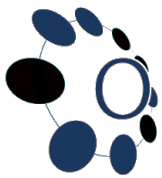
A Modular Approach to Construct Signature-Free BRB Algorithms under a Message Adversary

Timothé Albouy, Davide Frey, Michel Raynal, François Taïani

Univ Rennes, IRISA, CNRS, Inria, 35042 Rennes, France

15th December 2022



UNIVERSITÉ DE RENNES 1

Inria

UMR IRISA

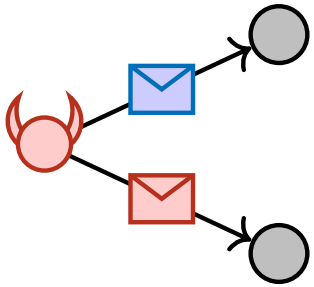
There is a World beyond Byzantium!

There is a World beyond Byzantium!

Byzantine Fault Tolerance (BFT) is not the end of the road

Process faults & Network faults

Byzantine faults

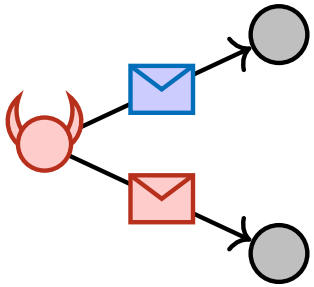


Model for **Process** faults:

- attackers
- crashes
- implementation errors
- ...

Process faults & Network faults

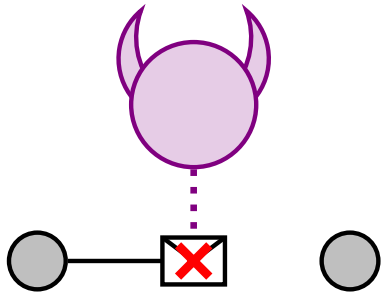
Byzantine faults



Model for **Process** faults:

- attackers
- crashes
- implementation errors
- ...

Message adversary (MA)

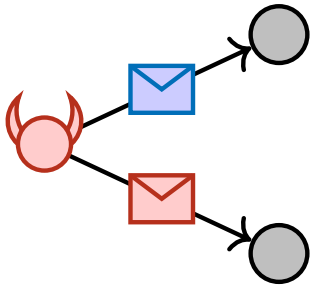


Model for **Network** faults:

- message losses
- disconnections
- interference
- ...

Process faults & Network faults

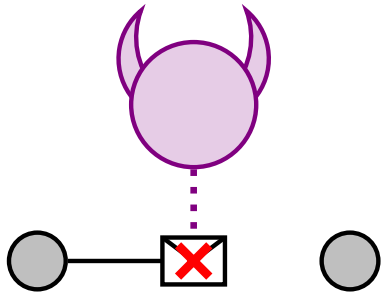
Byzantine faults



Model for **Process** faults:

- attackers
- crashes
- implementation errors
- ...

Message adversary (MA)



Model for **Network** faults:

- message losses
- disconnections
- interference
- ...

- **2 orthogonal fault models**
- What happens if we combine the 2?
- Case of **Reliable Broadcast**

Roadmap

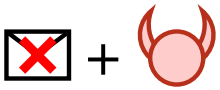
1. New fault model:  + 

Roadmap

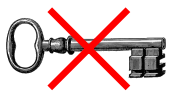
1. New fault model:  + 

2. New reliable broadcast specification: **MBRB**

Roadmap

1. New fault model:  +
2. New reliable broadcast specification: **MBRB**
3. New many-to-many primitive: ***k*2*ℓ*-cast**

Roadmap

1. New fault model:  +
2. New reliable broadcast specification: **MBRB**
3. New many-to-many primitive: ***k*2ℓ-cast**
4. 2 MBRB implementations 

System Model

- **Asynchronous** network (arbitrary message delays)

System Model

- **Asynchronous** network (arbitrary message delays)
- n processes in the network: $p_1, p_2, \ldots, p_n$

System Model

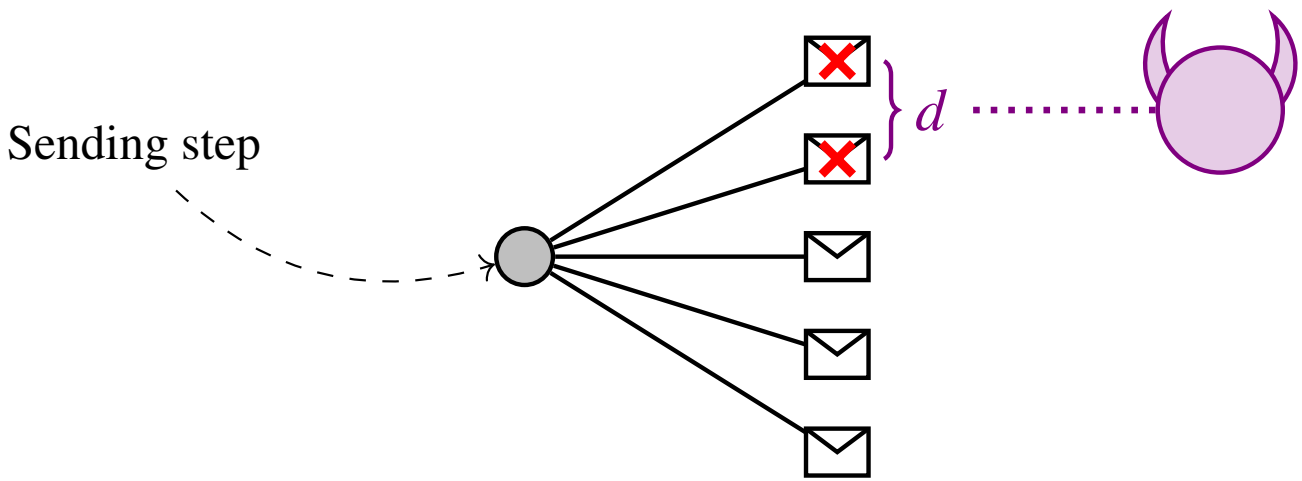
- **Asynchronous** network (arbitrary message delays)
- n processes in the network: $p_1, p_2, \ldots, p_n$
- $t < n$ processes can be Byzantine 

