# The Fault-Tolerant Cluster-Sending Problem

*Jelle Hellings*[1] Mohammad Sadoghi[2]

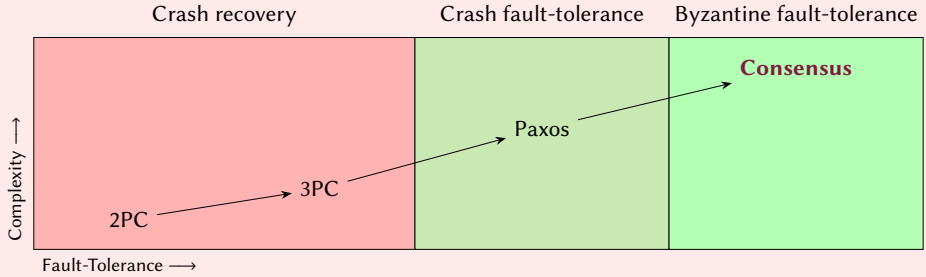
[1]McMaster University, 1280 Main St. W., Hamilton, ON L8S 4L7, Canada


[2]Department of Computer Science, University of California, Davis, USA

# Fault-Tolerance

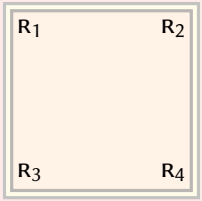
Distributed systems that can deal with replica failures (of any kind).

- Service continuity during (Byzantine) failures.
- Federated data management.



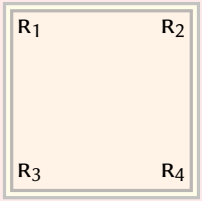
# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).



# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).

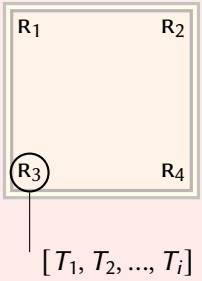



Each replica maintains the same state (consistency).

Useful abstraction: each replica maintains a ledger, an append-only sequence of transactions.

# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).




Each replica maintains the same state (consistency).

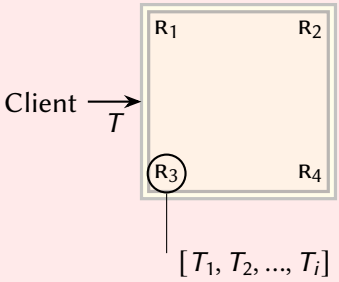
Useful abstraction: each replica maintains a ledger, an append-only sequence of transactions.

## Maintaining a Ledger via Consensus

1. Each replica holds the ledger of transactions.

# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).




Each replica maintains the same state (consistency).

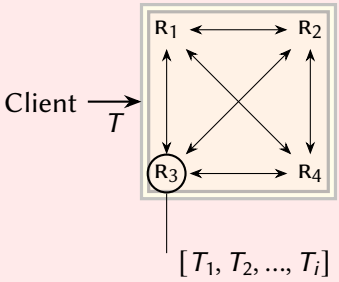
Useful abstraction: each replica maintains a ledger, an append-only sequence of transactions.

## Maintaining a Ledger via Consensus

1. Each replica holds the ledger of transactions.
2. Clients request new transactions.

# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).




Each replica maintains the same state (consistency).

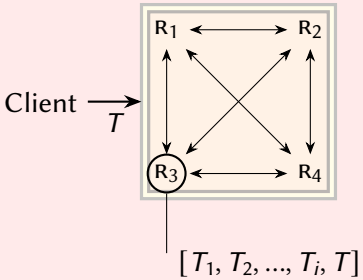
Useful abstraction: each replica maintains a ledger, an append-only sequence of transactions.

## Maintaining a Ledger via Consensus

1. Each replica holds the ledger of transactions.
2. Clients request new transactions.
3. Transaction agreement via *consensus*.

# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).




Each replica maintains the same state (consistency).

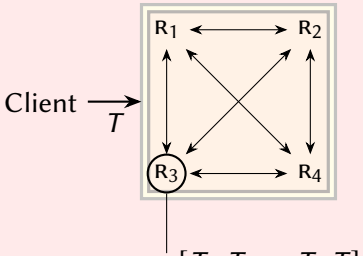
Useful abstraction: each replica maintains a ledger, an append-only sequence of transactions.

## Maintaining a Ledger via Consensus

1. Each replica holds the ledger of transactions.
2. Clients request new transactions.
3. Transaction agreement via *consensus*.
4. Append-only updates to ledger.

# Fault-Tolerant Data Management using Consensus

Distributed systems that can deal with replica failures (of any kind).




Each replica maintains the same state (consistency).

Useful abstraction: each replica maintains a ledger, an append-only sequence of transactions.

## Maintaining a Ledger via Consensus

1. Each replica holds the ledger of transactions.
2. Clients request new transactions.
3. Transaction agreement via *consensus*.
4. Append-only updates to ledger.

Design issue: More replicas *decreases* performance, no obvious scalability.

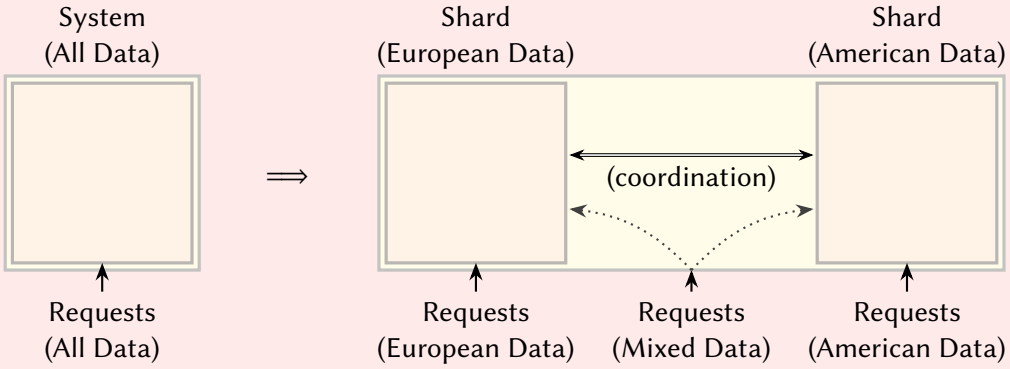
# Scalability in Data Management

Break-up system into mostly independently-operating parts with distinct roles.

