Optimistic, Signature-Free Reliable Broadcast and Its Applications

Nibesh Shrestha
Supra Research

Qianyu Yu
Hong Kong University of Science and Technology

Aniket Kate
Purdue University and Supra Research

Giuliano Losa
Stellar Development Foundation

Kartik Nayak
Duke University

Xuechao Wang
Hong Kong University of Science and Technology

SUPRA

香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

PURDUE UNIVERSITY

Stellar

Duke UNIVERSITY

We propose a new signature-free, asynchronous Byzantine Reliable Broadcast (RBC) algorithm that can improve the latency of many protocols and help achieve post-quantum security efficiently

Existing optimally-resilient algorithms

Tolerate 33% Byzantine failures

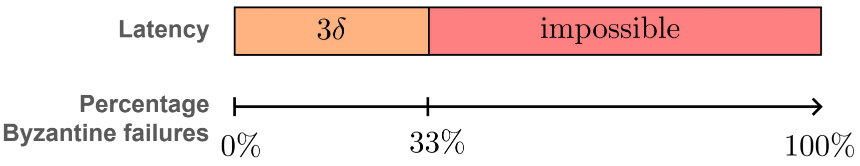
Latency of 3δ even without failures

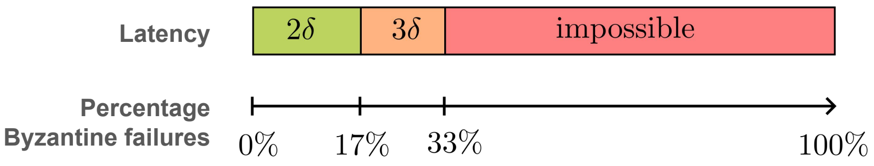
This work

Tolerates 33% Byzantine failures

Fast-path latency 2δ under 17% Byzantine failures (optimal)

Degrades to 3δ under 33% Byzantine failures





We propose a new signature-free, asynchronous Byzantine Reliable Broadcast (RBC) algorithm that can improve the latency of many protocols and help achieve post-quantum security efficiently

Many protocols rely on RBC, thus improving it can have a large impact

We apply our algorithm to reduce optimistic latency by one message delay and/or achieve post-quantum security efficiently in five distributed-computing schemes:

- Balanced RBC
- Asynchronous verifiable information dispersal (AVID)
- Asynchronous verifiable secret sharing (AVSS)
- Asynchronous complete secret sharing (ACSS)
- DAG-based BFT consensus with Sailfish++, a variant of Sailfish* that is post-quantum secure and achieves 3δ optimistic commit latency

* Shrestha et al. "Sailfish: Towards improving the latency of DAG-based BFT." S&P 2025

Algorithms with improved latency are important: users are sensitive to latency, and it often cannot be improved by just adding more resources

“In retail, we see that for every one second delay in page load time, conversions can fall by up to 20%”*

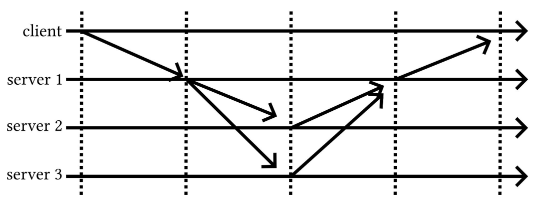
*https://blog.google/products/ads/speed-scorecard-impact-calculator/

Algorithms with improved latency are important: users are sensitive to latency, and it often cannot be improved by just adding more resources

“In retail, we see that for every one second delay in page load time, conversions can fall by up to 20%”*

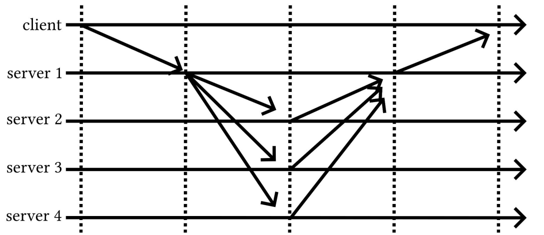
Often, one can buy throughput but not latency