System Model

- **Asynchronous** network (arbitrary message delays)
- n processes in the network: $p_1, p_2, \ldots, p_n$
- $t < n$ processes can be Byzantine 
- $c \in [n-t..n]$ correct processes

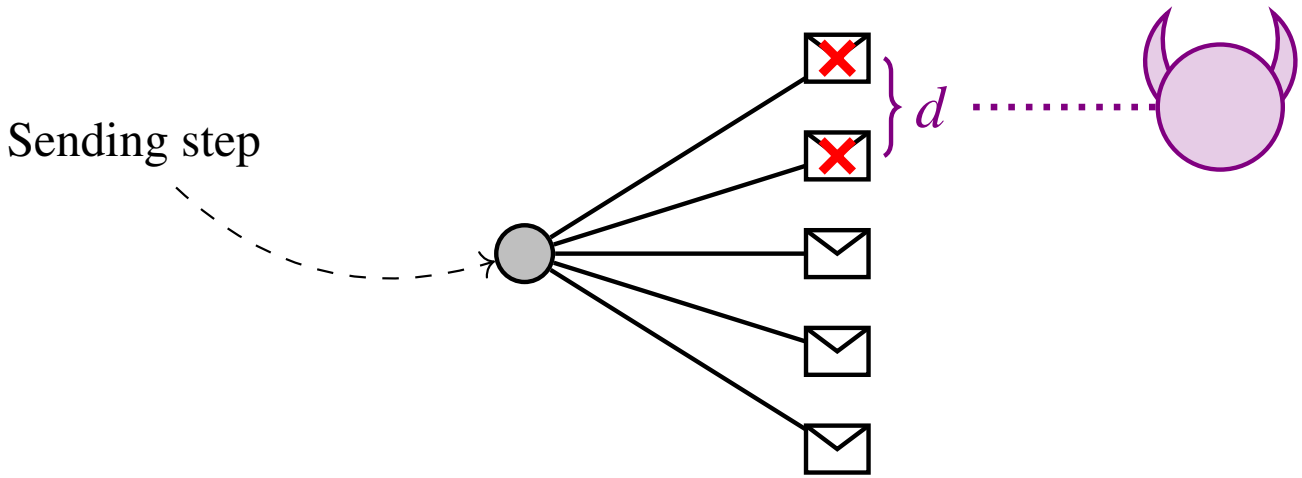
System Model

- **Asynchronous** network (arbitrary message delays)
- n processes in the network: $p_1, p_2, \ldots, p_n$
- $t < n$ processes can be Byzantine
- $c \in [n-t..n]$ correct processes
- $d < n$ messages can be deleted each sending step **(MA power)**