# Scalability in Data Management

Break-up system into mostly independently-operating parts with distinct roles.

## Example: (geo-aware) sharding

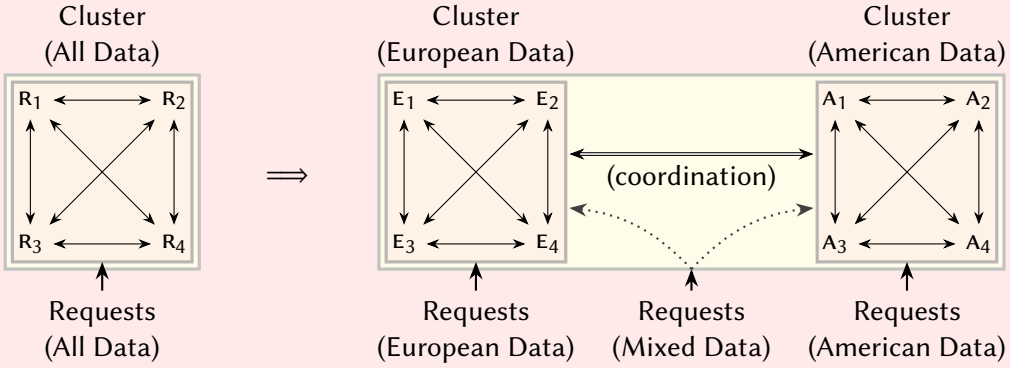



Adding shards ⟹ adding throughput (parallel processing), adding storage.

# Scalability in Data Management

Break-up system into mostly independently-operating parts with distinct roles.

## Example: (geo-aware) sharding




Idea: make each shard an independent fault-tolerant cluster.

Requires: *reliable communication between fault-tolerant clusters*.

# Outline of this Talk

Motivation for *cluster-sending*: reliable communication between fault-tolerant clusters.

# Outline of this Talk

Motivation for *cluster-sending*: reliable communication between fault-tolerant clusters.

## Remainder of the talk

- Formalization of cluster-sending.
- Straightforward solutions to the cluster-sending problem.
- Lower-bound on the cluster-sending problem.
- Worst-case optimal cluster-sending protocols.
- Evaluation.
- Conclusion.

# Reliable Communication between Fault-Tolerant Clusters

## Definition

Let $C_1, C_2$ be two clusters, both having non-faulty replicas.

The *cluster-sending problem* is the problem of sending a value $v$ from $C_1$ to $C_2$ such that:

1. non-faulty replicas in $C_2$ *receive* $v$;
2. non-faulty replicas in $C_1$ *confirm* that $v$ was received by the non-faulty replicas in $C_2$;
3. replicas in $C_2$ only receive $v$ if all non-faulty replicas in $C_1$ *agree* upon sending $v$.

# Reliable Communication between Fault-Tolerant Clusters

## Definition

Let $C_1, C_2$ be two clusters, both having non-faulty replicas.

The *cluster-sending problem* is the problem of sending a value $v$ from $C_1$ to $C_2$ such that:

1. non-faulty replicas in $C_2$ *receive* $v$;
2. non-faulty replicas in $C_1$ *confirm* that $v$ was received by the non-faulty replicas in $C_2$;
3. replicas in $C_2$ only receive $v$ if all non-faulty replicas in $C_1$ *agree* upon sending $v$.

## Informal Definition

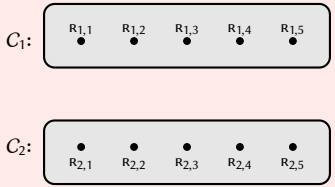
Successfully sending a value $v$ from a cluster $C_1$ to a $C_2$ without any faulty replicas being able to *disrupt sending* or send *alternative forged values*.

# Basic Cluster-Sending via Broadcasting

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica in $C_1$ has a *certificate* $\langle v \rangle_{C_1}$ that proves agreement.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.

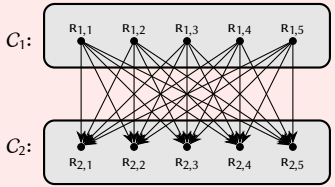


# Basic Cluster-Sending via Broadcasting

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica in $C_1$ has a *certificate* $\langle v \rangle_{C_1}$ that proves agreement.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



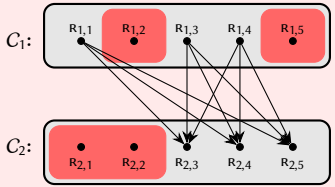

*Broadcast*: every replica in $C_1$ sends pairs $(v, \langle v \rangle_{C_1})$ to every replica in $C_2$.

# Basic Cluster-Sending via Broadcasting

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica in $C_1$ has a *certificate* $\langle v \rangle_{C_1}$ that proves agreement.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



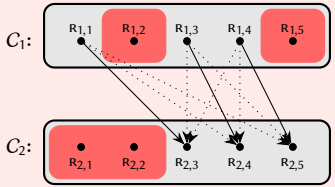

Faulty replicas can *fail* to send (in $C_1$) or to receive (in $C_2$).

# Basic Cluster-Sending via Broadcasting

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica in $C_1$ has a *certificate* $\langle v \rangle_{C_1}$ that proves agreement.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



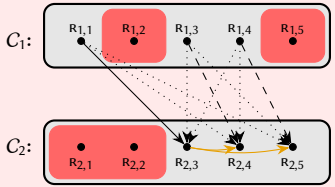

Non-faulty replicas in $C_2$ only need at-least one message $(v, \langle v \rangle_{C_1})$.

