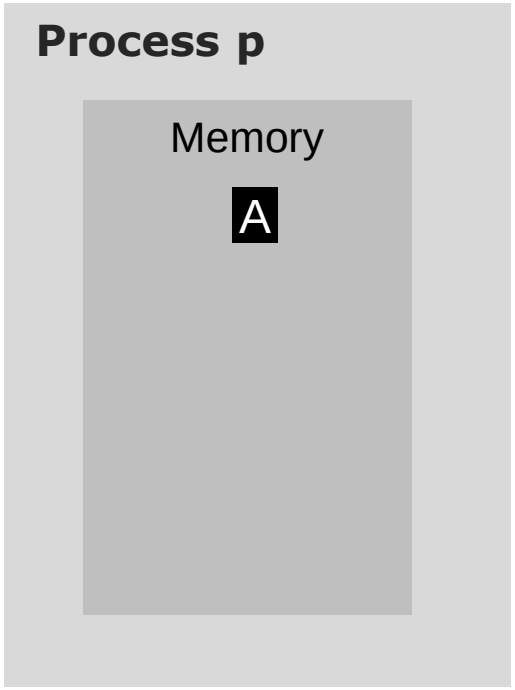
Fault Tolerance for Remote Memory Access Programming Models

MACIEJ BESTA, TORSTEN HOEFLER

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Remote Memory Access (RMA) Programming



Remote Memory Access (RMA) Programming

Remote Memory Access (RMA) Programming

Process p

Memory

A

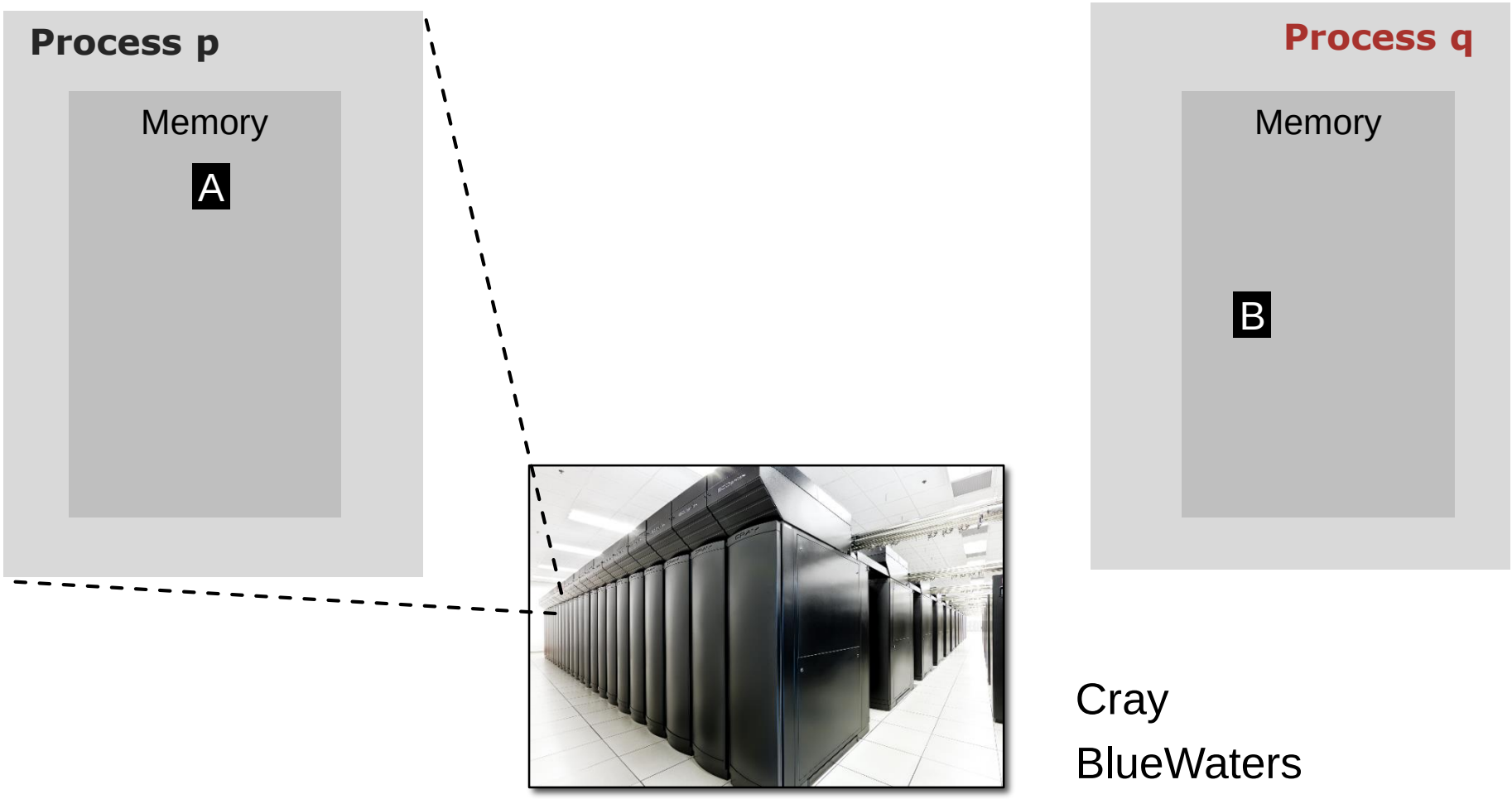
Process q

Memory

B

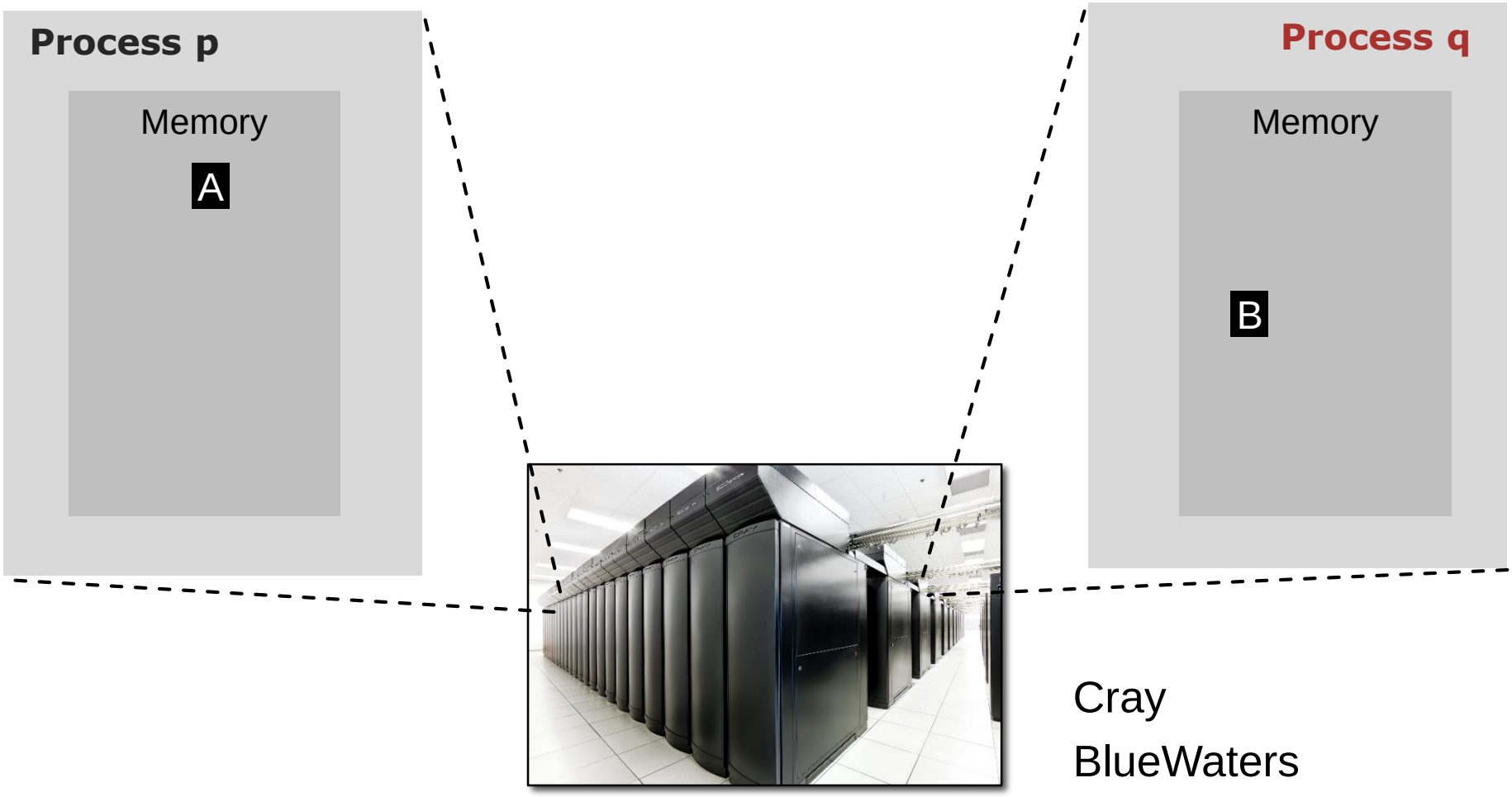
Cray BlueWaters

Remote Memory Access (RMA) Programming



Cray BlueWaters

Remote Memory Access (RMA) Programming



Remote Memory Access (RMA) Programming

Process p

Memory

A

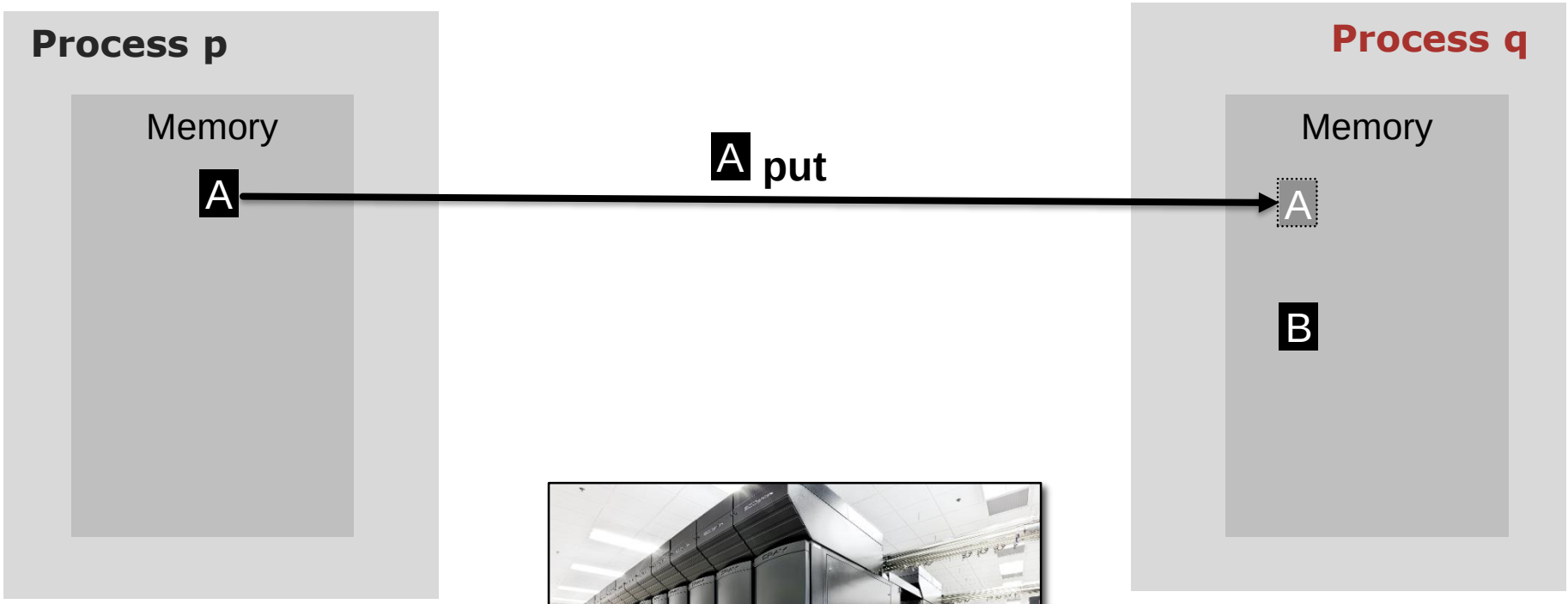
Process q

Memory

B

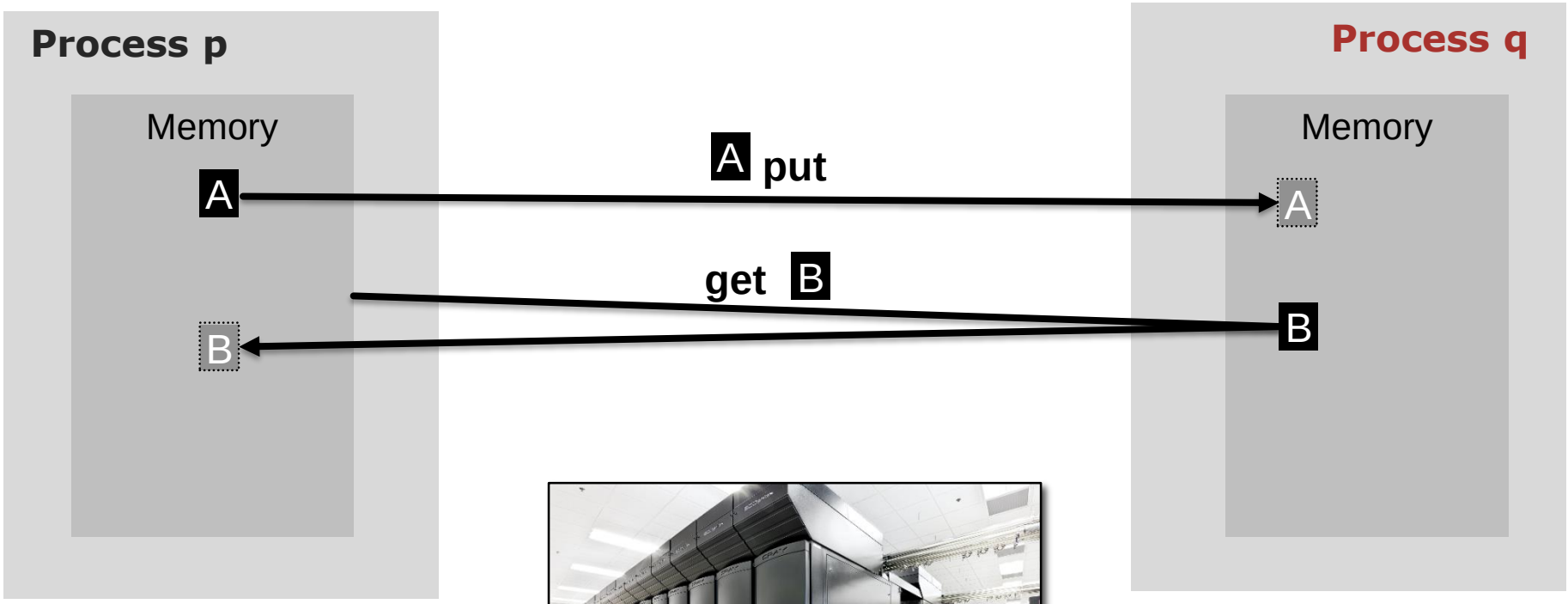
Cray BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING



