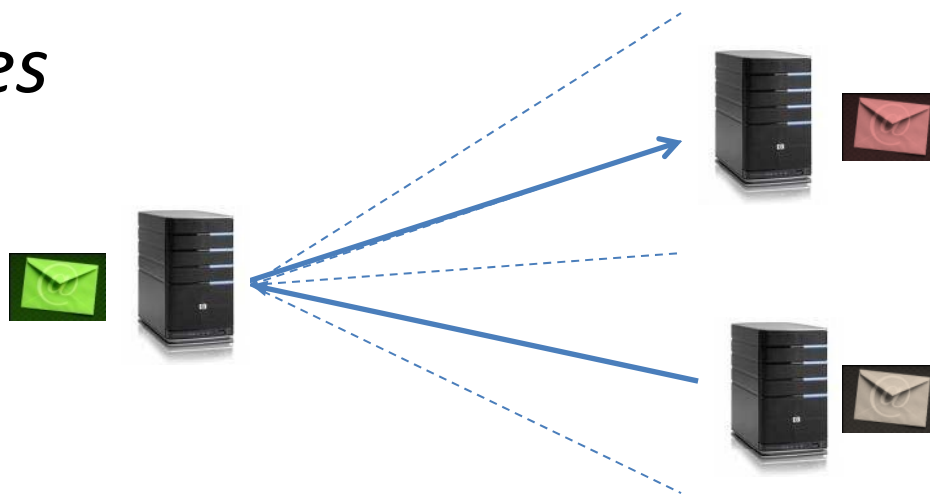
The question

Can we make gossip work against an
adaptive adversary?



The model

- N processes, t of which might fail
- Synchronous system, progressing in rounds
- *No consistent addresses*
 - $N - 1$ labeled ports



- The adversary
 - Controls process crashes
 - Crashes may depend on random choices



Adaptive
adversary⁶

Previous work

- [Karp et al.]
 - Spreading one rumor in $O(\log N)$ rounds using $O(N \log \log N)$ messages, using one “random phone-call” per round (oblivious adversary)
- [Chlebus, Kowalski et al.]
 - Deterministic Gossip against adaptive adversary with $O(\text{polylog } N)$ rounds and $O(N \text{ polylog } N)$ messages, assuming consistent identifiers
 - Won't work in this model...
- [many others]

Our gossip results

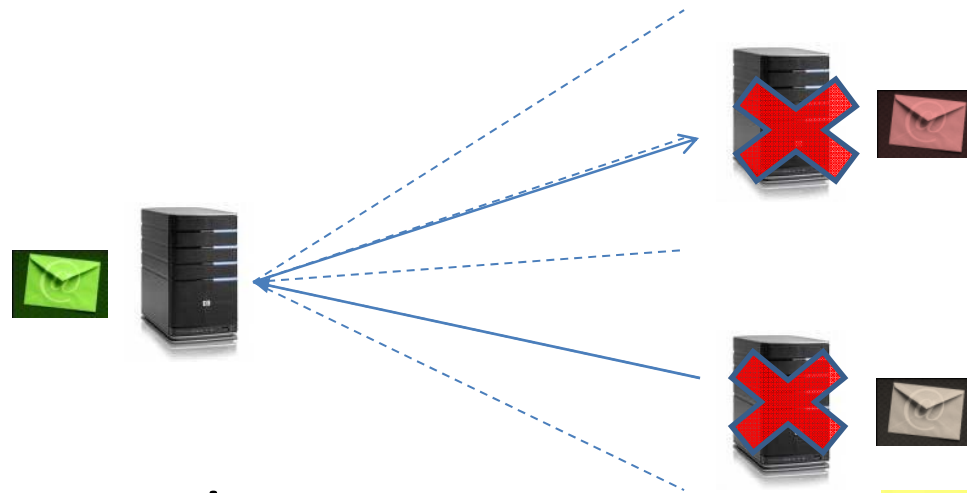
- The *TrickleGossip* protocol
 - dissemination with high probability
 - using $O(N \log^3 N)$ messages
 - $O(\log^2 N)$ rounds
- “Deterministic” *TrickleGossip*
 - Dissemination with probability 1
 - Expected $O(N \log^3 N)$ messages
- Lower bound of $\Omega(N \log N)$ messages for algorithms that tolerate $t > N - \sqrt{N}$ crash faults