# Basic Cluster-Sending via Broadcasting

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica in $C_1$ has a *certificate* $\langle v \rangle_{C_1}$ that proves agreement.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



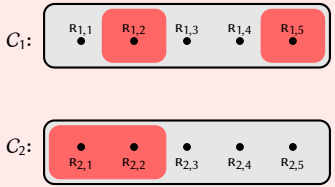

Replicas in $C_2$ can redistribute $(v, \langle v \rangle_{C_1})$.

# Basic Cluster-Sending via Broadcasting

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica in $C_1$ has a *certificate* $\langle v \rangle_{C_1}$ that proves agreement.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



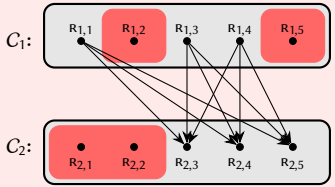

With certificates: a *single* message between non-faulty sender and receiver is sufficient!

# Basic Cluster-Sending via Broadcasting (Without Certificates)

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica $\mathrm{R} \in C_1$ can only *claim* agreement via a digital signature $\langle v \rangle_{\mathrm{R}}$.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.

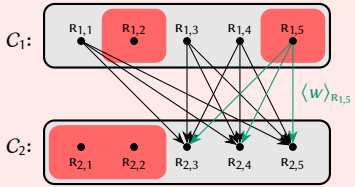


# Basic Cluster-Sending via Broadcasting (Without Certificates)

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica $\mathrm{R} \in C_1$ can only *claim* agreement via a digital signature $\langle v \rangle_{\mathrm{R}}$.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



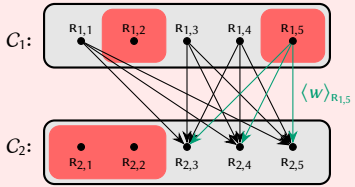

Faulty replicas can *lie* and send $\langle w \rangle_{\mathrm{R}}$ without agreement on $w$.

# Basic Cluster-Sending via Broadcasting (Without Certificates)

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica $\mathrm{R} \in C_1$ can only *claim* agreement via a digital signature $\langle v \rangle_{\mathrm{R}}$.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



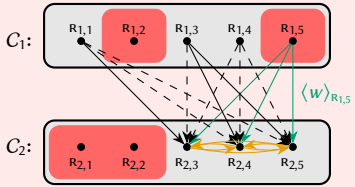

Claims from *three* distinct replicas in $C_1$: at-least one from a non-faulty replica.

# Basic Cluster-Sending via Broadcasting (Without Certificates)

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica $\mathrm{R} \in C_1$ can only *claim* agreement via a digital signature $\langle v \rangle_{\mathrm{R}}$.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.



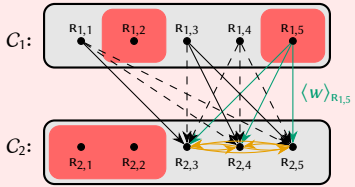

Replicas in $C_2$ can redistribute $(v, \langle v \rangle_{\mathrm{R}})$.

# Basic Cluster-Sending via Broadcasting (Without Certificates)

*Goal*: send a value $v$ from cluster $C_1$ to cluster $C_2$.

## Assumptions

- Every replica $\mathrm{R} \in C_1$ can only *claim* agreement via a digital signature $\langle v \rangle_{\mathrm{R}}$.
- Communication is *reliable*.
- At-most *two* replicas faulty in each cluster.




Without certificates: *at least* $\mathbf{f}_{C_1} + 1$ distinct received messages by non-faulty senders!

# Efficient Cluster-Sending

Cluster-Sending via broadcasting: straightforward, *not efficient*:

- With certificates: $(\mathbf{f}_{C_1} + 1)(\mathbf{f}_{C_2} + 1) \approx \mathbf{f}_{C_1} \times \mathbf{f}_{C_2}$ messages.
- With claims: $(2\mathbf{f}_{C_1} + 1)(\mathbf{f}_{C_2} + 1) \approx 2\mathbf{f}_{C_1} \times \mathbf{f}_{C_2}$ messages.

# Efficient Cluster-Sending

Cluster-Sending via broadcasting: straightforward, *not efficient*:

- With certificates: $(\mathbf{f}_{C_1} + 1)(\mathbf{f}_{C_2} + 1) \approx \mathbf{f}_{C_1} \times \mathbf{f}_{C_2}$ messages.
- With claims: $(2\mathbf{f}_{C_1} + 1)(\mathbf{f}_{C_2} + 1) \approx 2\mathbf{f}_{C_1} \times \mathbf{f}_{C_2}$ messages.

## Local communication versus global communication

| | *Ping round-trip times (*ms*)* | | | | | | *Bandwidth (*Mbit/s*)* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OR | IA | Mont. | BE | TW | Syd. | OR | IA | Mont. | BE | TW | Syd. |
| Oregon | ≤ 1 | 38 | 65 | 136 | 118 | 161 | 7998 | 669 | 371 | 194 | 188 | 136 |
| Iowa | | ≤ 1 | 33 | 98 | 153 | 172 | | 10004 | 752 | 243 | 144 | 120 |
| Montreal | | | ≤ 1 | 82 | 186 | 202 | | | 7977 | 283 | 111 | 102 |
| Belgium | | | | ≤ 1 | 252 | 270 | | | | 9728 | 79 | 66 |
| Taiwan | | | | | ≤ 1 | 137 | | | | | 7998 | 160 |
| Sydney | | | | | | ≤ 1 | | | | | | 7977 |

*Goal*: Minimize communication *between* clusters.

# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.

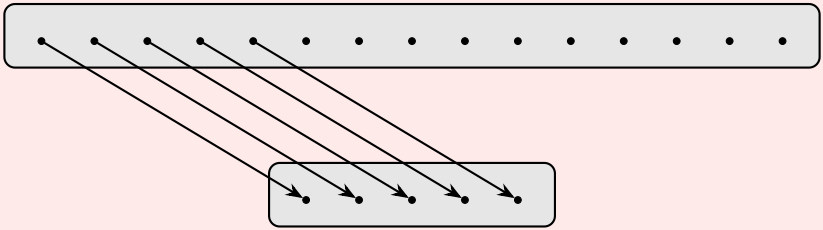
# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\quad$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\quad$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.