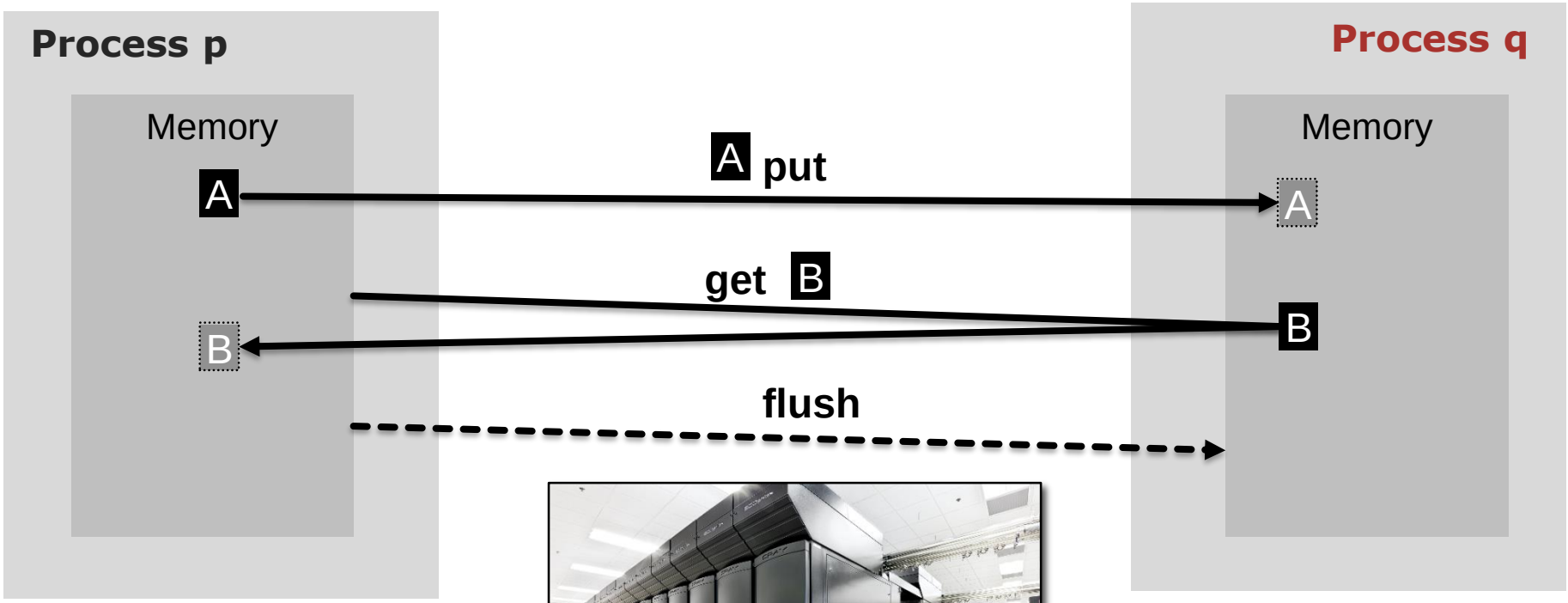
Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING



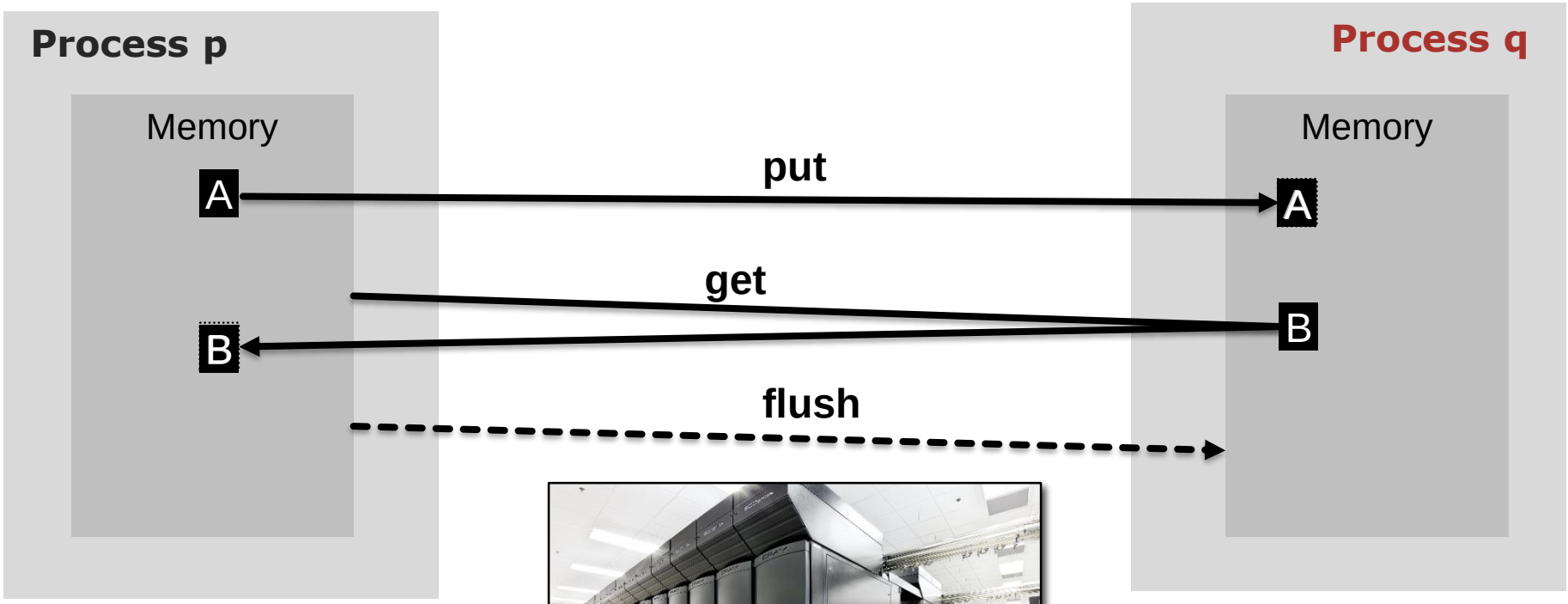
Cray BlueWaters

Remote Memory Access (RMA) Programming



Cray BlueWaters

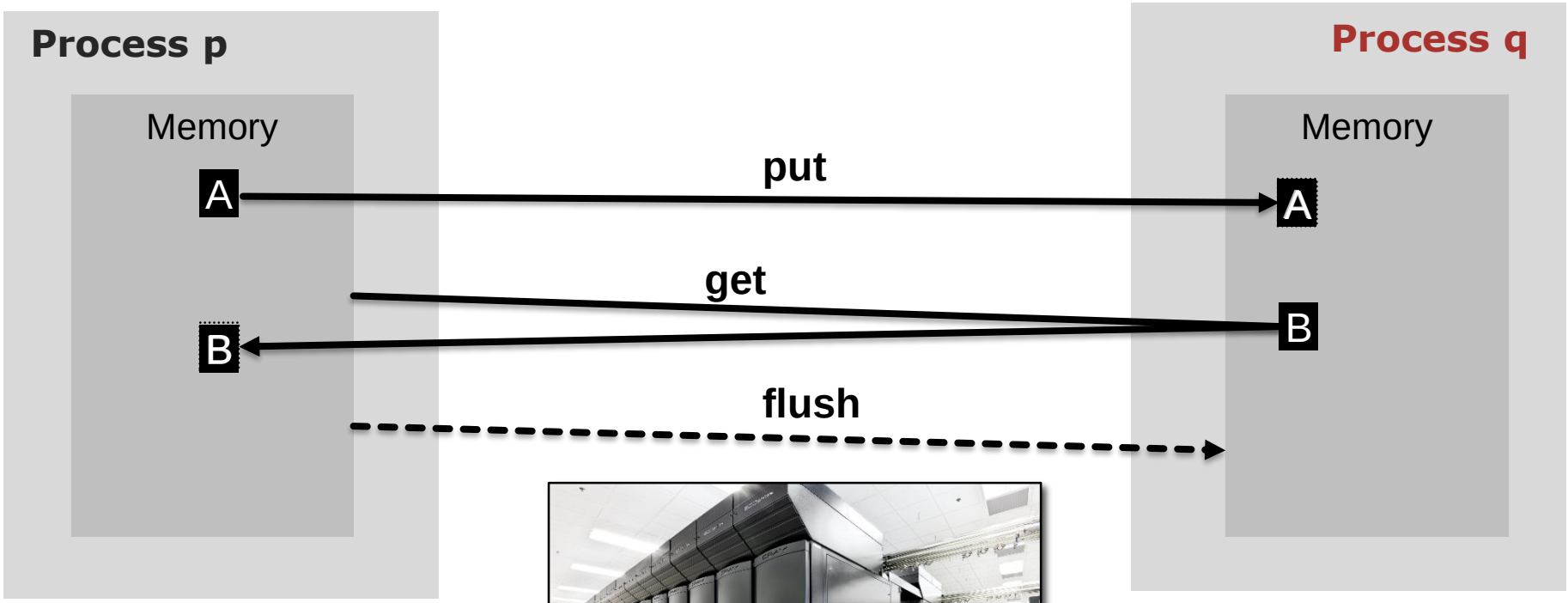
REMOTE MEMORY ACCESS (RMA) PROGRAMMING



Cray BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

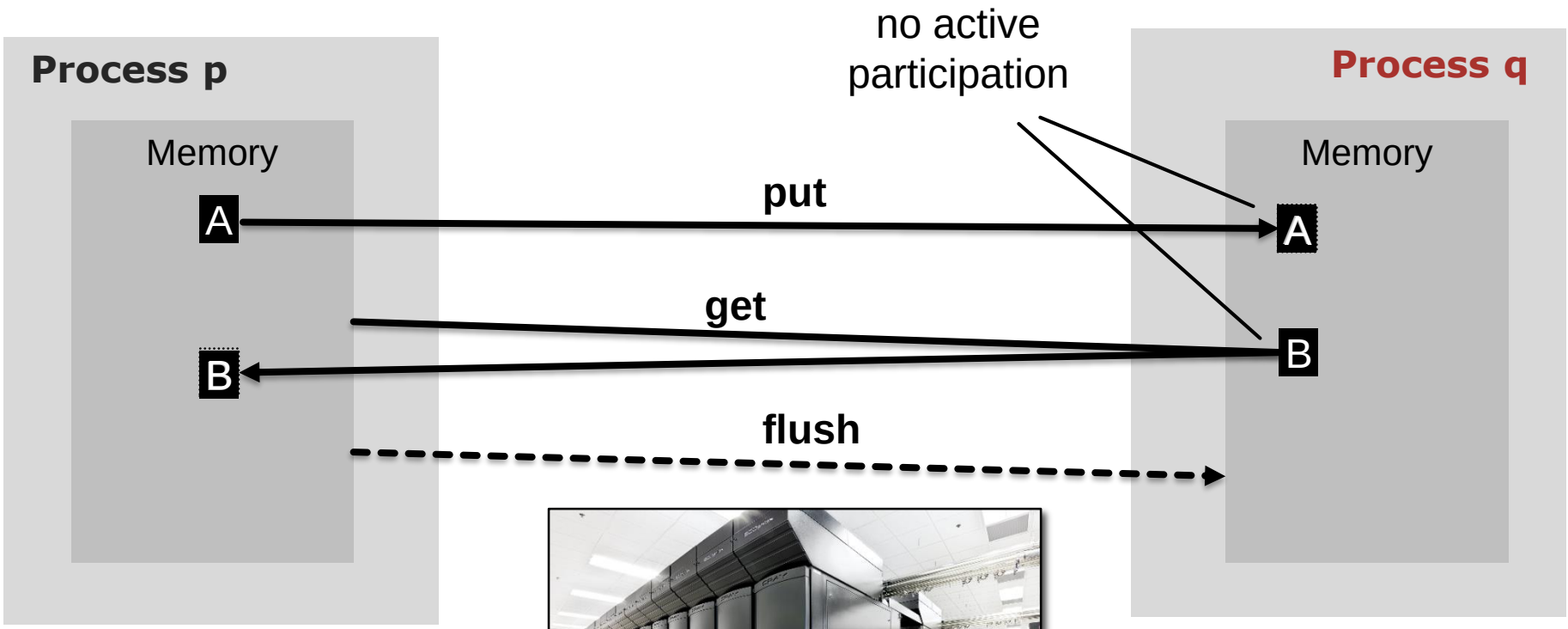
- One-sided communication



Cray BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

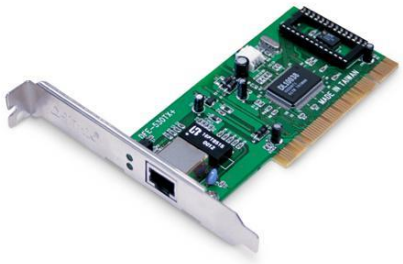
- One-sided communication



Cray BlueWaters

REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA

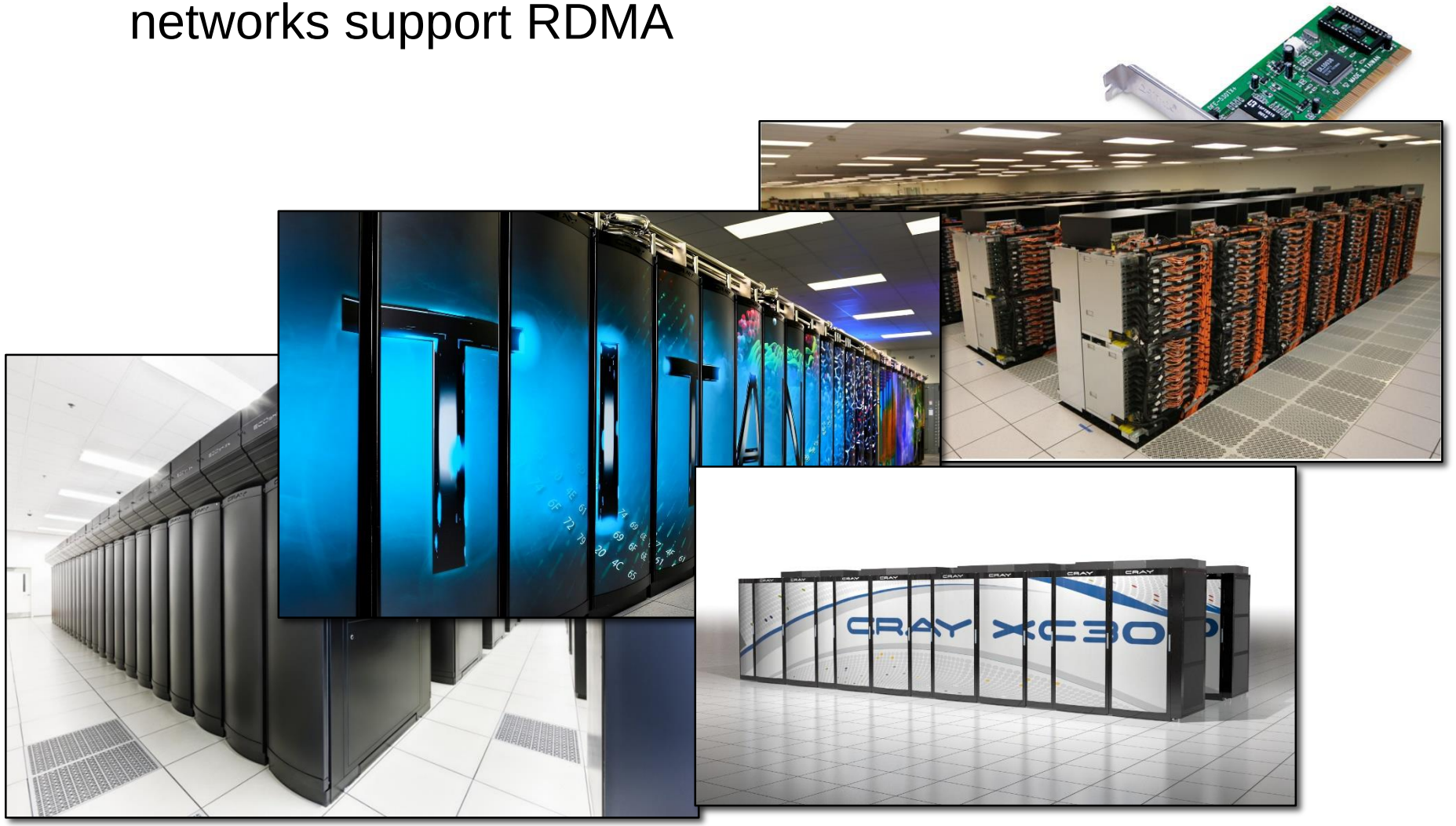


REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA

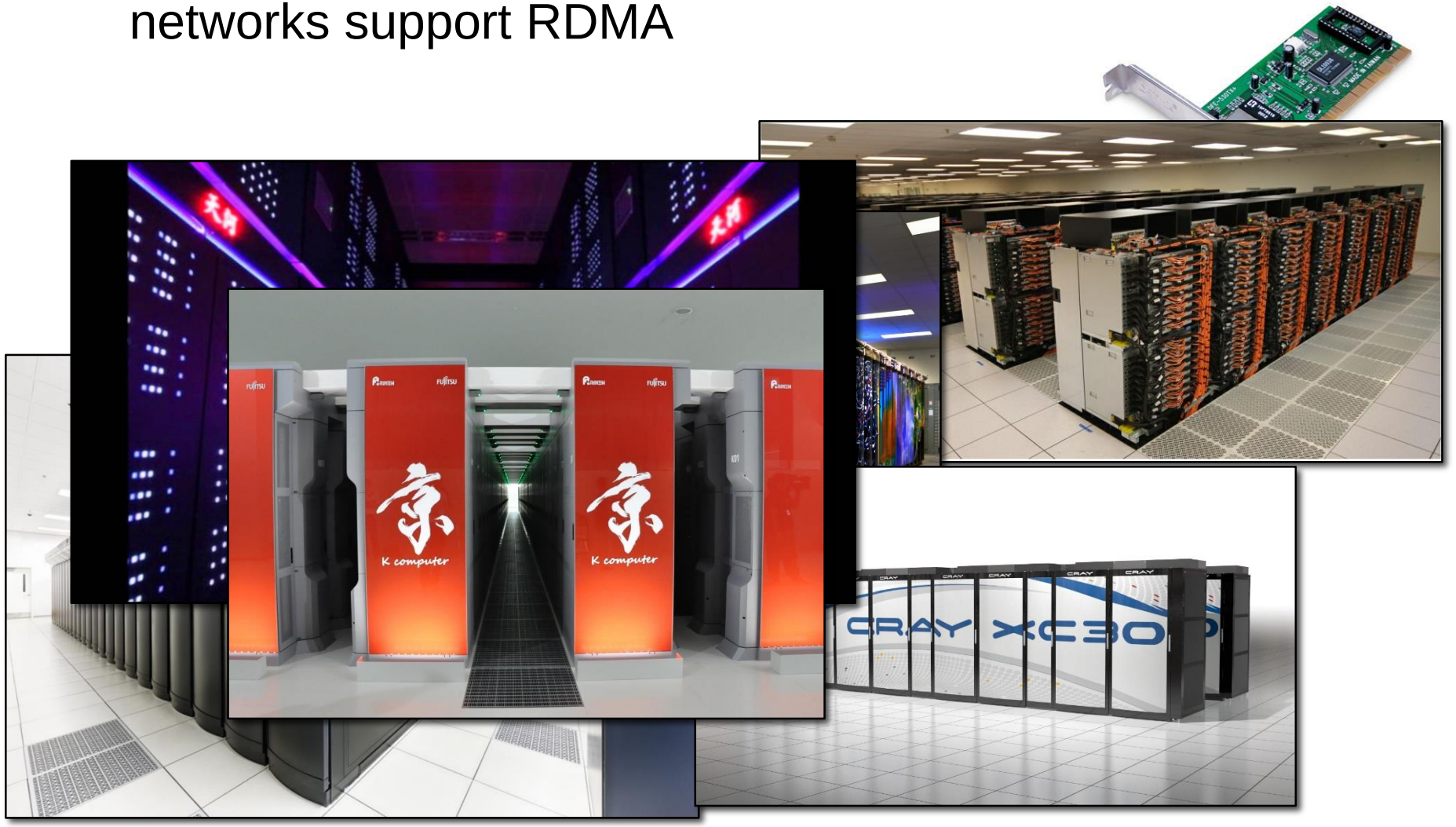
REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



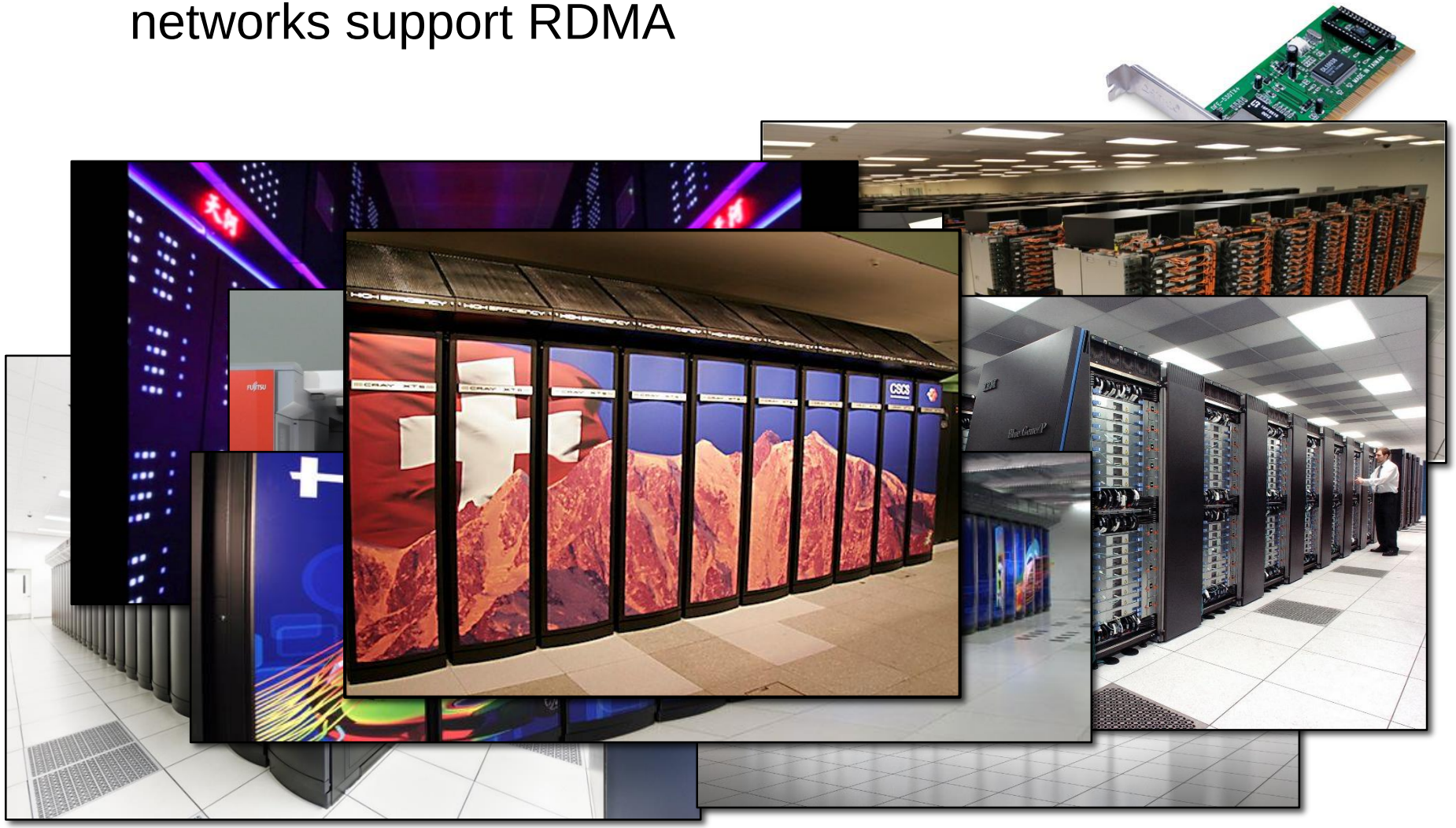
REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