latency is 4



+1 server

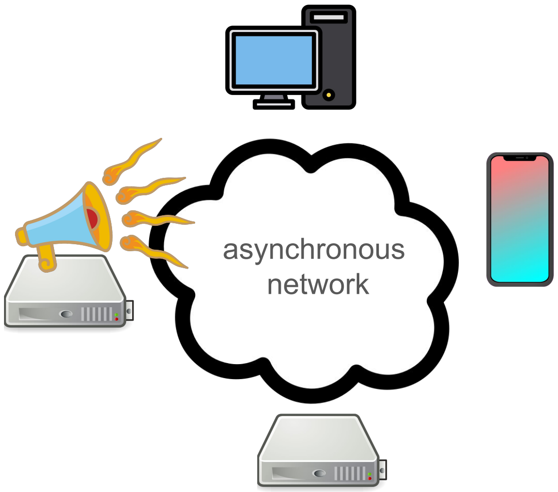
latency is still 4



*https://blog.google/products/ads/speed-scorecard-impact-calculator/

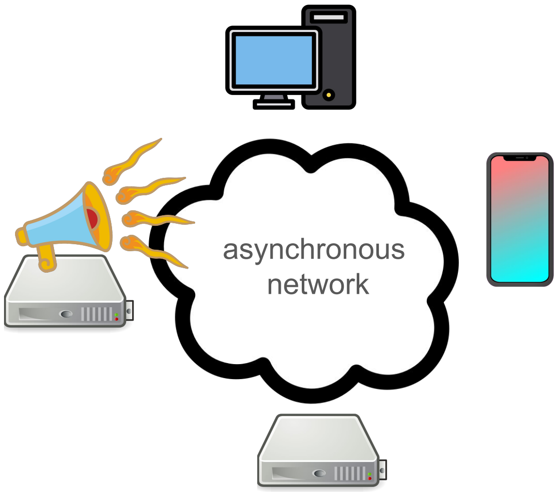
Byzantine Reliable Broadcasts (RBC) is a fundamental broadcast primitive ensuring all-or-nothing message delivery

- We have n parties among which f are Byzantine and the others are honest
- Communication is reliable but asynchronous
- We have a fixed broadcaster party that wants to broadcast a payload and we must ensure that:



Byzantine Reliable Broadcasts (RBC) is a fundamental broadcast primitive ensuring all-or-nothing message delivery

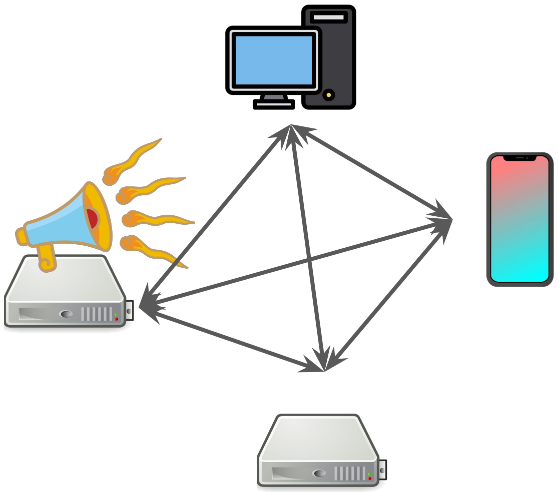
- We have n parties among which f are Byzantine and the others are honest
- Communication is reliable but asynchronous
- We have a fixed broadcaster party that wants to broadcast a payload and we must ensure that:
  - If the broadcaster is honest, then all honest parties eventually deliver the payload
  - Even if the broadcaster is Byzantine, either all honest parties eventually deliver the same payload or no honest party delivers any payload



In the signature-free setting, parties can only communicate "orally" and Byzantine parties can lie about what others said

We rely only on pairwise authenticated channels and not authenticated messages

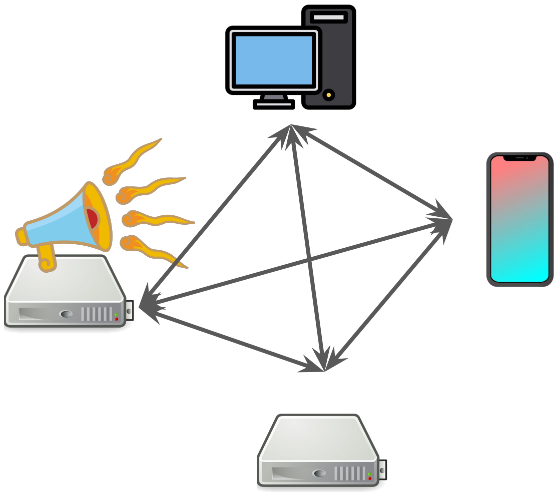
- Post-quantum secure
- Low computational overhead



In the signature-free setting, parties can only communicate "orally" and Byzantine parties can lie about what others said