Minimize impact of faulty replicas: minimum number of messages per participant.

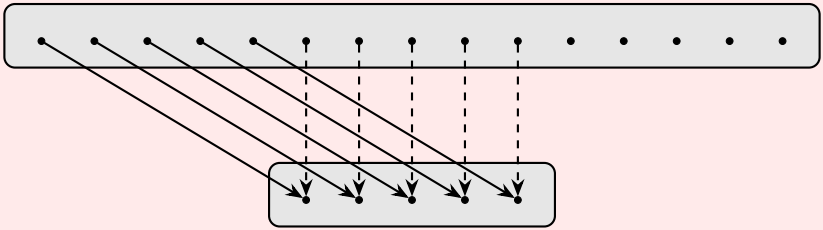
# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\quad$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\quad$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.



Minimize impact of faulty replicas: minimum number of messages per participant.

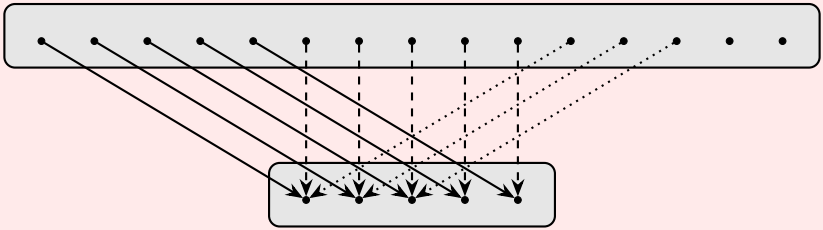
# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.



Minimize impact of faulty replicas: minimum number of messages per participant.

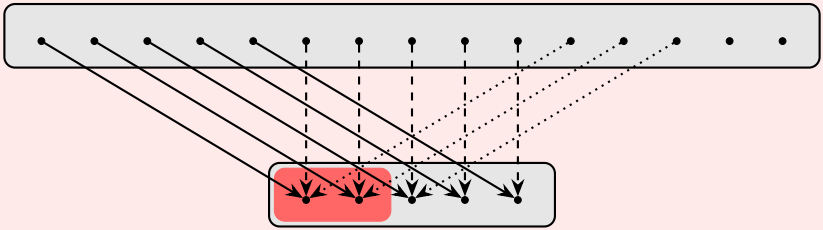
# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\quad$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\quad$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.



Any $\mathbf{f}_{C_2}$ replicas in $C_2$ can be faulty: top $\mathbf{f}_{C_2}$ receivers receive at-least 6 messages.

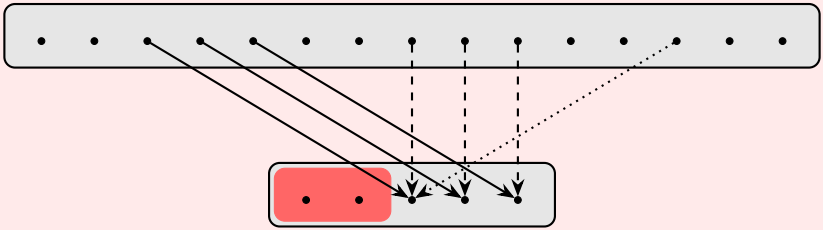
# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\quad$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\quad$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.



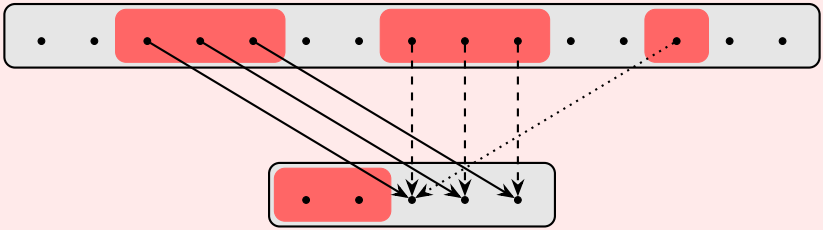
# Towards a Lower-Bound for Cluster-Sending (Example)

$\mathbf{n}_{C_1} = 15$ $\quad$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\quad$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.



Only $\mathbf{f}_{C_1}$ messages remaining, can all be sent by faulty replicas in $C_1$.

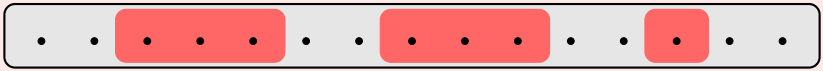
# Towards a Lower-Bound for Cluster-Sending (Example)

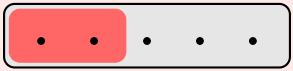
$\mathbf{n}_{C_1} = 15$ $\quad$ $\mathbf{f}_{C_1} = 7$

$\mathbf{n}_{C_2} = 5$ $\quad$ $\mathbf{f}_{C_2} = 2$

## Proposition (assuming certificates)

Any correct algorithm needs to send at least 14 *messages*.





# Lower-Bound for Cluster-Sending with Certificates

## Basic Idea

- One message needs to be exchanged between a non-faulty sender and receiver.
- Have to deal with size imbalances between clusters.

# Lower-Bound for Cluster-Sending with Certificates

## Basic Idea

- One message needs to be exchanged between a non-faulty sender and receiver.
- Have to deal with size imbalances between clusters.