Remote Memory Access Programming

- Implemented in hardware in NICs in the majority of HPC networks support RDMA

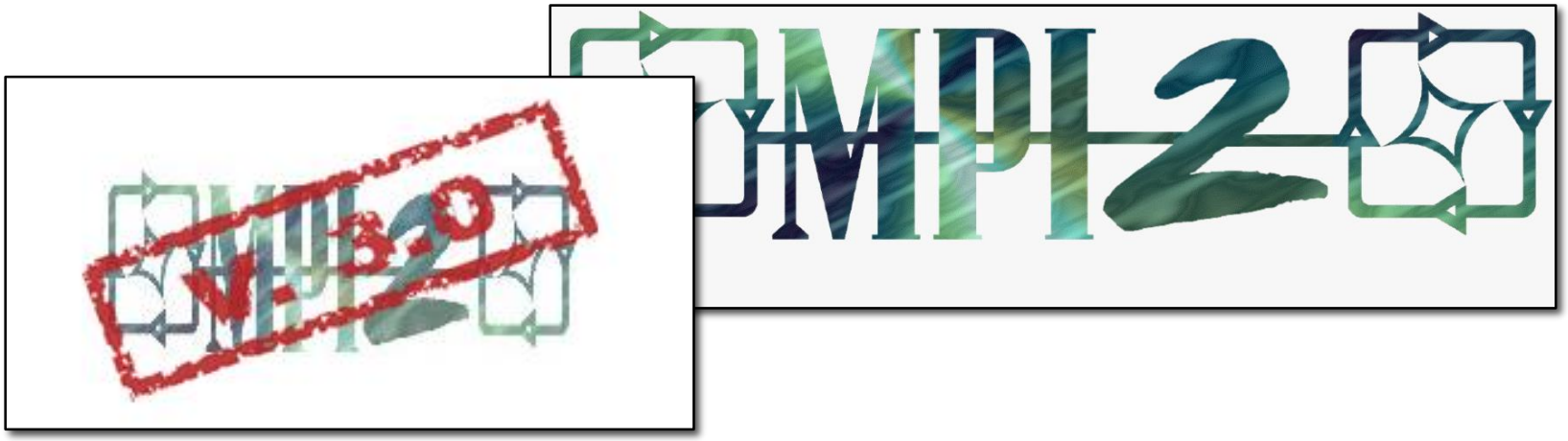


REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages

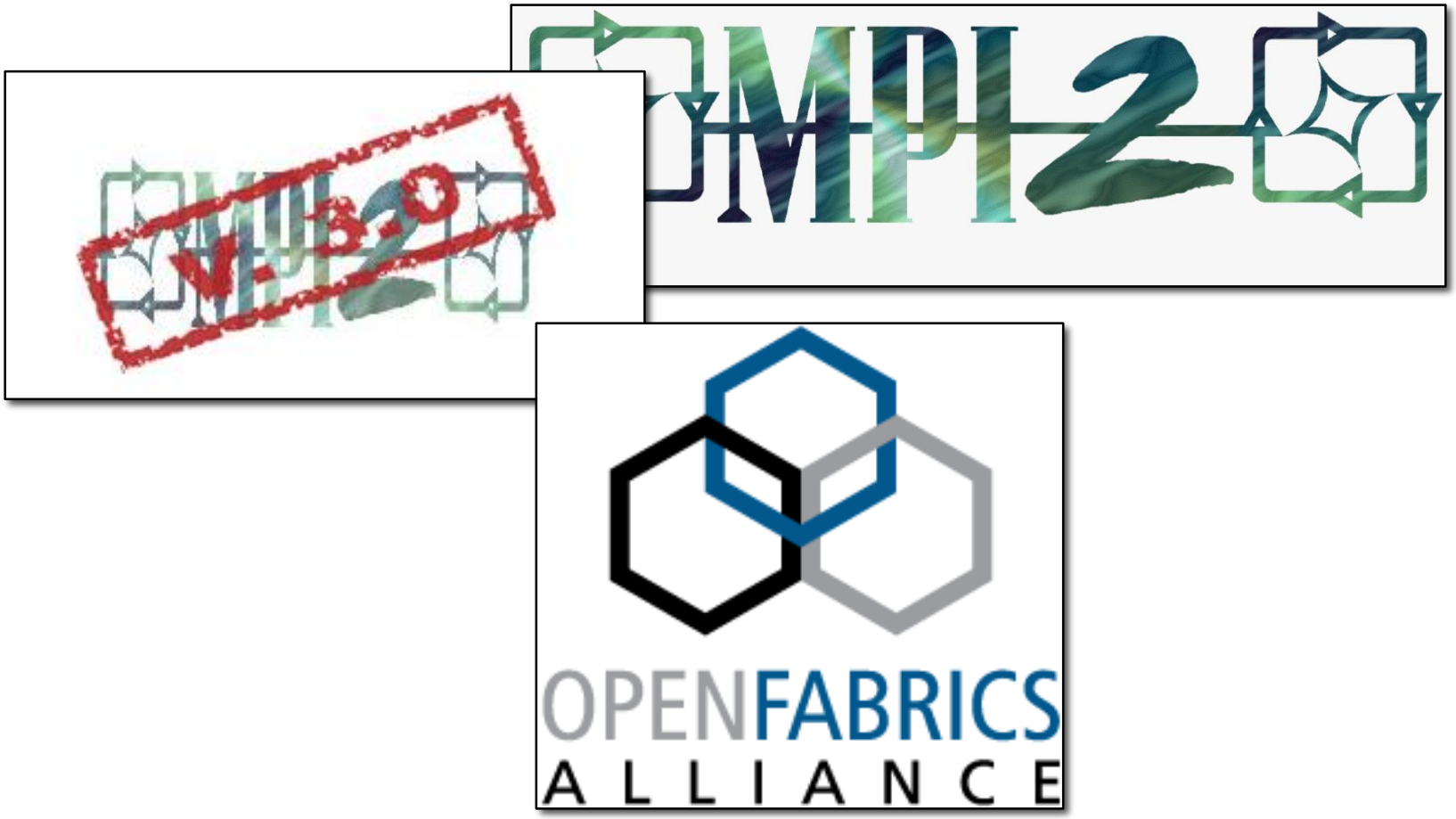
Remote Memory Access Programming

- Supported by many HPC libraries and languages



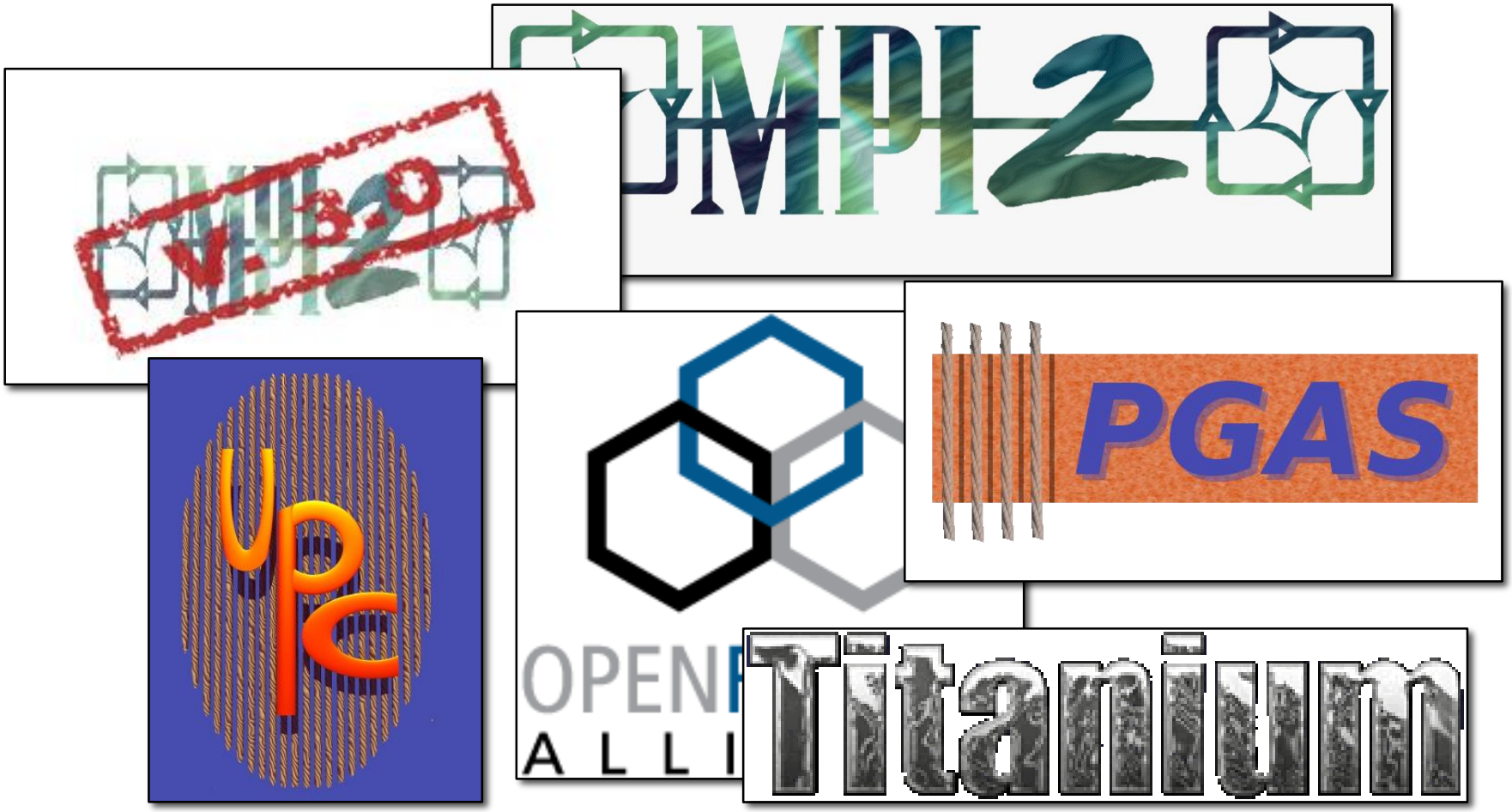
REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages



REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages



REMOTE MEMORY ACCESS PROGRAMMING

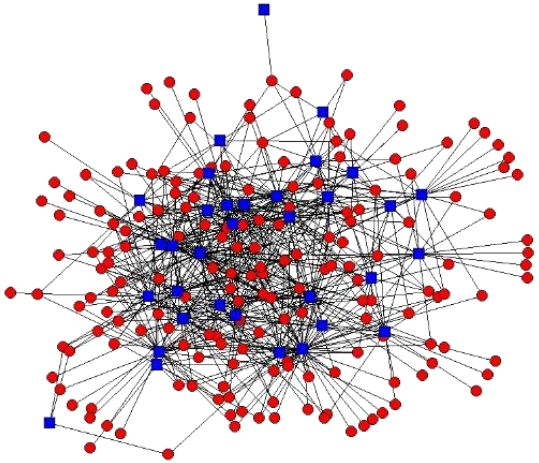
- Enables significant speedups over message passing in many types of applications, e.g.:

[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC**.** SPAA'12

REMOTE MEMORY ACCESS PROGRAMMING

- Enables significant speedups over message passing in many types of applications, e.g.:
  - Speedup of ~1.5 for communication patterns in graph analytics

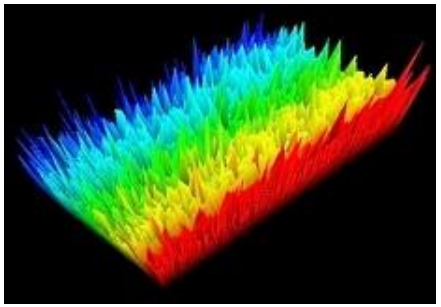


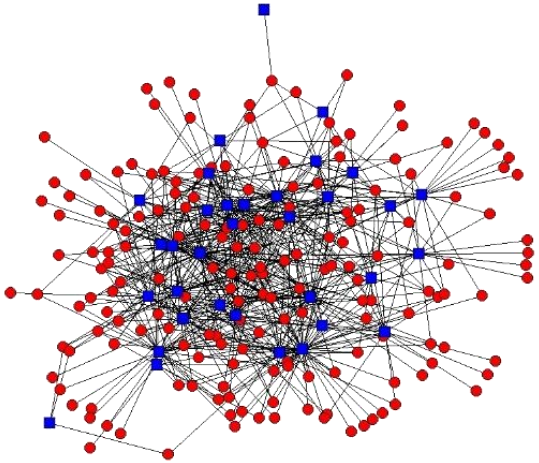
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

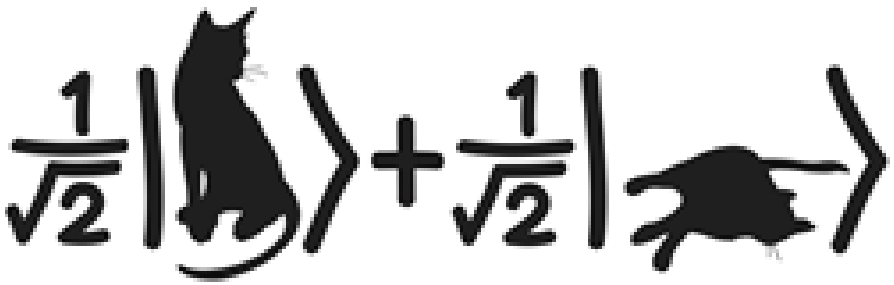
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC**.** SPAA'12

REMOTE MEMORY ACCESS PROGRAMMING

- Enables significant speedups over message passing in many types of applications, e.g.:
  - Speedup of ~1.5 for communication patterns in graph analytics
  - Speedup of ~1.4-2 in physics computations







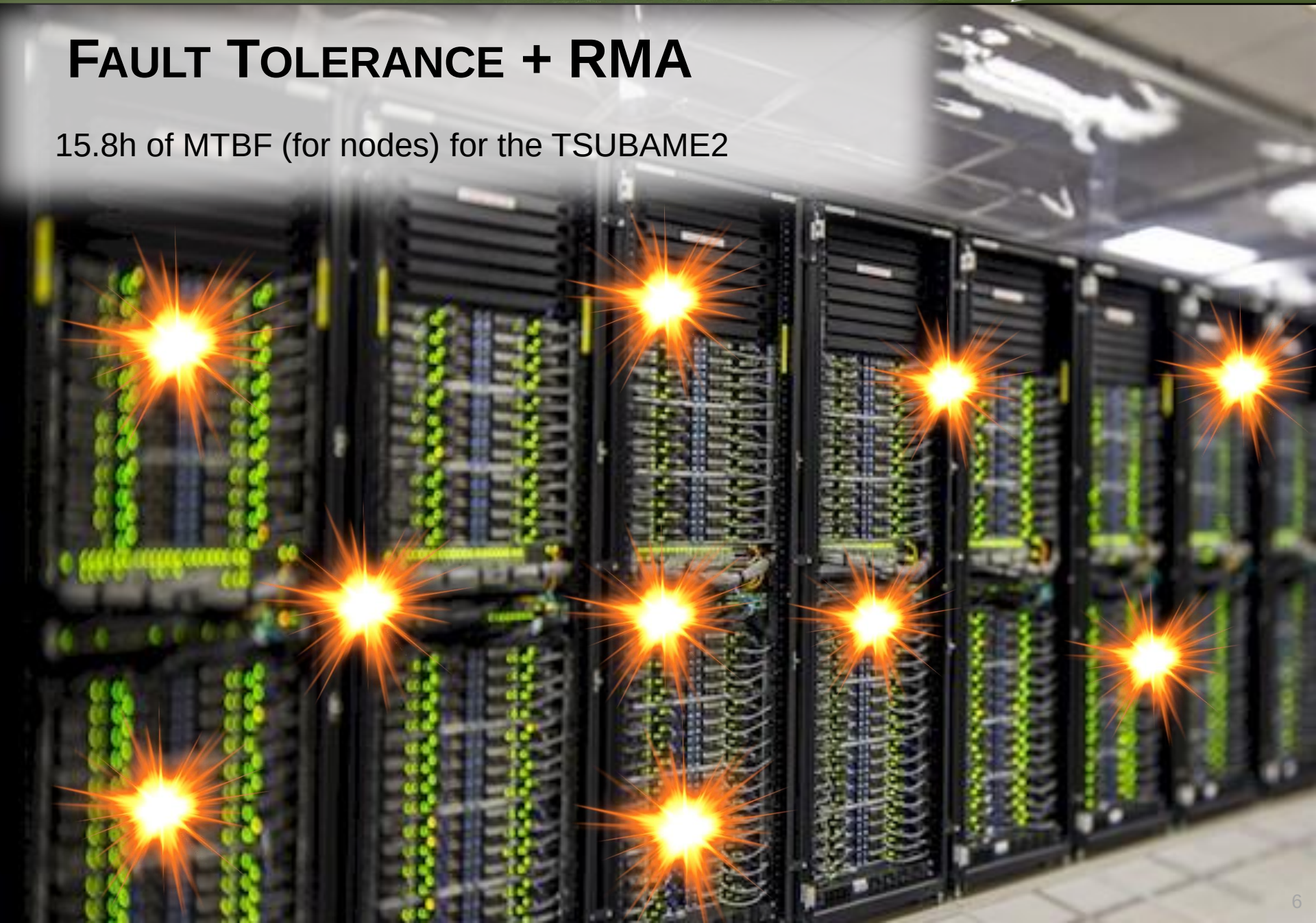
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC**.** SPAA'12