We rely only on pairwise authenticated channels and not authenticated messages

- Post-quantum secure
- Low computational overhead



Bracha's famous algorithm implements RBC under the optimal $n > 3f$ and achieves 3δ latency

(Gabriel Bracha. *Asynchronous Byzantine agreement protocols*. 1987)

Other work that achieves a latency of 2δ has non-optimal resilience or uses signatures

	resilience	Good-case latency
Bracha †	n > 3f	3
Abraham et al. *	n ≥ 4f	2
Abraham et al. *	n ≥ 5f-1	2
Imbs and Raynal ‡	n > 5f	2
Folklore	n > 3f + signatures	2

† Gabriel Bracha. Asynchronous byzantine agreement protocols, 1987

* Abraham et al., Good-Case and Bad-Case Latency of Unauthenticated Byzantine Broadcast, 2021

‡ Imbs and Raynal. Trading off t-resilience for efficiency in asynchronous byzantine reliable broadcast, 2016

Previous work that achieves a latency of 2δ has non-optimal resilience or uses signatures

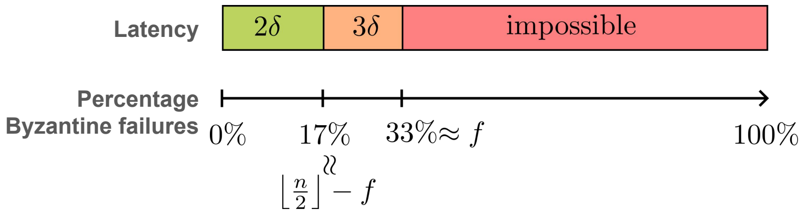
	resilience	Good-case latency
Bracha †	n > 3f	3
Abraham et al. *	n ≥ 4f	2
Abraham et al. *	n ≥ 5f-1	2
Imbs and Raynal ‡	n > 5f	2
Folklore	n > 3f + signatures	2
This work	n > 3f	(2,3)

RBC algorithm in this work

Optimal resilience to Byzantine failures: n > 3f or 33%

Fast-path with latency 2δ under at most ⌊n/2⌋-f Byzantine failures (≈17% asymptotically if n=3f+1)

3δ latency at most if the broadcaster is honest

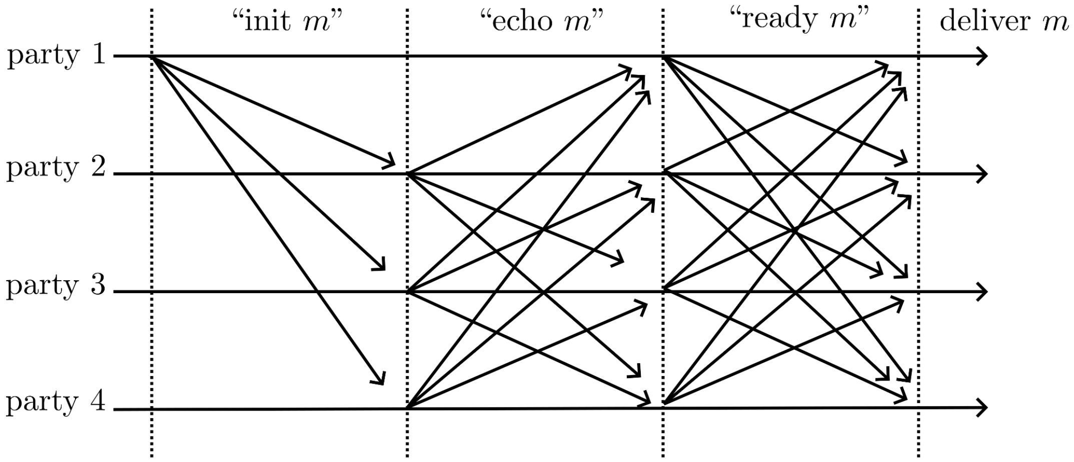


† Gabriel Bracha. Asynchronous byzantine agreement protocols, 1987

* Abraham et al., Good-Case and Bad-Case Latency of Unauthenticated Byzantine Broadcast, 2021