## Theorem

*Let $C_1, C_2$ be two clusters and let $\{i, j\} = \{1, 2\}$ such that $\mathbf{n}_{C_i} \geq \mathbf{n}_{C_j}$. Let*

$$q_i = (\mathbf{f}_{C_i} + 1) \operatorname{div} \mathbf{nf}_{C_j},$$
$$r_i = (\mathbf{f}_{C_i} + 1) \bmod \mathbf{nf}_{C_j},$$
$$\sigma_i = q_i \mathbf{n}_{C_j} + r_i + \mathbf{f}_{C_j} \operatorname{sgn} r_i.$$

*Any protocol that solves the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$ needs to exchange at least $\sigma_i$ messages.*

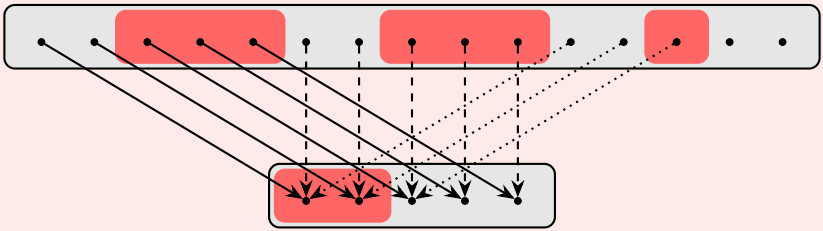
# Lower-Bound for Cluster-Sending with Certificates (Example)

## Theorem

*Let $C_1, C_2$ be two clusters and let*

$$q_1 = (\mathbf{f}_{C_1} + 1) \operatorname{div} \mathbf{nf}_{C_2} = 7 \operatorname{div} 3 = 2,$$
$$r_1 = (\mathbf{f}_{C_1} + 1) \bmod \mathbf{nf}_{C_2} = 7 \bmod 3 = 1,$$
$$\sigma_1 = q_1 \mathbf{n}_{C_2} + r_1 + \mathbf{f}_{C_2} \operatorname{sgn} r_1 = 2 \cdot 5 + 1 + 3 = 14.$$

*Any protocol that solves the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$ needs to exchange at least $\sigma_1 = 14$ messages.*



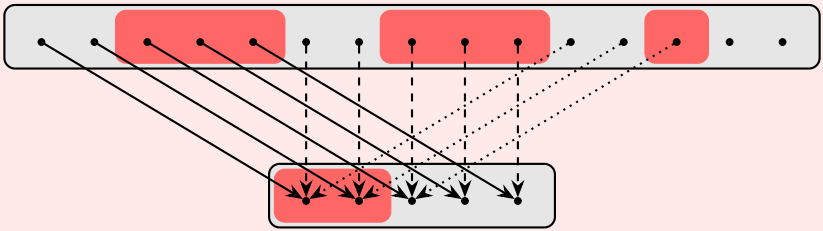
# Lower-Bound for Cluster-Sending with Certificates (Example)

## Theorem

*Let $C_1, C_2$ be two clusters and let*

$$q_1 = (\mathbf{f}_{C_1} + 1) \operatorname{div} \mathbf{nf}_{C_2} = 7 \operatorname{div} 3 = 2,$$
$$r_1 = (\mathbf{f}_{C_1} + 1) \bmod \mathbf{nf}_{C_2} = 7 \bmod 3 = 1,$$
$$\sigma_1 = \boxed{q_1 \mathbf{n}_{C_2}} + r_1 + \mathbf{f}_{C_2} \operatorname{sgn} r_1 = \boxed{2 \cdot 5} + 1 + 3 = 14.$$

*Any protocol that solves the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$ needs to exchange at least $\sigma_1 = 14$ messages.*



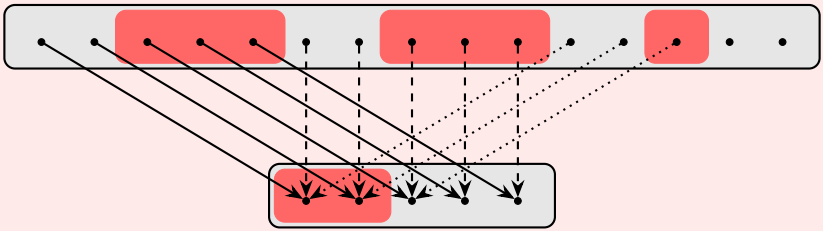
# Lower-Bound for Cluster-Sending with Certificates (Example)

## Theorem

*Let $C_1, C_2$ be two clusters and let*

$$q_1 = (\mathbf{f}_{C_1} + 1) \operatorname{div} \mathbf{nf}_{C_2} = 7 \operatorname{div} 3 = 2,$$
$$r_1 = (\mathbf{f}_{C_1} + 1) \bmod \mathbf{nf}_{C_2} = 7 \bmod 3 = 1,$$
$$\sigma_1 = q_1 \mathbf{n}_{C_2} + \boxed{r_1 + \mathbf{f}_{C_2} \operatorname{sgn} r_1} = 2 \cdot 5 + \boxed{1 + 3} = 14.$$

*Any protocol that solves the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$ needs to exchange at least $\sigma_1 = 14$ messages.*



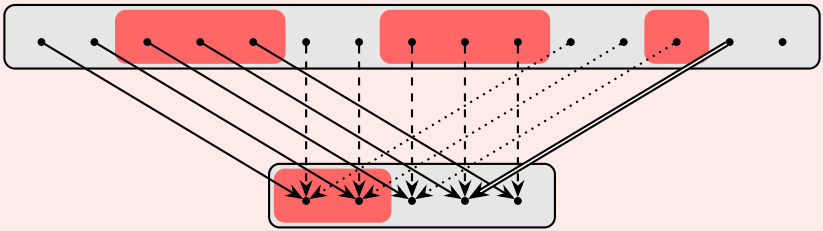
# Lower-Bound for Cluster-Sending with Certificates (Example)

## Theorem

*Let $C_1, C_2$ be two clusters and let*

$$q_1 = (\mathbf{f}_{C_1} + 1) \operatorname{div} \mathbf{nf}_{C_2} = 7 \operatorname{div} 3 = 2,$$
$$r_1 = (\mathbf{f}_{C_1} + 1) \bmod \mathbf{nf}_{C_2} = 7 \bmod 3 = 1,$$
$$\sigma_1 = q_1 \mathbf{n}_{C_2} + r_1 + \mathbf{f}_{C_2} \operatorname{sgn} r_1 = 2 \cdot 5 + 1 + 3 = 14.$$

*Any protocol that solves the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$ needs to exchange at least $\sigma_1 =$ 14 messages.*



# Lower-Bound for Cluster-Sending with Claims

## Basic Idea

- $\mathbf{f}_{C_1} + 1$ message needs to be sent by distinct non-faulty senders to non-faulty receiver.
- Have to deal with size imbalances between clusters.