FAULT TOLERANCE + RMA

Fault Tolerance + RMA

15.8h of MTBF (for nodes) for the TSUBAME2



FAULT TOLERANCE + RMA

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

Coordinated Checkpointing (CC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

Coordinated Checkpointing (CC)

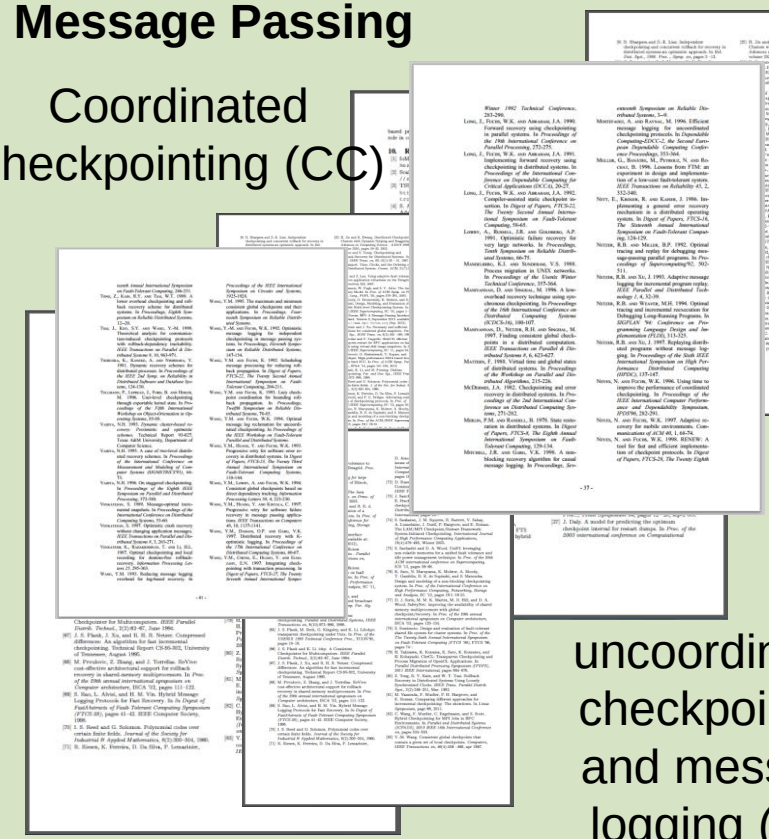
uncoordinated checkpointing and message logging (UC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

Coordinated Checkpointing (CC)



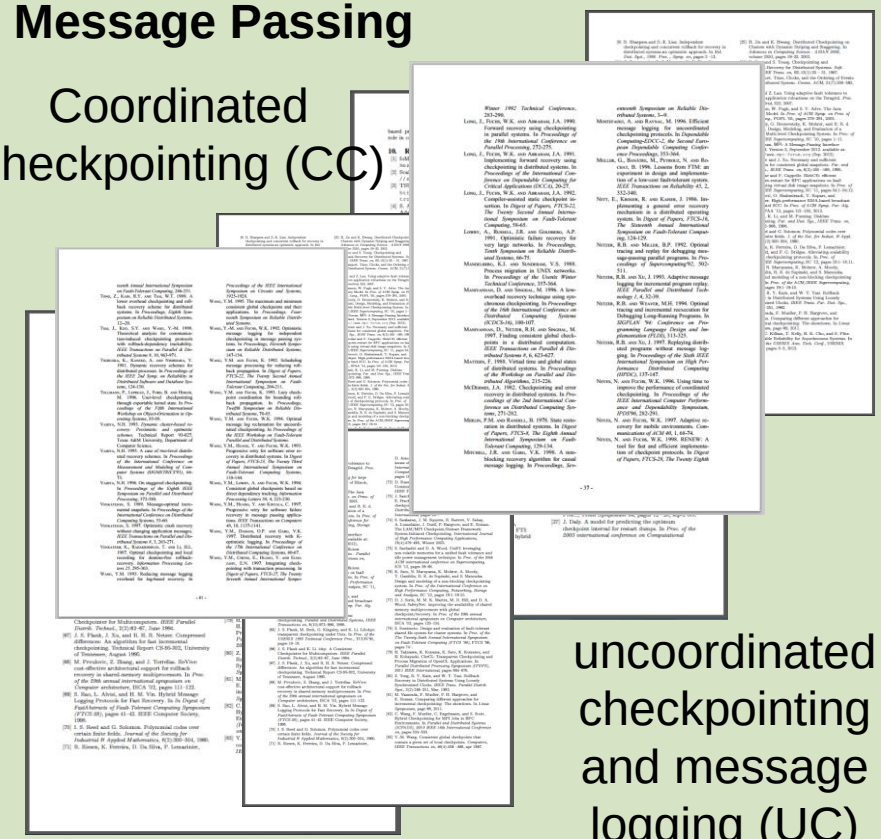
uncoordinated checkpointing and message logging (UC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

Message Passing

Coordinated Checkpointing (CC)



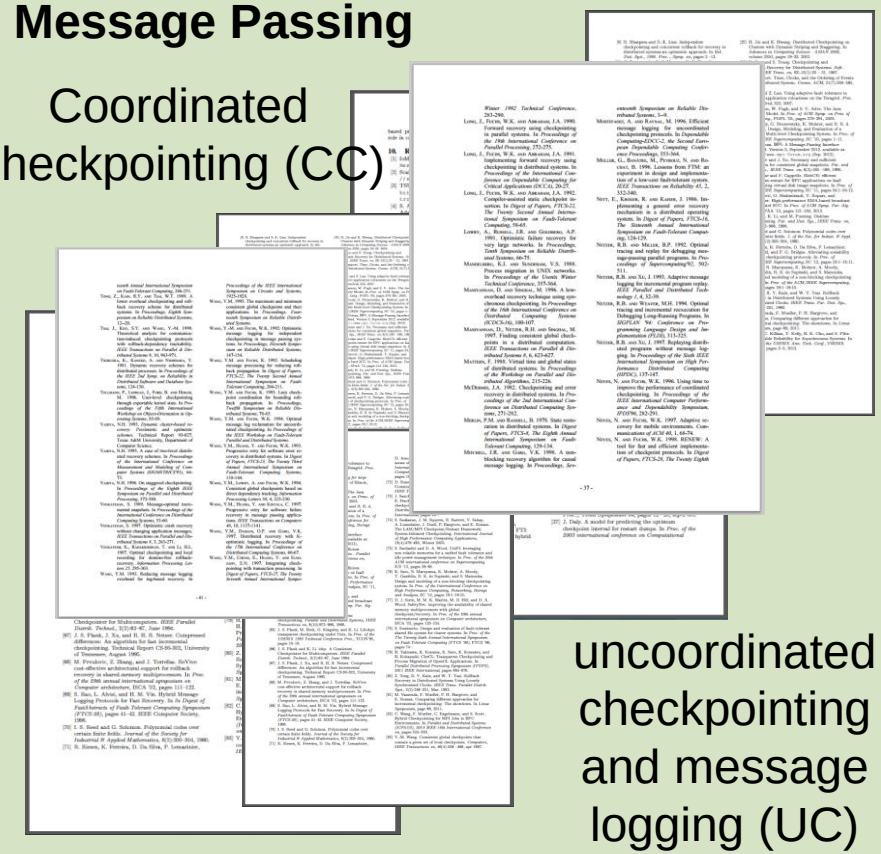
uncoordinated checkpointing and message logging (UC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

Message Passing

Coordinated Checkpointing (CC)



uncoordinated checkpointing and message logging (UC)

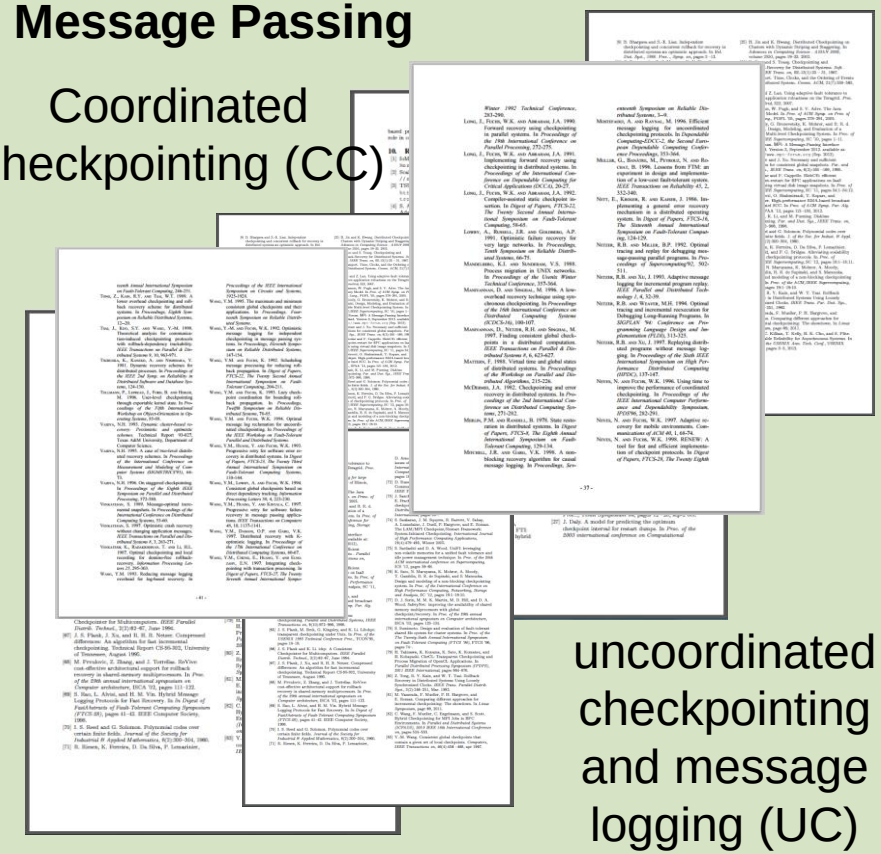
RMA

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

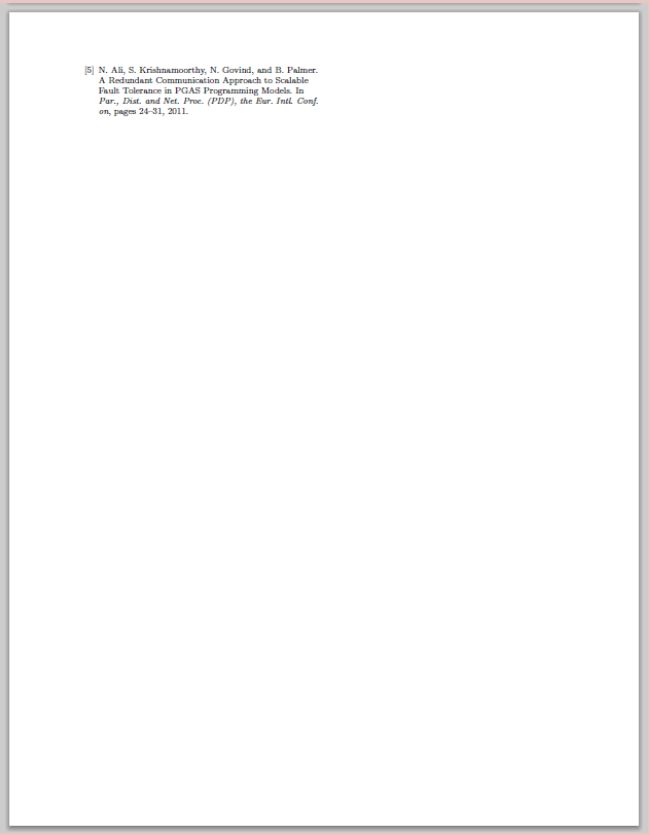
Message Passing

Coordinated Checkpointing (CC)



uncoordinated checkpointing and message logging (UC)