The plan

- Introduction
- Results
- TrickleGossip
 - Structure
 - Some tricks
- Lower bound: the main idea
- Open questions

TrickleGossip: A first observation

- If the adversary is adaptive, then a process may have to send $N - 1$ messages in an execution



- ...so processes have to increase the number of messages they send



TrickleGossip: Structure

- We split the execution into dissemination phases
- A process *doubles* the number of messages sent in every phase, until it is convinced that its message is disseminated

procedure TrickleGossip:

do at phase i

1. *send* rumor to $\Theta(2^i \log N)$ random neighbors

2. for $\log N$ rounds, send rumors to $\log N$ neighbors

while (*rumor not disseminated*)

TrickleGossip: Polling

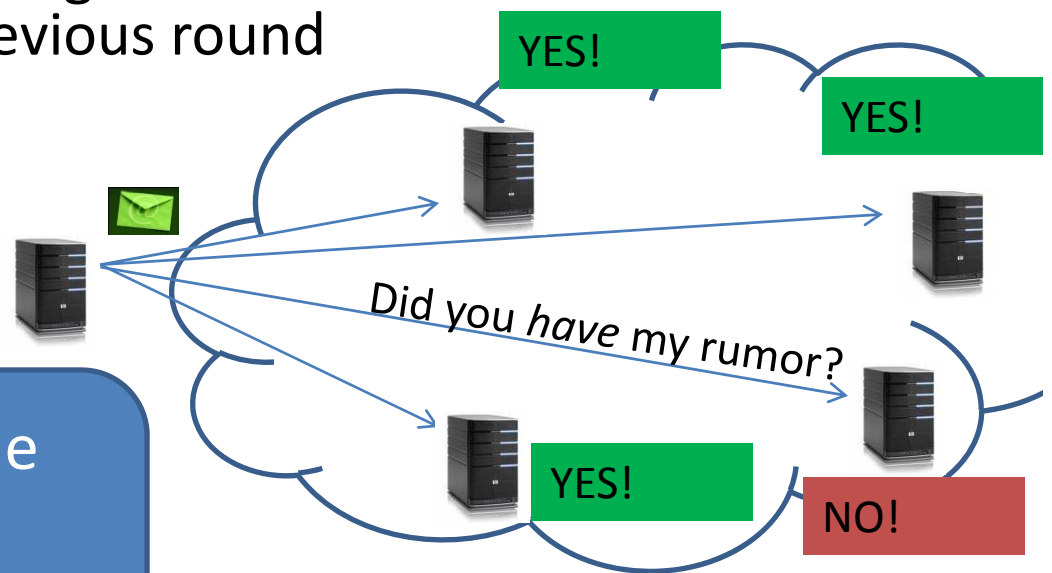
How can process p know that its rumor was disseminated (without asking everyone)?

- Process p randomly “polls” $\log N$ neighbors
- *Stop* if half of the polled neighbors *had* your rumor in the previous round

Why?

- At least $1/10$ of *all* participants had the rumor at the

We know how to determine when a rumor has been disseminated to a lot of processes.



whp (“coupon collector”)

TrickleGossip: Structure

Do rumors really get disseminated to lots of processes?

procedure Dissemination (phase i):

do

send rumor to $\Theta(2^i \log N)$ random neighbors
for $\log N$ rounds, send rumors to $\log N$ neighbors

while (*rumor not disseminated*)



TrickleGossip: Dissemination

Fix phase i

- *Fast rumors*: rumors that at least *double* their spread in each round in this phase
- **Step 1**: If F processes crash in a round, at most *constant* $\times F$ rumors are not *fast* in the round
- Why?
 - To *slow* R rumors, about $R \log N$ messages should go to processes that crash (*useless*)
 - A process gets $O(\log N)$ rumors in a round, whp
 - Hence at least R / K processes should crash

TrickleGossip: Dissemination

Fix phase i

- **Step 1:** If F processes crash in a round, at most *constant* $\times F$ rumors are not *fast* in the round, whp
- **Step 2*:** By the bound on t , at least $N / 2^i$ rumors are *fast* throughout phase i ...
- **Step 3:** at the end of phase i , at most N rumors have *not* been disseminated to (a) all, whp
- After $O(\log N)$ phases, we are done...