## Theorem

*Let $C_1, C_2$ be two clusters and let $\{i, j\} = \{1, 2\}$ such that $\mathbf{n}_{C_i} \geq \mathbf{n}_{C_j}$. Let*

$$q_1 = (2\mathbf{f}_{C_1} + 1) \operatorname{div} \mathbf{nf}_{C_2}, \qquad q_2 = (\mathbf{f}_{C_2} + 1) \operatorname{div} (\mathbf{nf}_{C_1} - \mathbf{f}_{C_1})$$
$$r_1 = (2\mathbf{f}_{C_1} + 1) \bmod \mathbf{nf}_{C_2}, \qquad r_2 = (\mathbf{f}_{C_2} + 1) \bmod (\mathbf{nf}_{C_1} - \mathbf{f}_{C_1})$$
$$\tau_1 = q_1\mathbf{n}_{C_2} + r_1 + \mathbf{f}_{C_2} \operatorname{sgn} r_1 \qquad \tau_2 = q_2\mathbf{n}_{C_1} + r_2 + 2\mathbf{f}_{C_1} \operatorname{sgn} r_2.$$

*Any protocol that solves the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$ needs to exchange at least $\tau_i$ messages.*

# Bijective Sending with Certificates

Assume $\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1 \leq \min(\mathbf{n}_{C_1}, \mathbf{n}_{C_2})$.

We have $\sigma_1 = \sigma_2 = \mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.

**Protocol for the sending cluster $C_1$:**

1: All replicas in $\mathrm{nf}(C_1)$ agree on $v$ and construct $\langle v \rangle_{C_1}$.
2: Choose replicas $S_1 \subseteq C_1$ and $S_2 \subseteq C_2$ with $\mathbf{n}_{S_2} = \mathbf{n}_{S_1} = \mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.
3: Choose a bijection $b : S_1 \longrightarrow S_2$.
4: **for** $\mathrm{R}_1 \in S_1$ **do**
5: $\quad \mathrm{R}_1$ sends $(v, \langle v \rangle_{C_1})$ to $b(\mathrm{R}_1)$.

**Protocol for the receiving cluster $C_2$:**

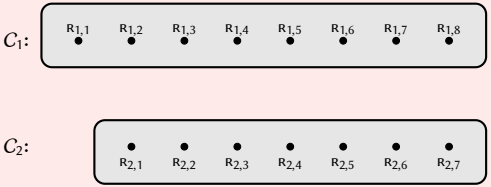
6: **event** $\mathrm{R}_2 \in \mathrm{nf}(C_2)$ receives $(w, \langle w \rangle_{C_1})$ from $\mathrm{R}_1 \in C_1$ **do**
7: $\quad$ Broadcast $(w, \langle w \rangle_{C_1})$ to all replicas in $C_2$.
8: **event** $\mathrm{R}'_2 \in \mathrm{nf}(C_2)$ receives $(w, \langle w \rangle_{C_1})$ from $\mathrm{R}_2 \in C_2$ **do**
9: $\quad \mathrm{R}'_2$ considers $w$ *received*.

# Bijective Sending with Certificates: Example

$\mathbf{n}_{C_1} = 8$ $\quad$ $\mathbf{f}_{C_1} = 3$

$\mathbf{n}_{C_2} = 7$ $\quad$ $\mathbf{f}_{C_2} = 2$

$\sigma_1 = 6.$

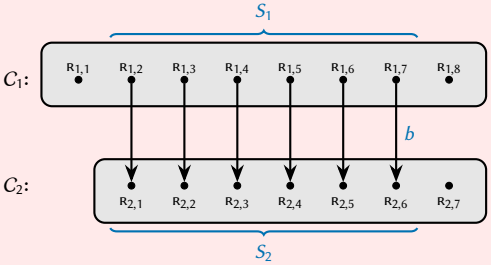


# Bijective Sending with Certificates: Example

$\mathbf{n}_{C_1} = 8$ $\quad$ $\mathbf{f}_{C_1} = 3$

$\mathbf{n}_{C_2} = 7$ $\quad$ $\mathbf{f}_{C_2} = 2$

$\sigma_1 = 6.$

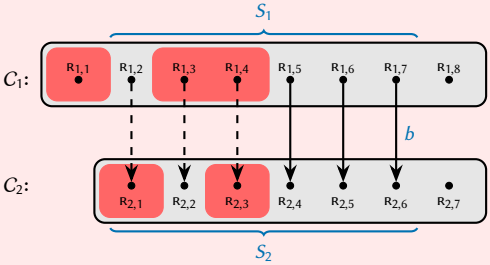


# Bijective Sending with Certificates: Example

$\mathbf{n}_{C_1} = 8$ $\quad \mathbf{f}_{C_1} = 3$

$\mathbf{n}_{C_2} = 7$ $\quad \mathbf{f}_{C_2} = 2$

$\sigma_1 = 6.$



# Bijective Sending with Claims

Assume $2\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1 \leq \min(\mathbf{n}_{C_1}, \mathbf{n}_{C_2})$.

We have $\tau_1 = \tau_2 = 2\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.

**Protocol for the sending cluster $C_1$:**

1: All replicas in $\mathrm{nf}(C_1)$ agree on $v$.
2: Choose replicas $S_1 \subseteq C_1$ and $S_2 \subseteq C_2$ with $\mathbf{n}_{S_2} = \mathbf{n}_{S_1} = 2\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.
3: Choose bijection $b : S_1 \to S_2$.
4: **for** $\mathrm{R}_1 \in S_1$ **do**
5: $\quad \mathrm{R}_1$ sends $(v, \langle v \rangle_{\mathrm{R}_1})$ to $b(\mathrm{R}_1)$.