RMA

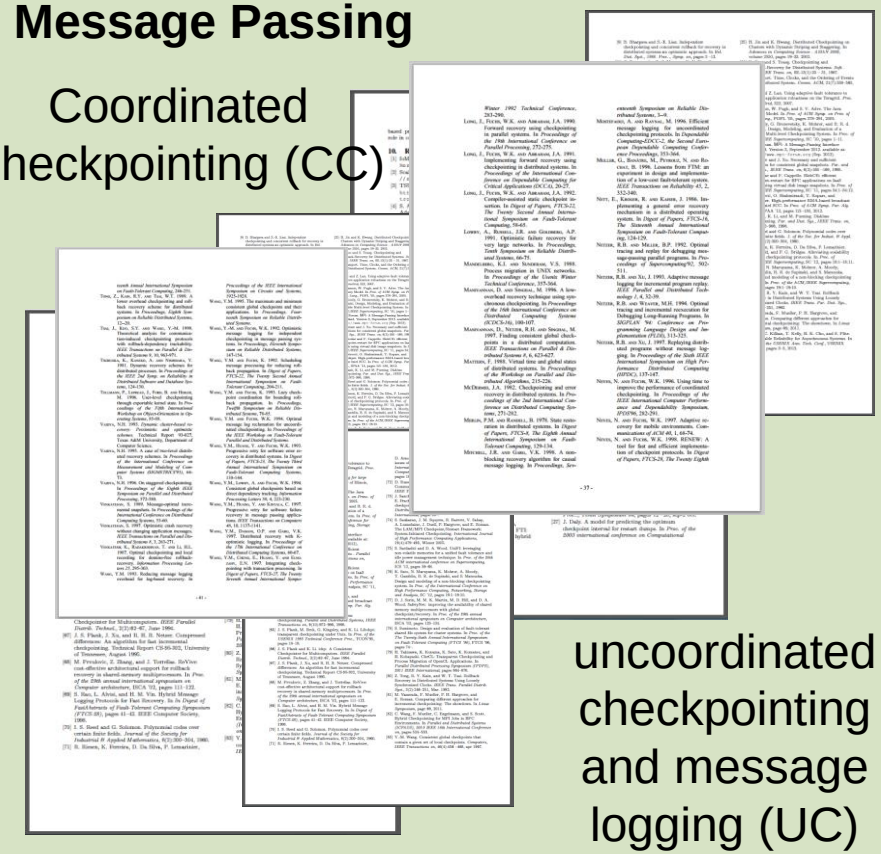


FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

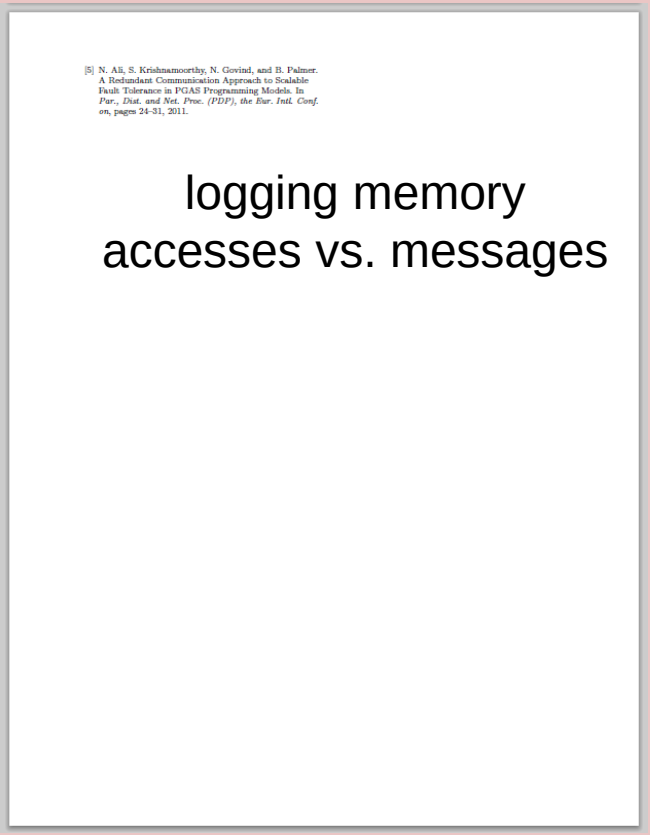
Message Passing

Coordinated Checkpointing (CC)



uncoordinated checkpointing and message logging (UC)

RMA



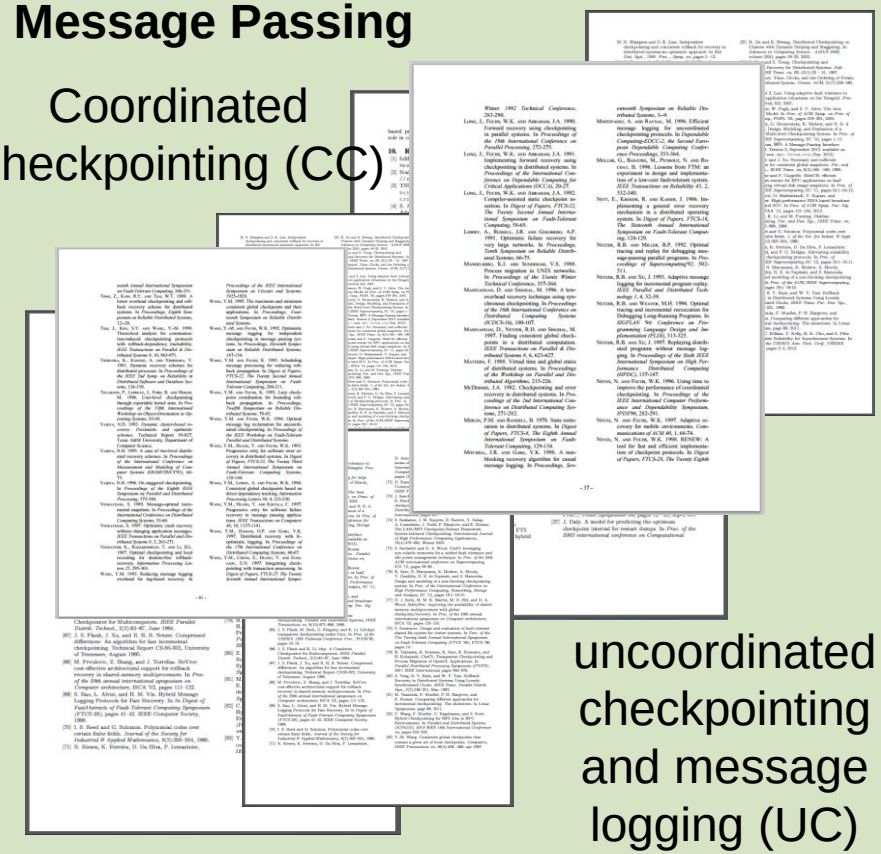
logging memory accesses vs. messages

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

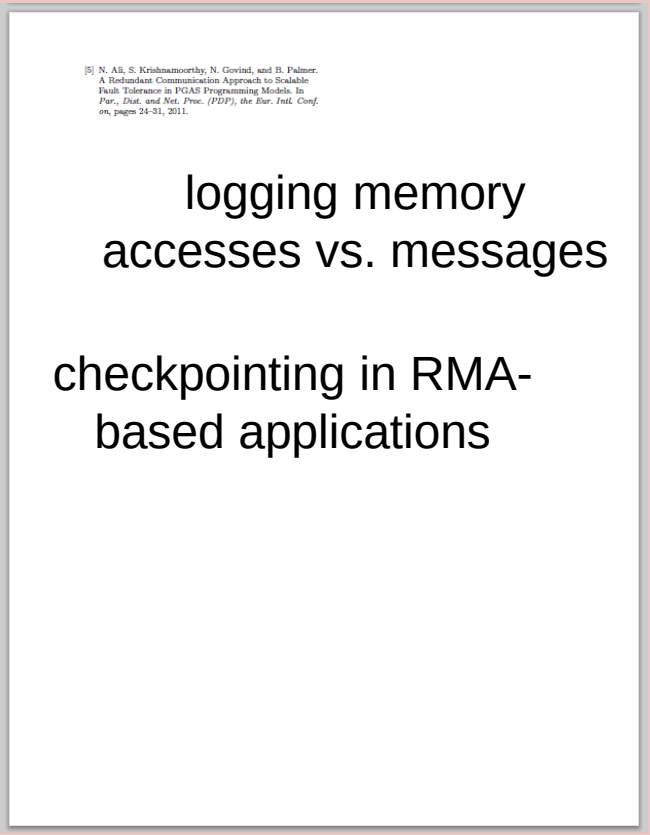
Message Passing

Coordinated Checkpointing (CC)



uncoordinated checkpointing and message logging (UC)

RMA



logging memory accesses vs. messages

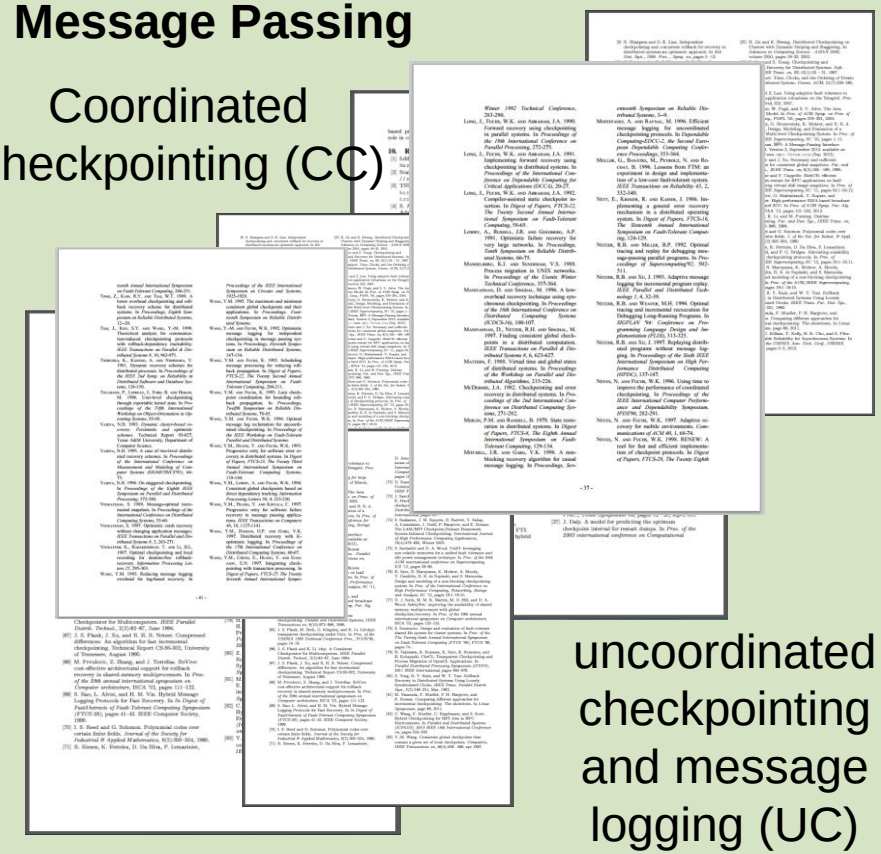
checkpointing in RMA-based applications

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

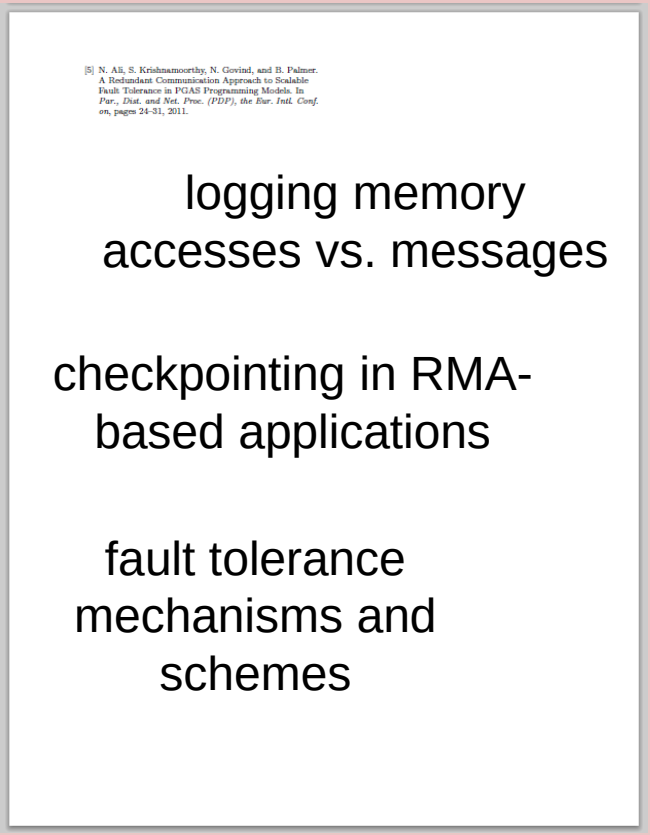
Message Passing

Coordinated Checkpointing (CC)



uncoordinated checkpointing and message logging (UC)