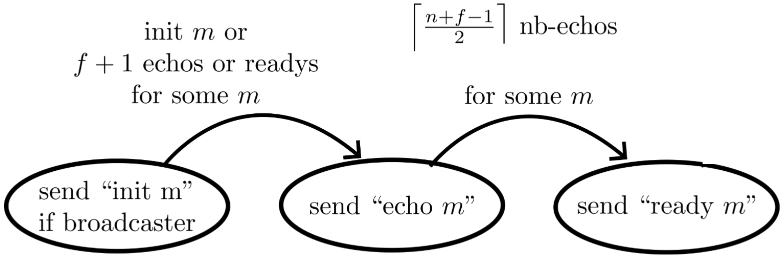
‡ Imbs and Raynal. Trading off t-resilience for efficiency in asynchronous byzantine reliable broadcast, 2016

Bracha’s algorithm uses 3 logical steps “init”, “echo”, and “ready”



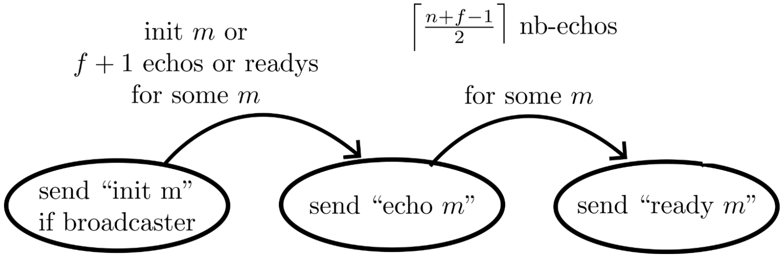
Bracha’s algorithm uses 3 logical steps “init”, “echo”, and “ready”

Each party takes the following steps:



Bracha's algorithm uses 3 logical steps "init", "echo", and "ready"

Each party takes the following steps:

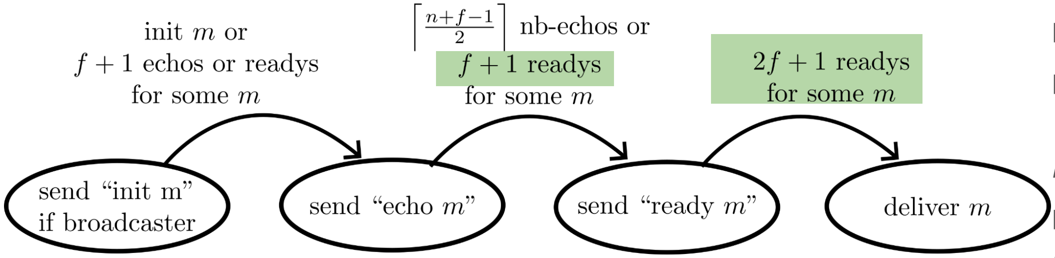


Key ideas to defend against a Byzantine broadcaster

No disagreement: any two sets of q=⌈(n+f-1)/2⌉ non-broadcaster parties have a common honest member because 2q - (f-1) > n-1

Bracha's algorithm uses 3 logical steps "init", "echo", and "ready"

Each party takes the following steps:



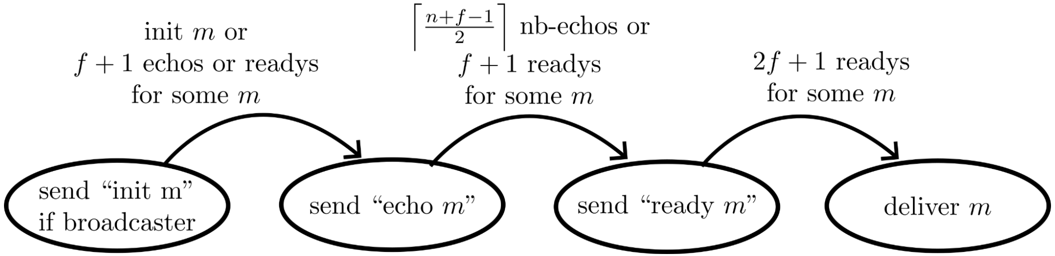
Key ideas to defend against a Byzantine broadcaster

No disagreement: any two sets of q=⌈(n+f-1)/2⌉ non-broadcaster parties have a common honest member because 2q - (f-1) > n-1