**Protocol for the receiving cluster $C_2$:**

6: ....

# Bijective Sending with Claims

Assume $2\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1 \leq \min(\mathbf{n}_{C_1}, \mathbf{n}_{C_2})$.

We have $\tau_1 = \tau_2 = 2\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.

**Protocol for the sending cluster $C_1$:**

1: ....

**Protocol for the receiving cluster $C_2$:**

6: **event** $\mathrm{R}_2 \in \mathrm{nf}(C_2)$ receives $(w, \langle w \rangle_{\mathrm{R}'_1})$ from $\mathrm{R}'_1 \in C_1$ **do**

7: Broadcast $(w, \langle w \rangle_{\mathrm{R}'_1})$ to all replicas in $C_2$.

8: **event** $\mathrm{R}'_2 \in \mathrm{nf}(C_2)$ receives $\mathbf{f}_{C_1} + 1$ messages $(w, \langle w \rangle_{\mathrm{R}'_1})$:

(i) each message is sent by a replica in $C_2$;
(ii) each message carries the same value $w$; and
(iii) each message has a distinct signature $\langle w \rangle_{\mathrm{R}'_1}$, $\mathrm{R}'_1 \in C_1$

**do**

9: $\mathrm{R}'_2$ considers $w$ *received*.

# Generalizing Bijective Sending

Consider bijective sending from $C_1$ to $C_2$, $\mathbf{n}_{C_1} \geq \sigma_1 > \mathbf{n}_{C_2}$, with certificates.

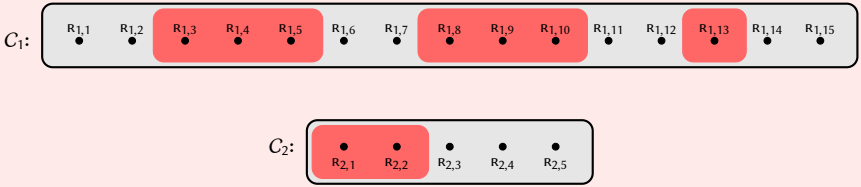
- Bijective sending requires $\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$ distinct replicas in both clusters.
- Restrictive: clusters of roughly the same size.

# Generalizing Bijective Sending

Consider bijective sending from $C_1$ to $C_2$, $\mathbf{n}_{C_1} \geq \sigma_1 > \mathbf{n}_{C_2}$, with certificates.

- Bijective sending requires $\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$ distinct replicas in both clusters.
- Restrictive: clusters of roughly the same size.

## Generalize bijective sending



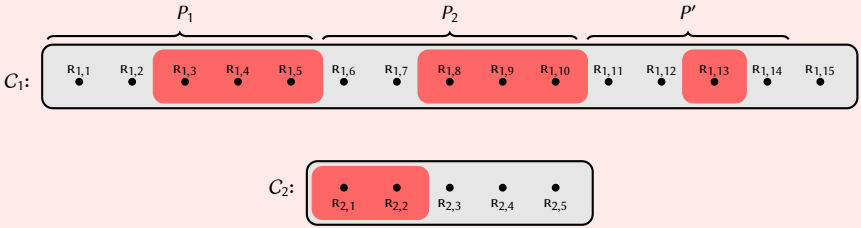
# Generalizing Bijective Sending

Consider bijective sending from $C_1$ to $C_2$, $\mathbf{n}_{C_1} \geq \sigma_1 > \mathbf{n}_{C_2}$, with certificates.

- Bijective sending requires $\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$ distinct replicas in both clusters.
- Restrictive: clusters of roughly the same size.

## Generalize bijective sending

- Partition $\sigma_1$ replicas of $C_1$ into $\mathbf{n}_{C_2}$-sized clusters.



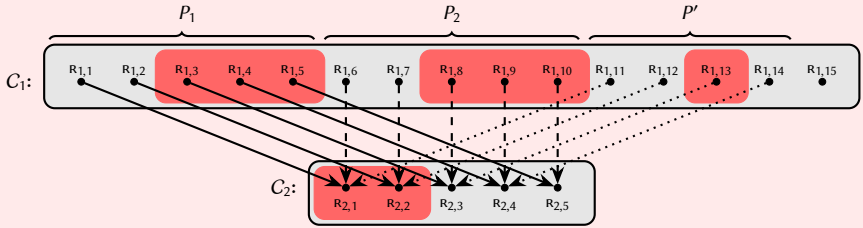
# Generalizing Bijective Sending

Consider bijective sending from $C_1$ to $C_2$, $\mathbf{n}_{C_1} \geq \sigma_1 > \mathbf{n}_{C_2}$, with certificates.

- Bijective sending requires $\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$ distinct replicas in both clusters.
- Restrictive: clusters of roughly the same size.

## Generalize bijective sending

- Partition $\sigma_1$ replicas of $C_1$ into $\mathbf{n}_{C_2}$-sized clusters.
- Bijective send from each cluster in the partition to $C_2$.



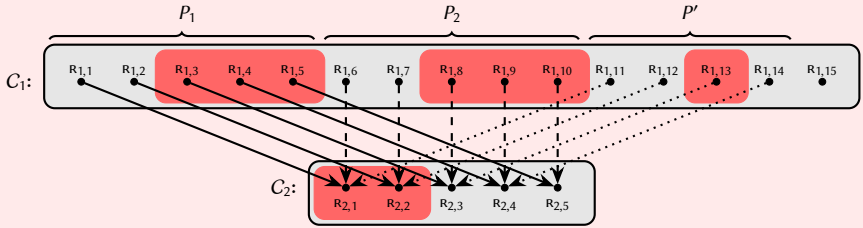
# Generalizing Bijective Sending

Consider bijective sending from $C_1$ to $C_2$, $\mathbf{n}_{C_1} \geq \sigma_1 > \mathbf{n}_{C_2}$, with certificates.

- Bijective sending requires $\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$ distinct replicas in both clusters.
- Restrictive: clusters of roughly the same size.