RMA



logging memory accesses vs. messages

checkpointing in RMA-based applications

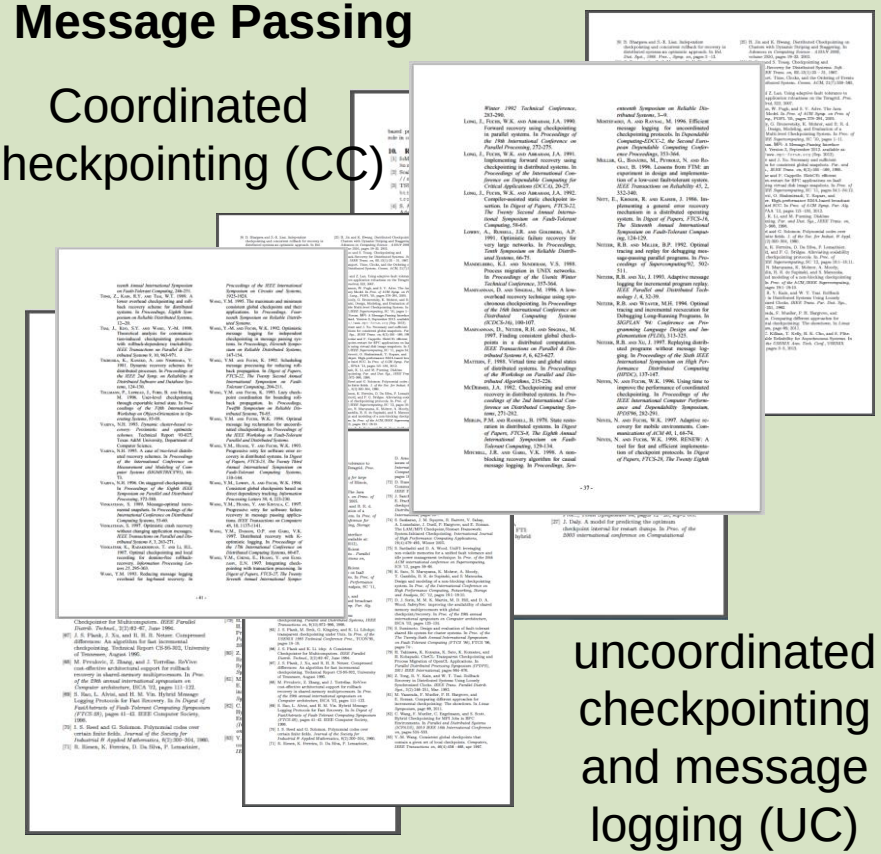
fault tolerance mechanisms and schemes

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

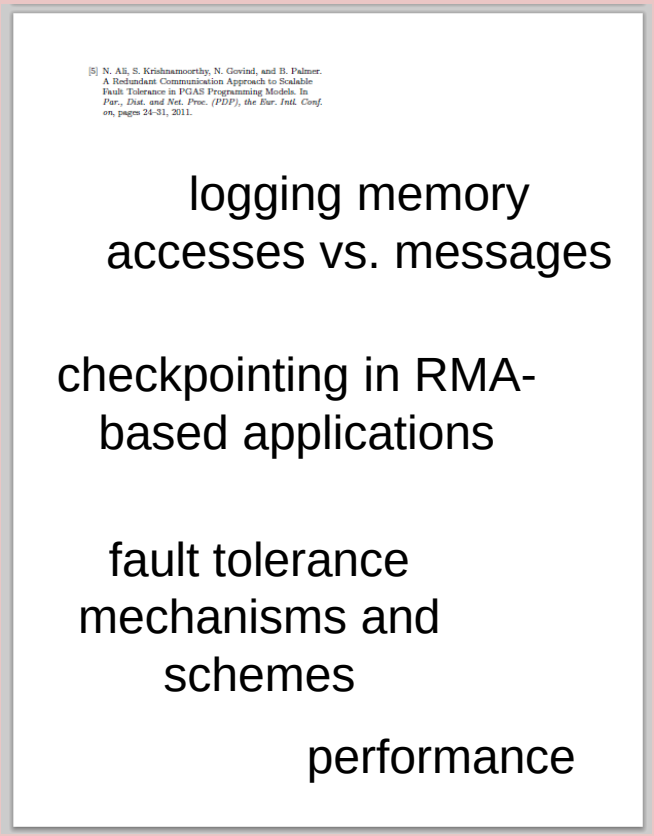
Message Passing

Coordinated Checkpointing (CC)



uncoordinated checkpointing and message logging (UC)

RMA



logging memory accesses vs. messages

checkpointing in RMA-based applications

fault tolerance mechanisms and schemes

performance

Overview of our Research

Overview of our Research

Generic model

Overview of our Research

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

Overview of our Research

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

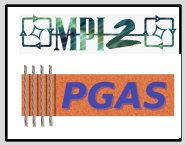
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

Overview of our Research

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

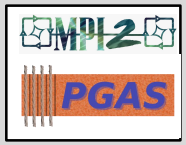


Overview of our Research

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



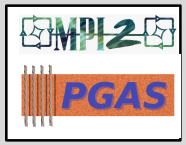
CC in RMA

Overview of our Research

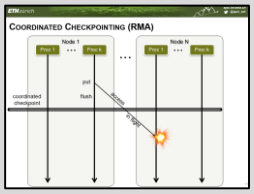
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

$\text{PUT}(p \rightrightarrows q)$
$\text{GET}(p \leftrightarrows q)$



CC in RMA



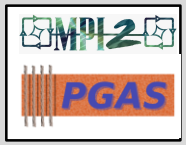
MP vs. RMA

Overview of our Research

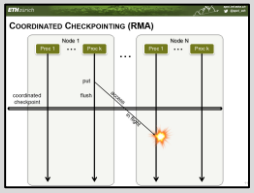
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

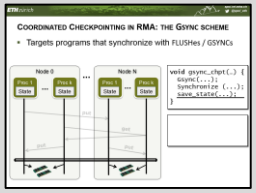
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



CC in RMA



MP vs. RMA



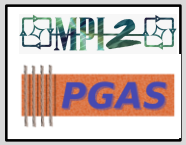
Schemes

Overview of our Research

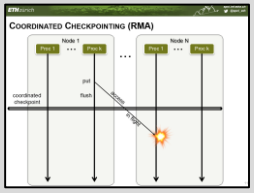
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

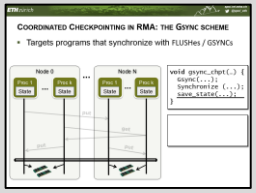
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



CC in RMA



MP vs. RMA



Schemes

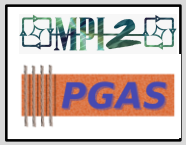
UC in RMA

Overview of our Research

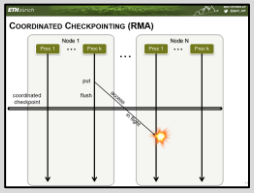
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

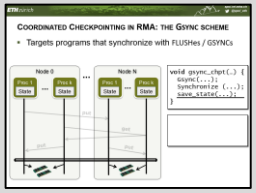
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



CC in RMA

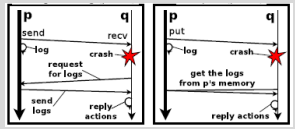


MP vs. RMA



Schemes

UC in RMA



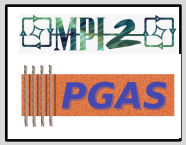
MP vs. RMA

Overview of our Research

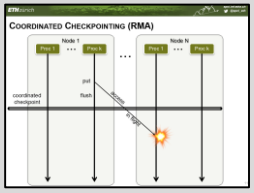
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

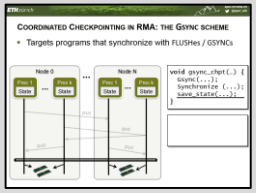
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



CC in RMA

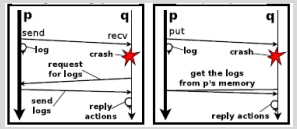


MP vs. RMA

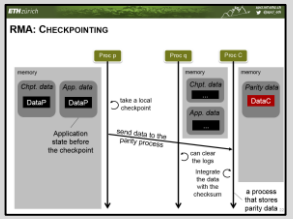


Schemes

UC in RMA



MP vs. RMA



Checkpointing schemes

Overview of our Research

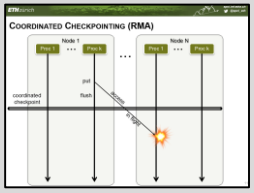
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

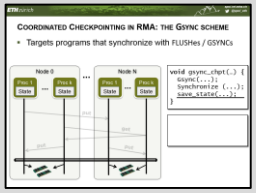
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

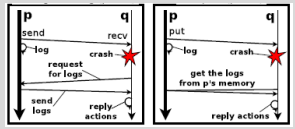


MP vs. RMA

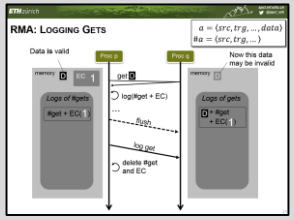


Schemes

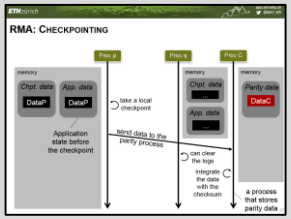
UC in RMA



MP vs. RMA



Logging accesses



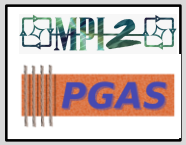
Checkpointing schemes

Overview of our Research

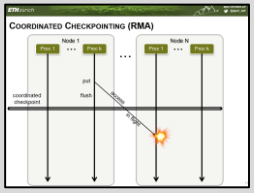
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

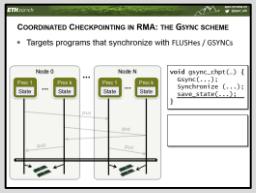
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



CC in RMA



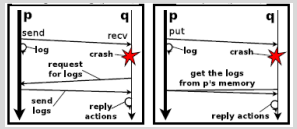
MP vs. RMA



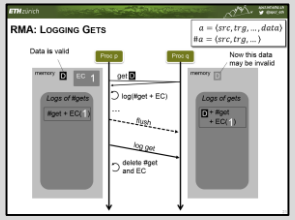
Schemes

Recovery in RMA

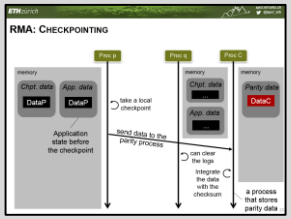
UC in RMA



MP vs. RMA



Logging accesses



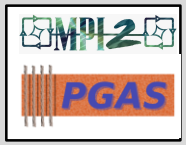
Checkpointing schemes

Overview of our Research

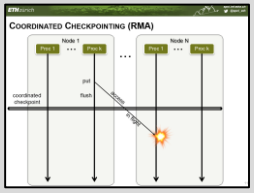
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

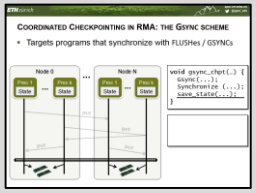
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$



CC in RMA

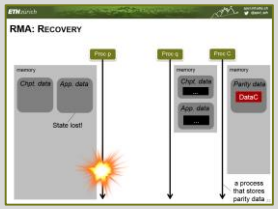


MP vs. RMA



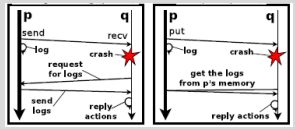
Schemes

Recovery in RMA

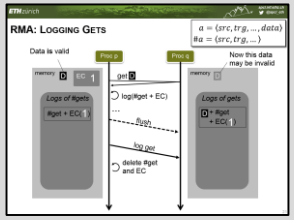


Basic scheme

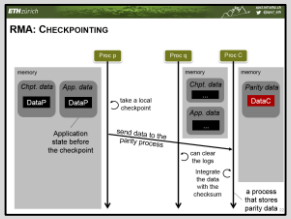
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

Overview of our Research

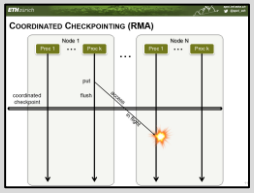
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

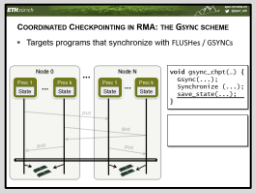
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

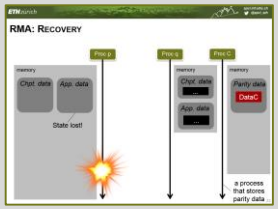


MP vs. RMA



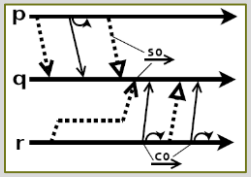
Schemes

Recovery in RMA

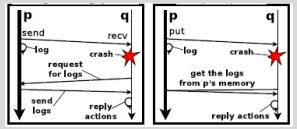


Basic scheme

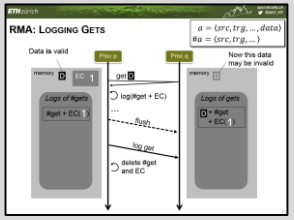
Extended RMA semantics



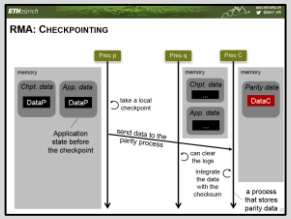
UC in RMA



MP vs. RMA



Logging accesses



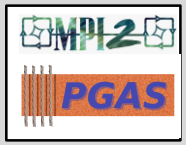
Checkpointing schemes

Overview of our Research

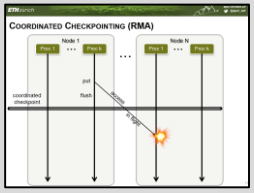
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

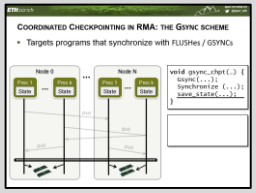
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$



CC in RMA

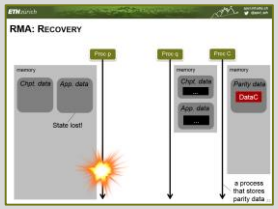


MP vs. RMA



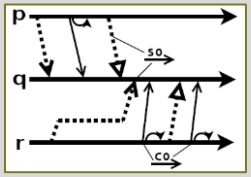
Schemes

Recovery in RMA

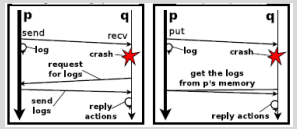


Basic scheme

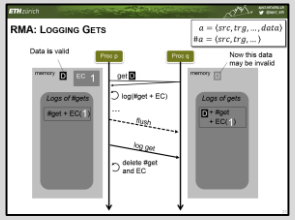
Extended RMA semantics



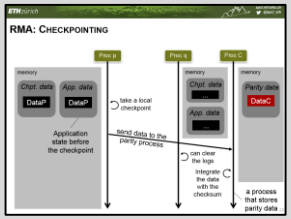
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

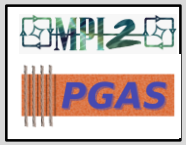
Topology-awareness

Overview of our Research

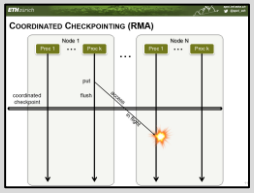
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