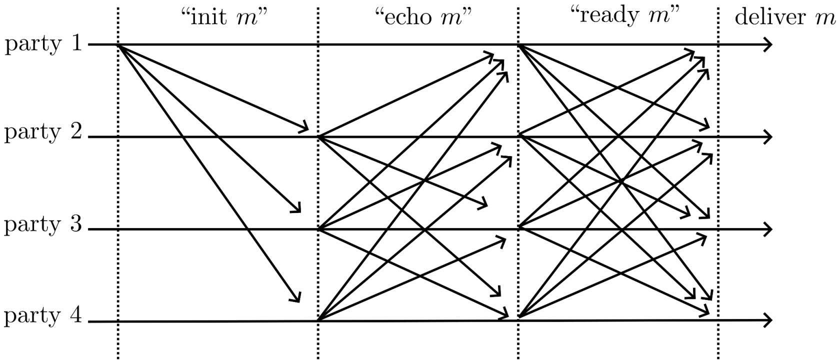
Eventual agreement: If a party observes 2f+1 readys for some m, then all observe f+1 readys for m and in turn send ready for m

Bracha's algorithm uses 3 logical steps "init", "echo", and "ready"

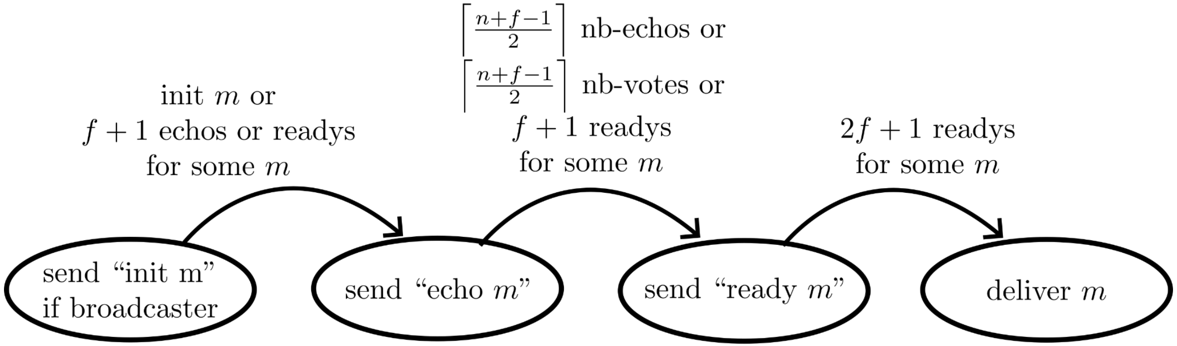
Each party takes the following steps:



A good-case execution looks like this:



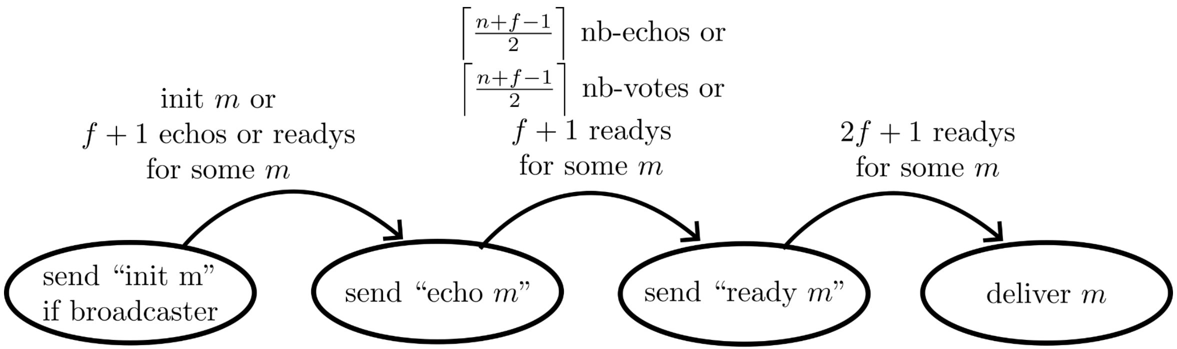
We allow 2-step delivery upon ⌈n/2⌉+f-1 ≈ 83% nb-echos, and we ensure eventual agreement by adding a new “vote” message



deliver m if $\left\lceil \frac{n}{2} \right\rceil + f - 1$ nb-echos for some m

send “vote m” if $\left\lceil \frac{n}{2} \right\rceil$ nb-echos for some m

We allow 2-step delivery upon ⌈n/2⌉+f-1 ≈ 83% nb-echos, and we ensure eventual agreement by adding a new “vote” message