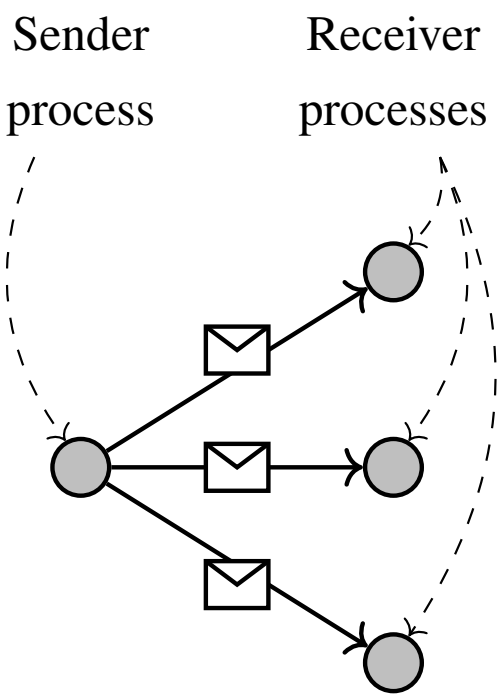
System Model

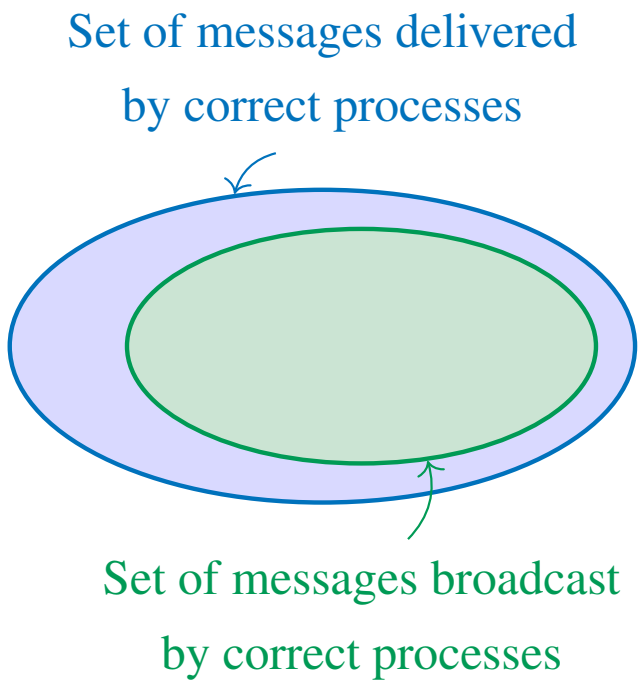
- **Asynchronous** network (arbitrary message delays)
- n processes in the network: $p_1, p_2, \ldots, p_n$
- $t < n$ processes can be Byzantine
- $c \in [n-t..n]$ correct processes
- $d < n$ messages can be deleted each sending step **(MA power)**



- **Deterministic** algorithm

Reminder: Byzantine Reliable Broadcast (BRB)





Many practical application: consensus algorithms, voting systems, state-machine replication, payment systems, distributed memory...

Classical BRB

BRB SAFETY:

- **Validity:** If a correct process p_i delivers m from a correct process p_j, then p_j broadcast m
- **No-duplication:** A correct process p_i delivers at most one app-message from a process p_j
- **No-duplicity:** No two correct processes deliver different app-messages from a process p_i

BRB LIVENESS:

- **Local-delivery:** If a correct process p_i broadcasts m, then at least one correct process p_j delivers m from p_i
- **Global-delivery:** If a correct process p_i delivers m, then all c correct processes deliver m

MA-Tolerant BRB (MBRB)

MBRB SAFETY:

- **Validity:** If a correct process p_i delivers m from a correct process p_j, then p_j broadcast m
- **No-duplication:** A correct process p_i delivers at most one app-message from a process p_j
- **No-duplicity:** No two correct processes deliver different app-messages from a process p_i

MBRB LIVENESS:

- **Local-delivery:** If a correct process p_i broadcasts m, then at least one correct process p_j delivers m from p_i
- **Global-delivery:** If a correct process p_i delivers m, then at least $\ell_{MBRB} \leq c - d$ correct processes deliver m

As the MA can totally isolate d processes

Signature-based MBRB

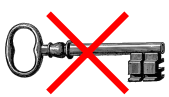
Albouy, Frey, Raynal and Taïani (SSS 2021):
Byzantine-tolerant reliable broadcast in the presence of silent churn

- First (signature-based) MBRB implementation 
  - assumes $n > 3t + 2d$ (optimal)
  - guarantees $\ell_{MBRB} = c - d$ (optimal)
- Theorem: $n > 3t + 2d$ **tight** for MBRB
  - **optimal** implementation

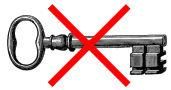
Signature-based MBRB 

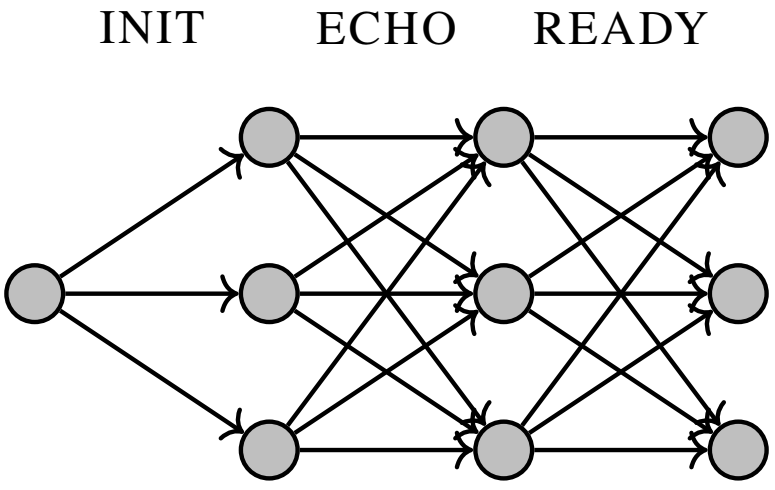
Albouy, Frey, Raynal and Taïani (SSS 2021):
Byzantine-tolerant reliable broadcast in the presence of silent churn

- First (signature-based) MBRB implementation 
  - assumes $n > 3t + 2d$ (optimal)
  - guarantees $\ell_{MBRB} = c - d$ (optimal)
- Theorem: $n > 3t + 2d$ **tight** for MBRB
  - **optimal** implementation

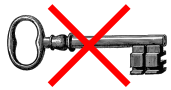
Question: Is signature-free MBRB possible? 