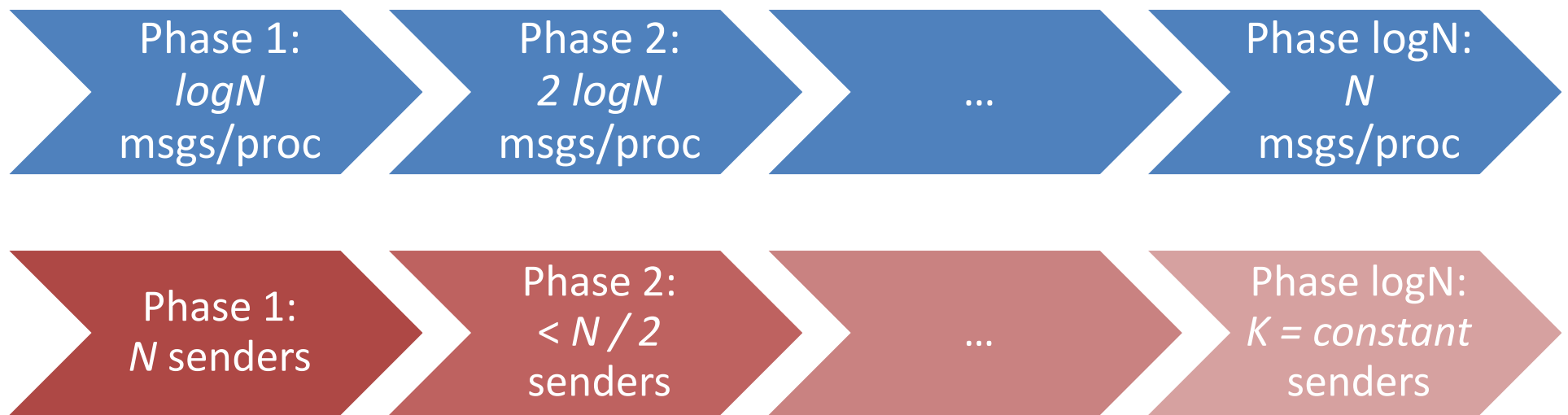
Lots of
details
left
out...



TrickleGossip so far

- Fact 1: A process can “see” when its rumor is well spread
- Fact 2: Many rumors get spread to many processes
- Round complexity $O(\log^2 N)$
- How about *message complexity*?

TrickleGossip: Message complexity



- At least half of the active senders (owners of *fast rumors*) stop by the end of each phase
- *Lemma*: There are $O(N \log^2 N)$ messages being sent in a phase, w.h.p.

The protocol

- *TrickleGossip*
 - dissemination with high probability
 - uses $O(N \log^3 N)$ messages
 - uses $O(\log^2 N)$ rounds
- “Deterministic” *TrickleGossip*
 - Dissemination with probability 1
 - Expected $O(N \log^3 N)$ messages, $t < N / 2$

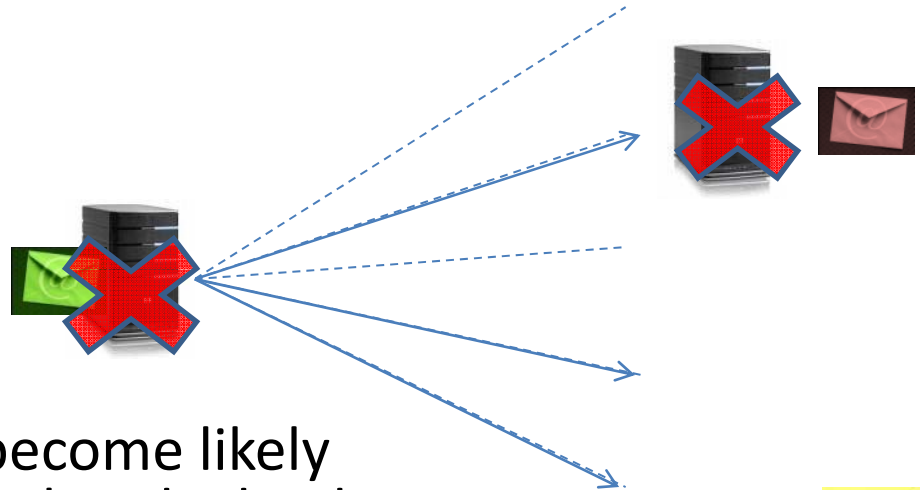
Lower bound

Can we do better than $N \log N$?