deliver m if $\left\lceil \frac{n}{2} \right\rceil + f - 1$ nb-echos for some m

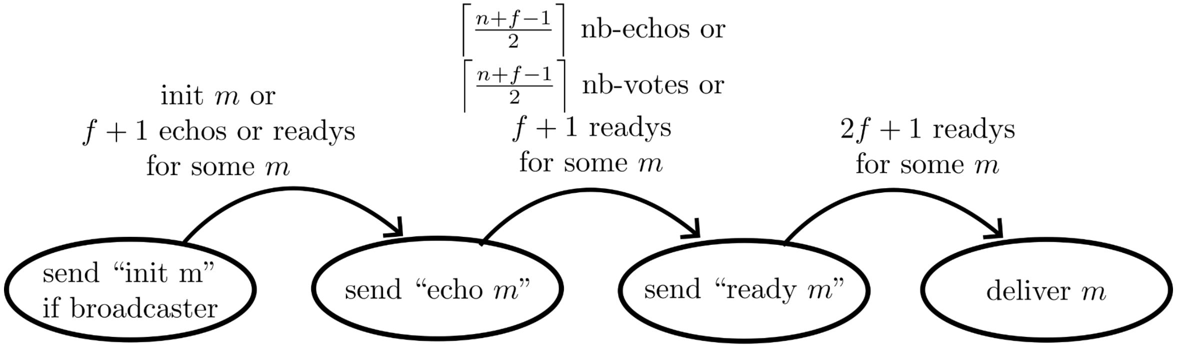
send “vote m” if $\left\lceil \frac{n}{2} \right\rceil$ nb-echos for some m

Key ideas to ensure eventual agreement under a Byzantine broadcaster

If a party observes fd=⌈n/2⌉+f-1 nb-echos for some m, no party can observe v=⌈n/2⌉ nb-echos for m’ ≠ m because fd+v-(f-1) > n-1

If a party observes ⌈n/2⌉+f-1 nb-echos for some m, then all observe > ⌈n/2⌉ nb-echos for m and vote for m, which leads to n-f readys and then delivery

We allow 2-step delivery upon ⌈n/2⌉+f-1 ≈ 83% nb-echos, and we ensure eventual agreement by adding a new “vote” message



deliver m if $\left\lceil \frac{n}{2} \right\rceil + f - 1$ nb-echos for some m

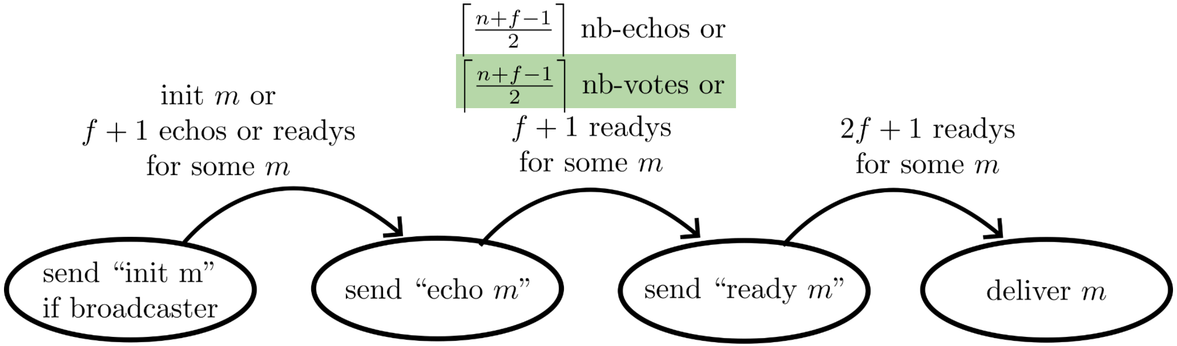
send “vote m” if $\left\lceil \frac{n}{2} \right\rceil$ nb-echos for some m

Key ideas to ensure eventual agreement under a Byzantine broadcaster

If a party observes fd=⌈n/2⌉+f-1 nb-echos for some m, no party can observe v=⌈n/2⌉ nb-echos for m’ ≠ m because fd+v-(f-1) > n-1

If a party observes ⌈n/2⌉+f-1 nb-echos for some m, then all observe > ⌈n/2⌉ nb-echos for m and vote for m, which leads to n-f readys and then delivery