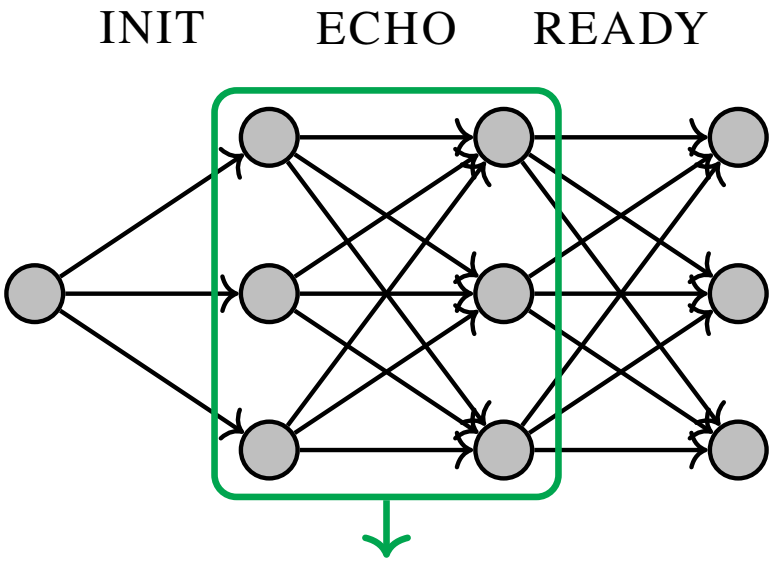
Intuition

How BRB algorithms work (e.g. Bracha 1987) 

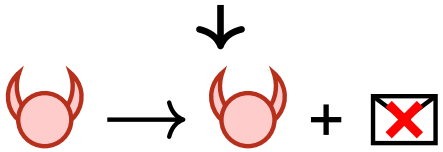


Intuition

How BRB algorithms work (e.g. Bracha 1987) 



Extract quorum construction pattern: ***k*2ℓ-cast**



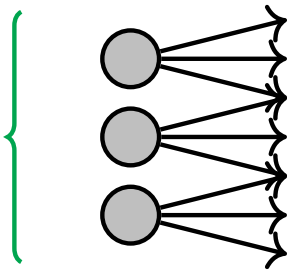
Quorum Construction Pattern

Let $q_f, q_d \in \mathbb{N}^*$, with $1 \leq q_f \leq q_d \leq n$

Quorum Construction Pattern

Let $q_f, q_d \in \mathbb{N}^*$, with $1 \leq q_f \leq q_d \leq n$

1. initiator processes

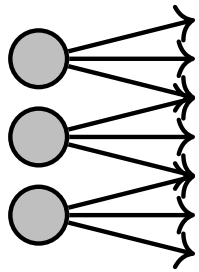


broadcast of $\text{MSG}(m)$

Quorum Construction Pattern

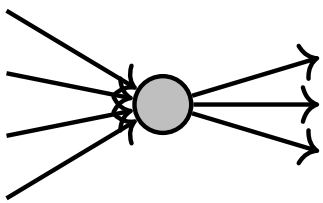
Let $q_f, q_d \in \mathbb{N}^*$, with $1 \leq q_f \leq q_d \leq n$

1. initiator processes



broadcast of $\text{MSG}(m)$

2. q_f messages $\text{MSG}(m)$

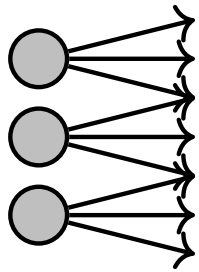


forwarding of $\text{MSG}(m)$

Quorum Construction Pattern

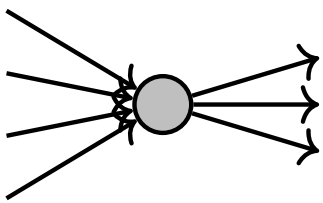
Let $q_f, q_d \in \mathbb{N}^*$, with $1 \leq q_f \leq q_d \leq n$