## Generalize bijective sending

- Partition $\sigma_1$ replicas of $C_1$ into $\mathbf{n}_{C_2}$-sized clusters.
- Bijective send from each cluster in the partition to $C_2$.
- $\mathbf{n}_{C_1} \geq \sigma_1$ holds always if $\mathbf{n}_{C_1} > 3\mathbf{f}_{C_1}$ and $\mathbf{n}_{C_2} > 3\mathbf{f}_{C_2}$.



# Partitioned Bijective Sending

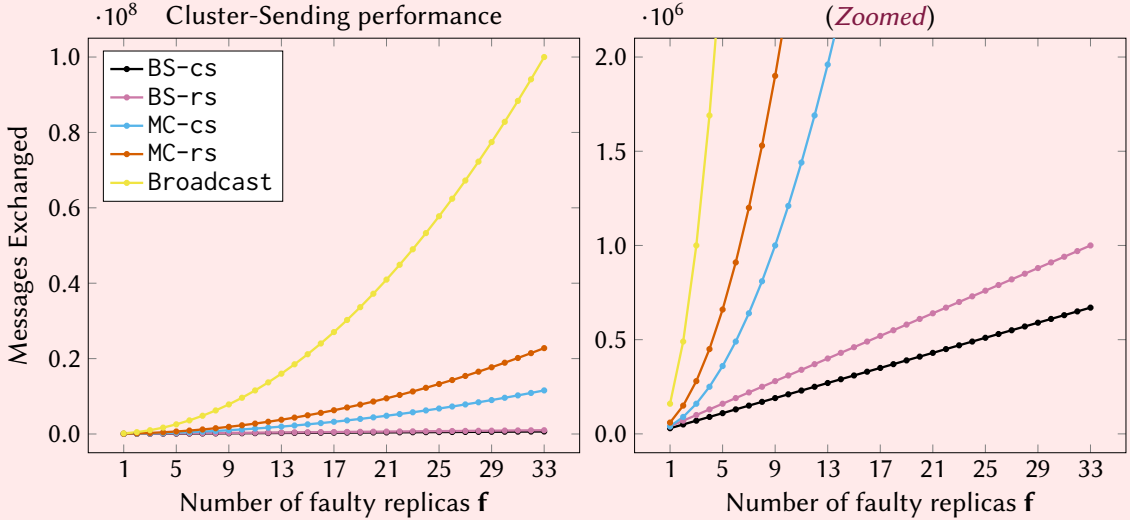
## Corollary

*Consider the cluster-sending problem in which $C_1$ sends a value $v$ to $C_2$.*

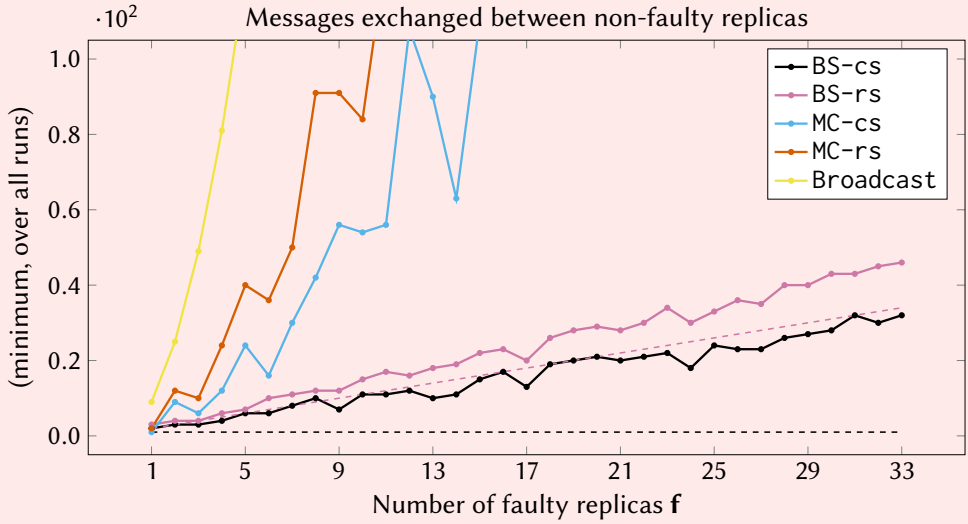
1. *If $\mathbf{n}_C > 3\mathbf{f}_C$ for all clusters $C$ and replicas only have crash failures or omit failures, then (partitioned) bijective sending solves cluster-sending with optimal message complexity.*
2. *If $\mathbf{n}_C > 3\mathbf{f}_C$ for all clusters $C$ and clusters can produce certificates, then (partitioned) bijective sending solves cluster-sending with optimal message complexity.*
3. *If $\mathbf{n}_C > 4\mathbf{f}_C$ for all clusters $C$ and replicas can digitally sign claims, then (partitioned) bijective sending solves cluster-sending with optimal message complexity.*

*These protocols solve cluster-sending using $O(\max(\mathbf{n}_{C_1}, \mathbf{n}_{C_2}))$ messages of size $O(\|v\|)$ each.*

# Performance of (Partitioned) Bijective Sending



# Performance of (Partitioned) Bijective Sending



# Cluster-sending: Can we do better?

## Pessimistic

**No**: these algorithms are worst-case optimal.

Cannot do better than *linear communication* in the size of the clusters.

# Cluster-sending: Can we do better?

## Pessimistic

**No**: these algorithms are worst-case optimal.
Cannot do better than *linear communication* in the size of the clusters.

## Probabilistic—upcoming results

**Yes**: if we randomly choose sender and receiver, then we often do much better!
Probabilistic approach: expected-case only *constant communication* (four steps).

# Conclusion

We studied reliable communication between fault-tolerant clusters by

- formalizing it as the *cluster-sending problem*;
- showing lower bounds on the complexity of this problem;
- providing optimal cluster-sending protocols for practical environments; and
- providing a small-scale evaluation of our techniques.

## More information

`https://jhellings.nl`.