We allow 2-step delivery upon ⌈n/2⌉+f-1 ≈ 83% nb-echos, and we ensure eventual agreement by adding a new “vote” message



deliver m if $\left\lceil \frac{n}{2} \right\rceil + f - 1$ nb-echos for some m

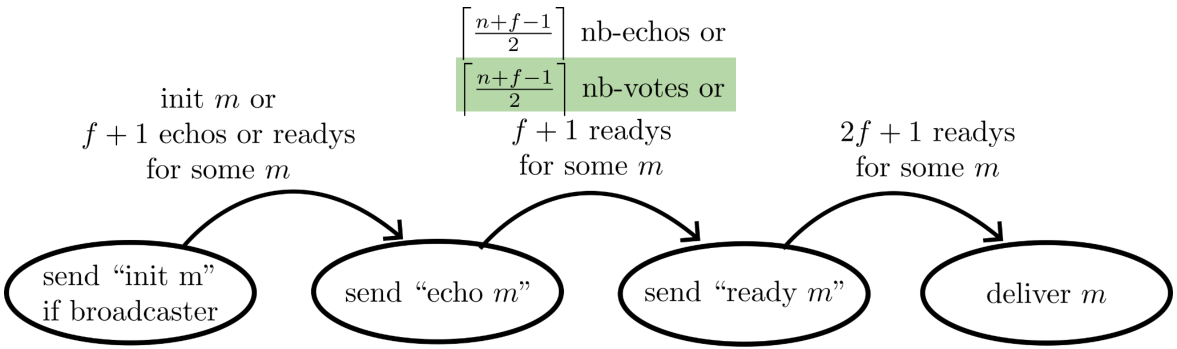
send “vote m” if $\left\lceil \frac{n}{2} \right\rceil$ nb-echos for some m

Key ideas to ensure eventual agreement under a Byzantine broadcaster

If a party observes fd=⌈n/2⌉+f-1 nb-echos for some m, no party can observe v=⌈n/2⌉ nb-echos for m’ ≠ m because fd+v-(f-1) > n-1

If a party observes ⌈n/2⌉+f-1 nb-echos for some m, then all observe > ⌈n/2⌉ nb-echos for m and vote for m, which leads to n-f readys and then delivery

We allow 2-step delivery upon ⌈n/2⌉+f-1 ≈ 83% nb-echos, and we ensure eventual agreement by adding a new “vote” message



deliver m if $\left\lceil \frac{n}{2} \right\rceil + f - 1$ nb-echos for some m

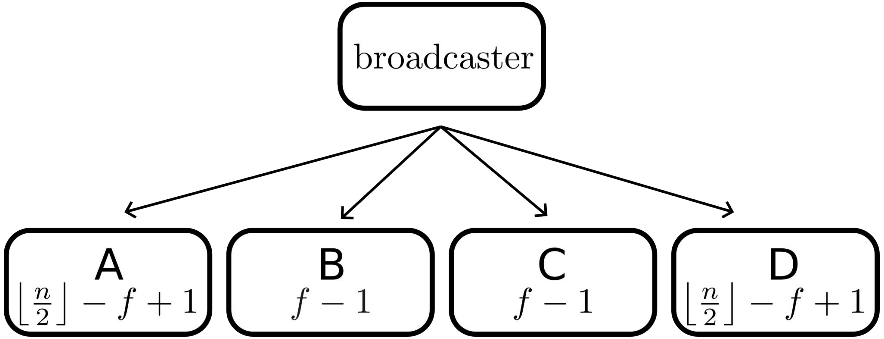
send “vote m” if $\left\lceil \frac{n}{2} \right\rceil$ nb-echos for some m

Key ideas to ensure eventual agreement under a Byzantine broadcaster

If a party observes fd=⌈n/2⌉+f-1 nb-echos for some m, no party can observe v=⌈n/2⌉ nb-echos for m’ ≠ m because fd+v-(f-1) > n-1