1. initiator processes



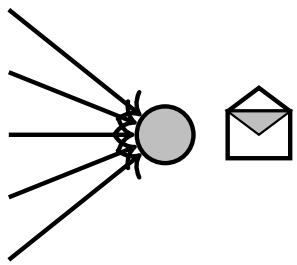
broadcast of $\text{MSG}(m)$

2. q_f messages $\text{MSG}(m)$



forwarding of $\text{MSG}(m)$

3. q_d messages $\text{MSG}(m)$

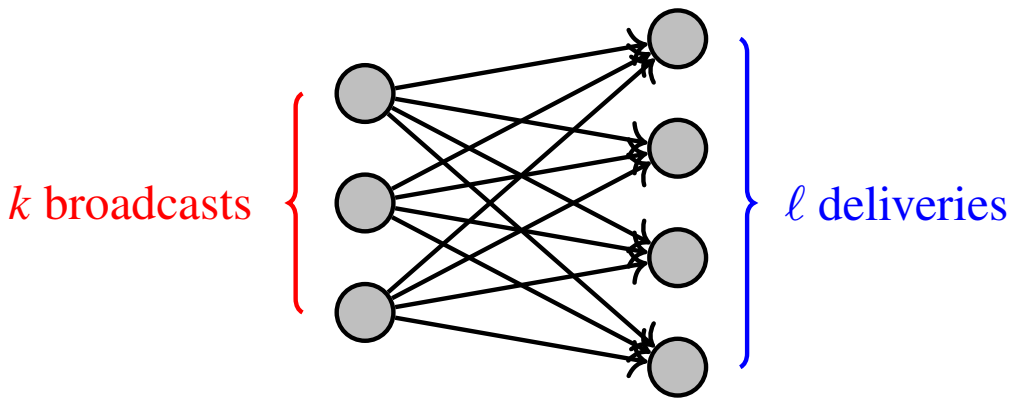


delivery of m

$k2\ell$-Cast Abstraction

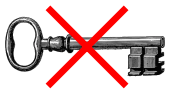
$k2\ell$-cast (for k-to-ℓ-cast): many-to-many protocol

(k correct processes *$k2\ell$-cast*) $\rightarrow$ (ℓ correct processes *$k2\ell$-deliver*)



Guarantee	Meaning	Best value
k	(**k** $k2\ell$-casts) $\rightarrow$ (**one** $k2\ell$-delivery)	Low
ℓ	(**one** $k2\ell$-delivery) $\rightarrow$ (**ℓ** $k2\ell$-deliveries)	High

$k2\ell$-Cast Implementation

Encapsulates *Quorum Construction Pattern* 

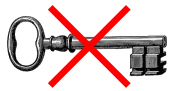
Parameter	Meaning
q_f	Forwarding threshold
q_d	Delivery threshold

Theorem: This implementation guarantees 

- $k = \left\lfloor \frac{c(q_f-1)}{c-d-q_d+q_f} \right\rfloor + 1$
- $\ell = c - \left\lfloor \frac{cd}{c-q_d+1} \right\rfloor$

$k2\ell$-Cast in Action: Signature-Free MBRB

$k2\ell$-cast allows

- ☒-tolerant & simpler BRB 
- systematic approach to analyze quorums
  - **side-effect:** optimized algorithms

Examples:

- **Bracha**'s BRB (1987)
- **Imbs-Raynal**'s BRB (2016)

Reconstructing Bracha's BRB (1/2)

```
operation brb_broadcast(m, sn) is
(1)  broadcast(INIT(m, sn)).

when INIT(m, sn) is received from p_j do
(2)  if (ECHO(−, sn, j) not yet broadcast)
(3)     then broadcast(ECHO(m, sn, j))
(4)  end if.

when ECHO(m, sn, j) is received do
(5)  if (ECHO(m, sn, j) is received from strictly more than (n+t)/2 processes) then
(6)     if (ECHO(−, sn, j) not yet broadcast)
(7)        then broadcast(ECHO(m, sn, j))
(8)     end if;
(9)     if (READY(−, sn, j) not yet broadcast)
(10)       then broadcast(READY(m, sn, j))
(11)    end if
(12) end if.

when READY(m, sn, j) is received do
(13) if (READY(m, sn, j) is received from at least t + 1 processes
         ∧ READY(−, sn, j) not yet broadcast)
(14)    then broadcast(READY(m, sn, j))
(15) end if;
(16) if (READY(m, sn, j) is received from at least 2t + 1 processes
         ∧ (−, sn, j) not already brb-delivered)
(17)    then brb_deliver(m, sn, j)
(18) end if.
```

Original version

BRB

18 lines

k2ℓ-cast

```
init: obj_E ← K2LCast(q_d=⌊(n+t)/2⌋+1, q_f=t+1);
      obj_R ← K2LCast(q_d=2t+d+1, q_f=t+1).
(1) operation mbrb_broadcast(m, sn) is broadcast(INIT(m, sn)).
(2) when INIT(m, sn) is received from p_j do obj_E.k2ℓ_cast(ECHO(m), (sn, j)).
(3) when (ECHO(m), (sn, j)) is obj_E.k2ℓ_delivered do obj_R.k2ℓ_cast(READY(m), (sn, i)).
(4) when (READY(m), (sn, j)) is obj_R.k2ℓ_delivered do mbrb_deliver(m, sn, j).
```

Reconstructed version

MBRB

4 lines

Reconstructing Bracha's BRB (1/2)

operation brb_broadcast(m, sn) **is**
(1) broadcast(INIT(m, sn)).

when INIT(m, sn) **is received from** p_j **do**
(2) **if** (ECHO$(-, sn, j)$ not yet broadcast)
(3) **then** broadcast(ECHO(m, sn, j))
(4) **end if**.

when ECHO(m, sn, j) **is received do**
(5) **if** (ECHO(m, sn, j) is received from strictly more than $\frac{n+t}{2}$ processes) **then**
(6) **if** (ECHO$(-, sn, j)$ not yet broadcast)
(7) **then** broadcast(ECHO(m, sn, j))
(8) **end if**;
(9) **if** (READY$(-, sn, j)$ not yet broadcast)
(10) **then** broadcast(READY(m, sn, j))
(11) **end if**
(12) **end if**.

when READY(m, sn, j) **is received do**
(13) **if** (READY(m, sn, j) is received from at least $t + 1$ processes
∧ READY$(-, sn, j)$ not yet broadcast)
(17) **then** brb_deliver(m, sn, j)
(18) **end if**.