**NO
WE
CAN'T**

Lower bound: the strategy

- *Two rules*
 - Receivers are *killed* after they receive their messages for the round
 - Procs that send $> n / 4$ messages are *killed*
- *The claim:*
 - N / K messages sent kill roughly *half* of the remaining processes
 - *Insight:* later messages become likely to hit processes that are already dead...
 - $\frac{1}{2} \log N$ iterations generate $N \log N / 2K$ messages and kill all but \sqrt{N} processes, whp



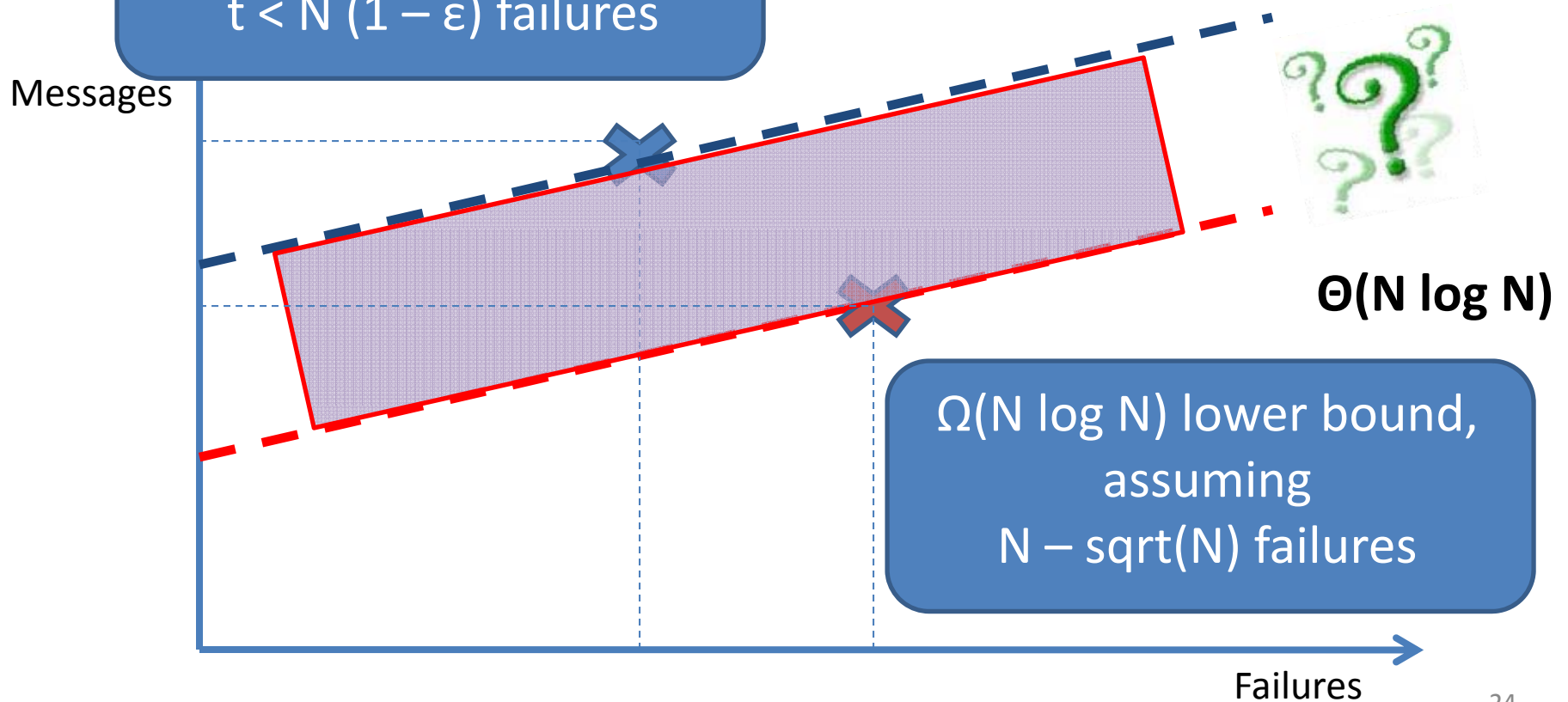
The Lower Bound

Theorem. In this model, any gossip algorithm that tolerates $t > N - \sqrt{N}$ crash failures needs to send $\Omega(N \log N)$ messages whp.