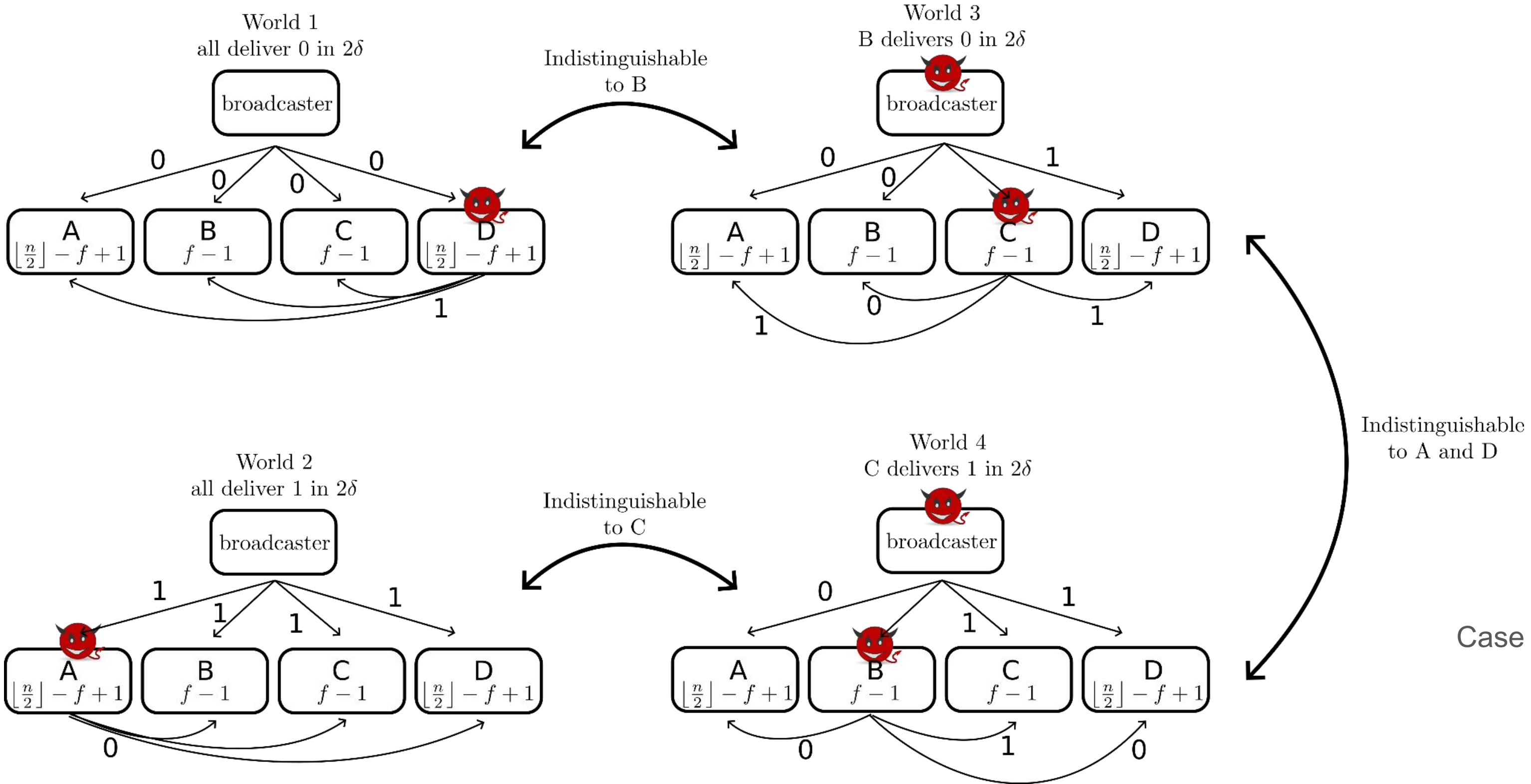
If a party observes ⌈n/2⌉+f-1 nb-echos for some m, then all observe > ⌈n/2⌉ nb-echos for m and vote for m, which leads to n-f readys and then delivery

We show optimality by contradiction: suppose we can deliver in 2δ despite ⌊n/2⌋-f+1 Byzantine failures…



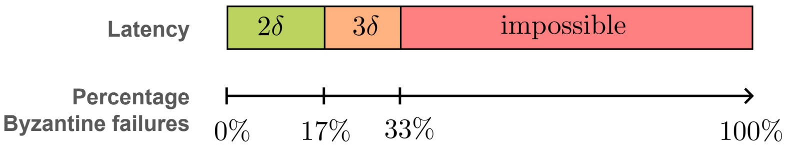
Case of odd n

We show optimality by contradiction: suppose we can deliver in 2δ despite ⌊n/2⌋-f+1 Byzantine failures…



Case of odd n

We present a new *signature-free* asynchronous Byzantine Reliable Broadcast algorithm with optimal resilience, optimal optimistic latency of 2δ even under 17% Byzantine failures, and latency of 3δ up to 33% Byzantine failures



Our algorithm can improve the latency of many distributed-computing schemes and enable post-quantum security at low latency. Examples in the paper include:

- Balanced RBC
- Asynchronous verifiable information dispersal (AVID)
- Asynchronous verifiable secret sharing (AVSS)
- Asynchronous complete secret sharing (ACSS)
- Post-quantum secure DAG-based consensus with Sailfish++

Can you apply our optimistic signature-free RBC to improve your protocols?