Takeaway: Better & simpler algorithms

$k2\ell$**-cast**

init: $obj_{\rm E} \leftarrow$ K2LCast$(q_d{=}\lfloor\frac{n+t}{2}\rfloor{+}1, q_f{=}t{+}1)$;
$obj_{\rm R} \leftarrow$ K2LCast$(q_d{=}2t{+}d{+}1, q_f{=}t{+}1)$.

(1) **operation** mbrb_broadcast(m, sn) **is** broadcast(INIT(m, sn)).
(2) **when** INIT(m, sn) **is** received **from** p_j **do** $obj_{\rm E}.k2\ell$_cast(ECHO(m), (sn, j)).
(3) **when** (ECHO(m), (sn, j)) **is** $obj_{\rm E}.k2\ell$_delivered **do** $obj_{\rm R}.k2\ell$_cast(READY(m), (sn, i)).
(4) **when** (READY(m), (sn, j)) **is** $obj_{\rm R}.k2\ell$_delivered **do** mbrb_deliver(m, sn, j).

Original version
BRB
18 lines

Reconstructed version
MBRB
4 lines

Reconstructing Bracha's BRB (2/2)

When $d = 0$ (no MA), 2 versions have same

- t-resilience: $n > 3t$
- round complexity: ≤ 3 rounds
- message complexity: $\leq n + 2n^2$ messages

Reconstructing Bracha's BRB (2/2)

When $d = 0$ (no MA), 2 versions have same

- t-resilience: $n > 3t$
- round complexity: ≤ 3 rounds
- message complexity: $\leq n + 2n^2$ messages

However, new version uses smaller quorums (ECHO phase):

Threshold	Original version	New version	
Forwarding q_f	$\left\lfloor \frac{n+t}{2} \right\rfloor + 1$	$t + 1$	BETTER
Delivery q_d	$\left\lfloor \frac{n+t}{2} \right\rfloor + 1$	$\left\lfloor \frac{n+t}{2} \right\rfloor + 1$	

Needs less messages to progress $\rightarrow$ **terminates earlier**

Reconstructing Imbs-Raynal's BRB

When $d = 0$ (no MA), 2 versions have same

- t-resilience: $n > 5t$
- round complexity: ≤ 2 rounds
- message complexity: $\leq n + n^2$ messages

Reconstructing Imbs-Raynal's BRB

When $d = 0$ (no MA), 2 versions have same

- t-resilience: $n > 5t$
- round complexity: ≤ 2 rounds
- message complexity: $\leq n + n^2$ messages

However, new version uses smaller quorums:

Threshold	Original version	New version
Forwarding q_f	$n - 2t$	$\lfloor \frac{n+t}{2} \rfloor + 1$
Delivery q_d	$n - t$	$\lfloor \frac{n+3t}{2} \rfloor + 1$

BETTER

Needs less messages to progress $\rightarrow$ **terminates earlier**

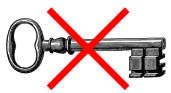
Conclusion

- New computing model: ☒ + 

Conclusion

- New computing model: ☒ + 
- New modular many-to-many primitive: ***k*2ℓ-cast**
  - Better quorum engineering

Conclusion

- New computing model: ☒ + 
- New modular many-to-many primitive: ***k*2ℓ-cast**
  - Better quorum engineering
- Applications to quorum-based algorithms:
  - BRB $\rightarrow$ ☒-tolerant BRB 
  - Self-stabilizing systems?

Conclusion

- New computing model:  +
- New modular many-to-many primitive: ***k*2ℓ-cast**
  - Better quorum engineering
- Applications to quorum-based algorithms:
  - BRB →  -tolerant BRB
  - Self-stabilizing systems?

Thank you for your attention!

Reconstructing Imbs-Raynal's BRB (Cont'd)

Original version (BRB)

operation brb_broadcast(m, sn) **is**
(1) broadcast(INIT(m, sn)).

when INIT(m, sn) **is** received **from** p_j **do**
(2) **if** (WITNESS$(-, sn, j)$ not yet broadcast)
(3) **then** broadcast WITNESS(m, sn, j)
(4) **end if**.

when WITNESS(m, sn, j) **is** received **do**
(5) **if** (WITNESS(m, sn, j) is received from at least $n - 2t$ processes
$\wedge$ WITNESS(m, sn, i) not yet broadcast)
(6) **then** broadcast(WITNESS(m, sn, i))
(7) **end if**;
(8) **if** (WITNESS(m, sn, j) **is** received from at least $n - t$ processes
$\wedge$ $(-, sn, j)$ not already brb-delivered)
(9) **then** brb_deliver(m, sn, j)
(10) **end if**.