- Uses the “anonymity” of the ports
- Applies to broadcast or renaming as well

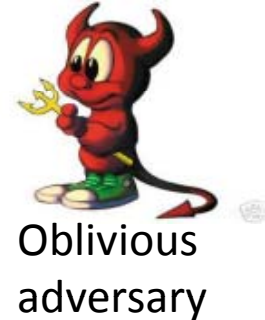
Mind the gap!

- $O(N \log^3 N)$ upper bound, assuming $t < N(1 - \epsilon)$ failures



Other results

- CoordinatedGossip:
 - Gossip with high probability against an oblivious adversary
 - $O(\text{polylog } N)$ rounds
 - $O(N)$ messages!
- Distributed renaming
 - *TrickleGossip* can be used as basis for a message-efficient renaming algorithm
 - Lower bound approach can be adapted to apply



Open questions



- Closing the gap between the algorithm and the lower bound
 - We believe that $\Theta(N \log N)$ is the tight threshold
- Relaxing the connectivity assumptions
- Communication complexity?

The Last Slide

Randomized information spreading **works** even against an **adaptive** adversary.

$\Omega(N \log N)$ seems to be the message complexity threshold.

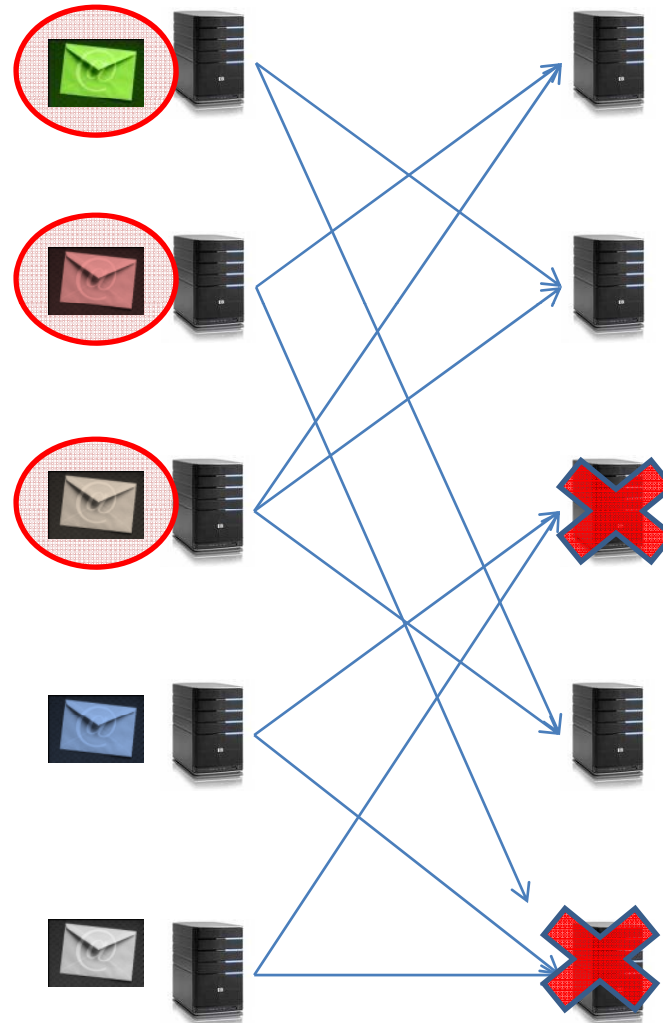
There are some promising applications to other problems.

Questions?

TrickleGossip: Dissemination

Phase one:

- Each process sends $3 \log N$ messages to random neighbors
- *Fast rumors*: rumors that are spread to at least $\log N$ processes that do not crash in this round
- Each process receives $3 \log N$ rumors on average
- By failing **one** process in this round, the adversary decreases the spread of $3 \log N$ rumors, in expectation
- *Claim*: by failing **F** process in this round, the adversary prevents at most **constant x F** rumors from being *fast*, w.h.p.



The gap!

- Closing the gap between the algorithm and the lower bound