$\downarrow$ $k2\ell$-cast $\downarrow$

Reconstructed version (MBRB)

init: $obj_{\rm W} \leftarrow$ K2LCast($q_d{=}\lfloor\frac{n+3t}{2}\rfloor + 3d + 1, q_f{=}\lfloor\frac{n+t}{2}\rfloor + 1, single{=}$`false`).

(1) **operation** mbrb_broadcast(m, sn) **is** broadcast(INIT(m, sn)).

(2) **when** INIT(m, sn) **is** received **from** p_j **do** $obj_{\rm W}.k2\ell$_cast(WITNESS(m), (sn, j)).

(3) **when** (WITNESS(m), (sn, j)) **is** $obj_{\rm W}.k2\ell$_delivered **do** mbrb_deliver(m, sn, j).

Signature-Based $k2\ell$-Cast Implementation (1/2)

object SigBasedK2LCast(q_d) **is**

(1) **operation** $k2\ell$_cast(m, id) **is**
(2) **if** $\big((-, id)$ not already signed by $p_i\big)$ **then**
(3) $sig_i \leftarrow$ signature of (m, id) by p_i;
(4) $sigs_i \leftarrow \{$all valid signatures for (m, id) broadcast by $p_i\} \cup \{sig_i\}$;
(5) broadcast(BUNDLE$(m, id, sigs_i)$);
(6) check_delivery()
(7) **end if**.

(8) **when** BUNDLE$(m, id, sigs)$ **is** received **do**
(9) **if** $\big(sigs$ contains valid signatures for (m, id) not already broadcast by $p_i\big)$ **then**
(10) $sigs_i \leftarrow \{$all valid signatures for (m, id) broadcast by $p_i\}$
$\cup \{$all valid signatures for (m, id) in $sigs\}$;
(11) broadcast(BUNDLE$(m, id, sigs_i)$);
(12) check_delivery()
(13) **end if**.

(14) **internal operation** check_delivery() **is**
(15) **if** $\big(p_i$ broadcast at least q_d valid signatures for (m, id)
$\wedge\ (-, id)$ not already $k2\ell$-delivered$\big)$
(16) **then** $k2\ell$_deliver(m, id)
(17) **end if**.

end object.

Signature-Based $k2\ell$-Cast Implementation (2/2)

Parameter:

- q_d: nb of matching signatures that must be received in order to $k2\ell$-deliver an app-message

No q_f (no forwarding) and no *single* (only a single app-message can be endorsed)

Guarantees:

- $k' = q_d - n + c$
- $\delta = q_d > \frac{n+t}{2}$
- $k = q_d$
- $\ell = c - d$ (optimal)

This algorithm can still be greatly improved (forwarding, better round and message complexity...)

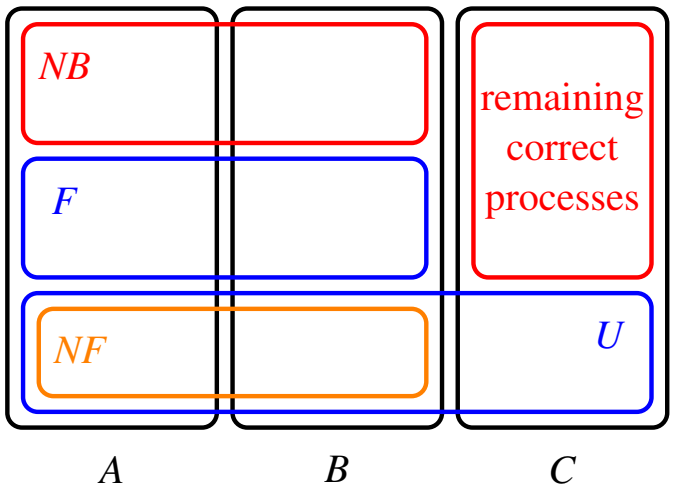
Liveness of Signature-Free $k2\ell$-Cast (1/3)

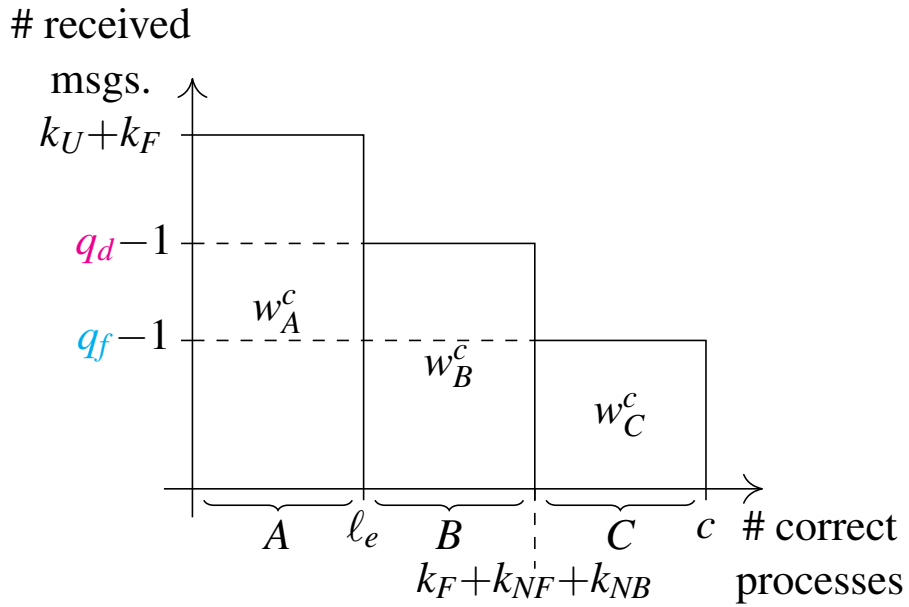
Let us assume that k_I correct processes $k2\ell$-cast (m, id)

We define 7 sets of correct processes:

- A: processes that reach the delivery threshold q_d ($|A| = \ell_e$)
- B: processes that reach the forwarding threshold q_f but not q_d
- C: remaining processes that do not reach q_f
- U: processes that broadcast $\text{MSG}(m, id)$ after a $k2\ell$-cast
- F: processes of $A \cup B$ that forward $\text{MSG}(m, id)$
- NF: processes of $A \cup B$ that broadcast $\text{MSG}(m, id)$ after a $k2\ell$-cast
- NB: processes of $A \cup B$ that never broadcast $\text{MSG}(m, id)$, because they have already broadcast some other $\text{MSG}(m' \neq m, id)$

Liveness of Signature-Free $k2\ell$-Cast (2/3)





w_A^c, w_B^c, w_C^c: nb of $\text{MSG}(m, id)$ received by A, B and C

$$w_A^c \leq (k_U + k_F)\ell_e \quad (1)$$

$$w_B^c \leq (q_d - 1)(k_{NF} + k_{NB} + k_F - \ell_e) \quad (2)$$

$$w_C^c \leq (q_f - 1)(c - k_{NF} - k_{NB} - k_F) \quad (3)$$

Liveness of Signature-Free $k2\ell$-Cast (3/3)

Given that the MA can suppress d copies of each $\text{MSG}(m, id)$ broadcast, we have

$$w_A^c + w_B^c + w_C^c \geq (k_U + k_F)(c - d) \tag{4}$$

By combining (1), (2), (3) and (4) we can obtain

$$\begin{aligned} \ell_e \times (k_U + k_F - q_d + 1) \geq (k_U + k_F)(c - d - q_d + q_f) \\ - c(q_f - 1) - k_{NB}(q_d - q_f) \end{aligned} \tag{5}$$

Proving liveness: For each property to prove, plug the values obtained from the hypotheses in the above equation, and derive

Signature-Free $k2\ell$-Cast Implementation (Cont'd)

Parameters:

- q_d: nb of $\text{MSG}(m, id)$ to receive to $k2\ell$_deliver(m, id)
- q_f: nb of $\text{MSG}(m, id)$ to receive to forward $\text{MSG}(m, id)$
- *single*: `true` if processes can send only one $\text{MSG}(-, id)$, `false` otherwise

Guarantees:

- $k' = q_f - n + c$
- $\delta = (q_f > \frac{n+t}{2}) \vee (single \wedge q_d > \frac{n+t}{2})$
- $k = \lfloor \frac{c(q_f-1)}{c-d-q_d+q_f} \rfloor + 1$
- $\ell = c - \lfloor \frac{cd}{c-q_d+1} \rfloor$

$k2\ell$-Cast Variables

Variable	Meaning	Best value
k'	minimal nb of correct $k2\ell$-casts if there is a correct $k2\ell$-delivery	High
δ	`true` iff no-duplicity is guaranteed, `false` otherwise	`true`
k	minimal nb of correct processes that $k2\ell$-cast a message	Low
ℓ	minimal nb of correct processes that $k2\ell$-deliver a message	High

$k2\ell$-Cast Safety

$k2\ell$-Validity:
If a correct process p_i $k2\ell$-delivers (m, id), then $k' < c$ correct processes $k2\ell$-cast (m, id)

$k2\ell$-No-duplication:
A correct process p_i $k2\ell$-delivers at most one $(-, id)$

$k2\ell$-Conditional-no-duplicity:
If $\delta = \texttt{true}$, then no two correct processes $k2\ell$-deliver (m, id) and $(m' \neq m, id)$

$k2\ell$-Cast Liveness

$k2\ell$-Local-delivery:
If $k \leq c$ correct processes $k2\ell$-cast (m, id) and no correct process $k2\ell$-casts $(m' \neq m, id)$, then at least one correct process p_i $k2\ell$-delivers (m, id)

$k2\ell$-Weak-global-delivery:
If a correct process p_i $k2\ell$-delivers (m, id), then at least $\ell \leq c - d$ correct processes $k2\ell$-deliver $(-, id)$

$k2\ell$-Strong-global-delivery:
If a correct process p_i $k2\ell$-delivers (m, id) and no correct process $k2\ell$-casts $(m' \neq m, id)$, then at least $\ell \leq c - d$ correct processes $k2\ell$-deliver (m, id)