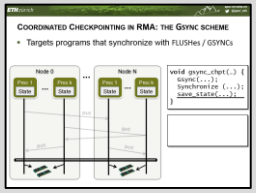
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$



CC in RMA

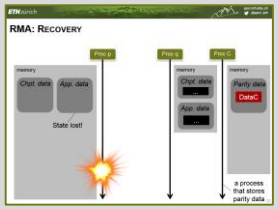


MP vs. RMA



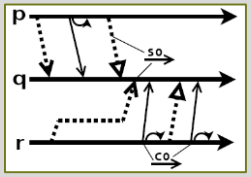
Schemes

Recovery in RMA

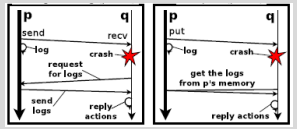


Basic scheme

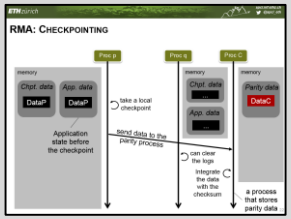
Extended RMA semantics



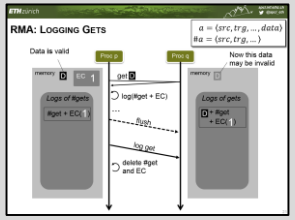
UC in RMA



MP vs. RMA



Checkpointing schemes



Logging accesses

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathscr{M} \rangle$$

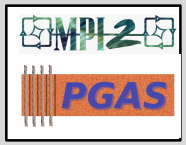
Model extensions

Overview of our Research

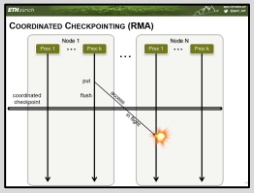
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

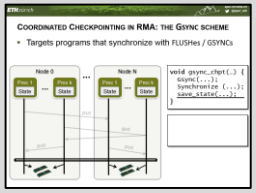
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$



CC in RMA

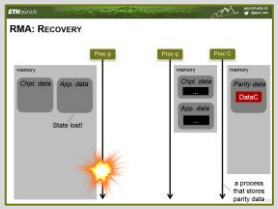


MP vs. RMA



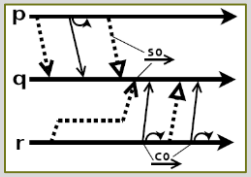
Schemes

Recovery in RMA

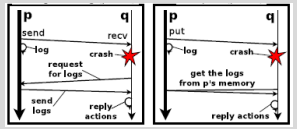


Basic scheme

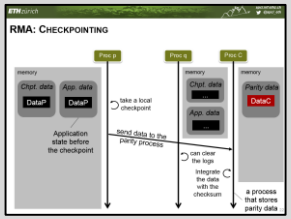
Extended RMA semantics



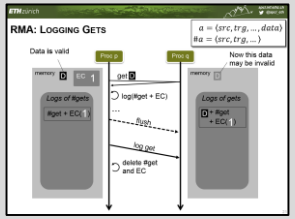
UC in RMA



MP vs. RMA



Checkpointing schemes

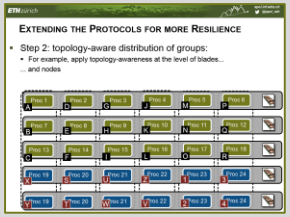


Logging accesses

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathscr{M} \rangle$$

Model extensions



Distribution of processes

Overview of our Research

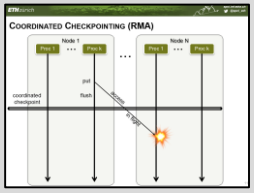
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

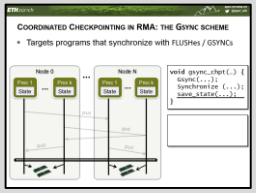
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA

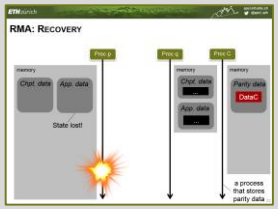


MP vs. RMA



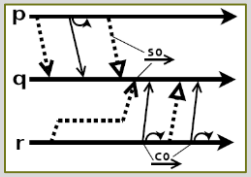
Schemes

Recovery in RMA

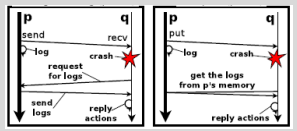


Basic scheme

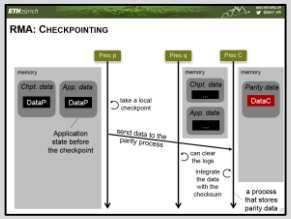
Extended RMA semantics



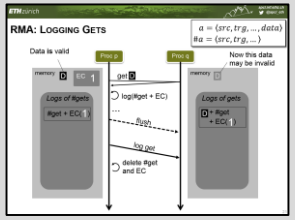
UC in RMA



MP vs. RMA



Checkpointing schemes

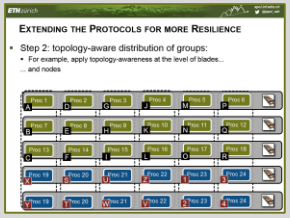


Logging accesses

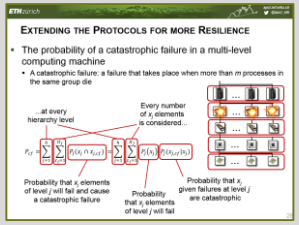
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

Overview of our Research

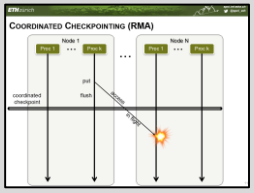
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

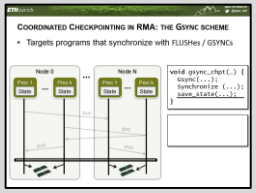
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

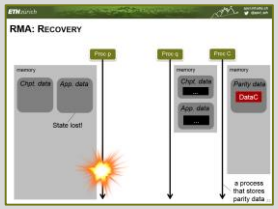


MP vs. RMA



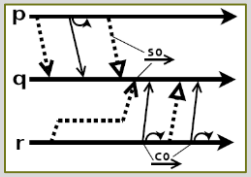
Schemes

Recovery in RMA



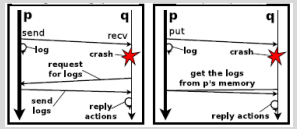
Basic scheme

Extended RMA semantics

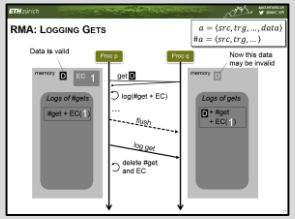


Holistic fault-tolerance library

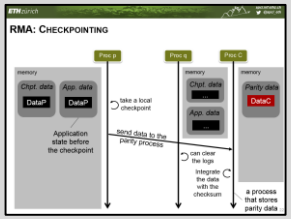
UC in RMA



MP vs. RMA



Logging accesses

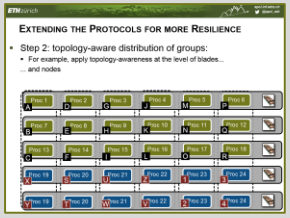


Checkpointing schemes

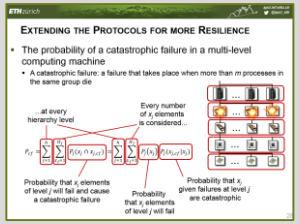
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



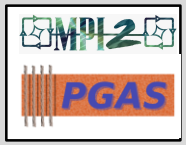
Decreasing failure prob.

Overview of our Research

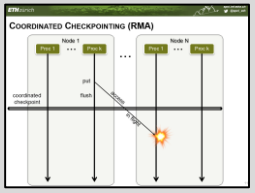
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

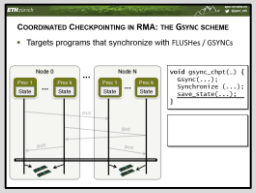
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$



CC in RMA

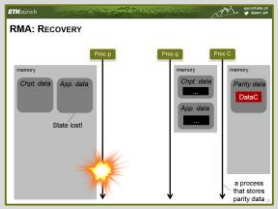


MP vs. RMA



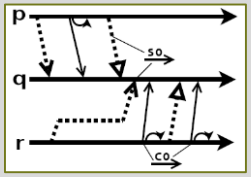
Schemes

Recovery in RMA

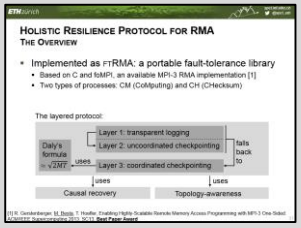


Basic scheme

Extended RMA semantics

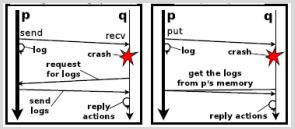


Holistic fault-tolerance library

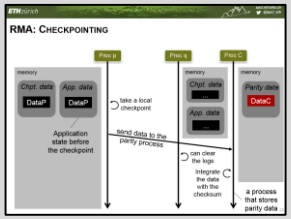


Design

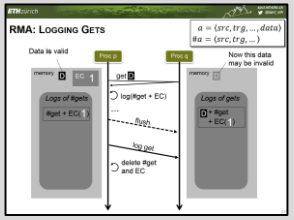
UC in RMA



MP vs. RMA



Checkpointing schemes

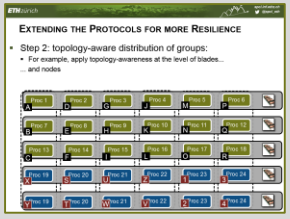


Logging accesses

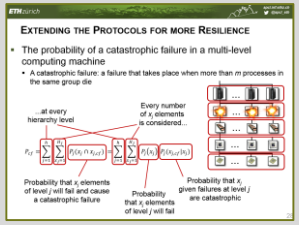
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

Overview of our Research

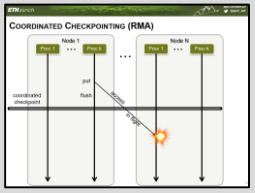
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

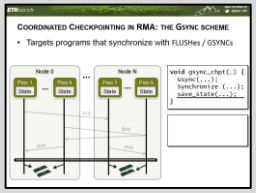
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

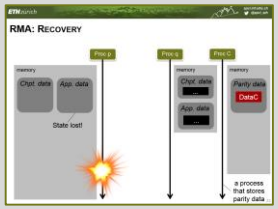


MP vs. RMA



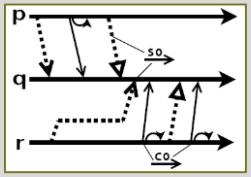
Schemes

Recovery in RMA

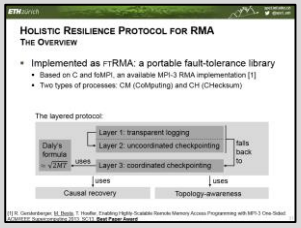


Basic scheme

Extended RMA semantics



Holistic fault-tolerance library

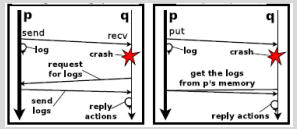


Design

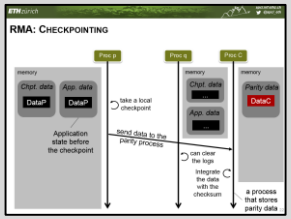
Checkpoints on demand

Optimizations

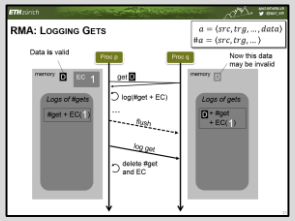
UC in RMA



MP vs. RMA



Checkpointing schemes

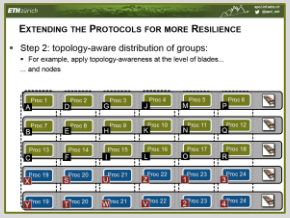


Logging accesses

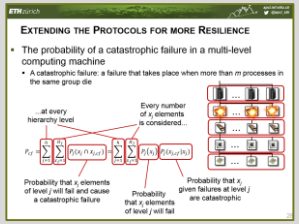
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

Overview of our Research

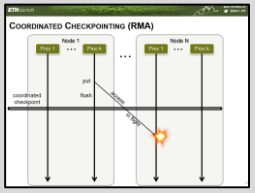
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

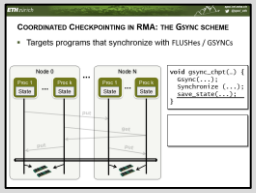
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

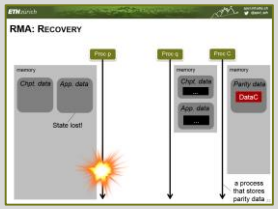


MP vs. RMA



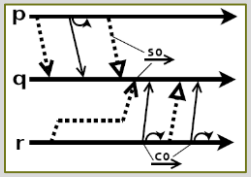
Schemes

Recovery in RMA

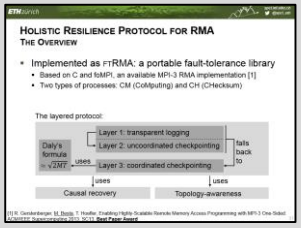


Basic scheme

Extended RMA semantics



Holistic fault-tolerance library



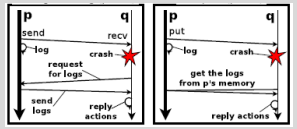
Design

Checkpoints on demand

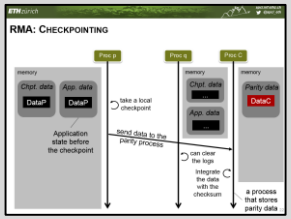
Optimizations

Performance

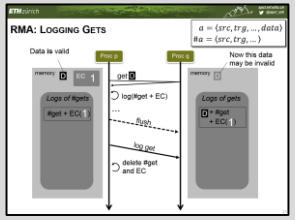
UC in RMA



MP vs. RMA



Checkpointing schemes

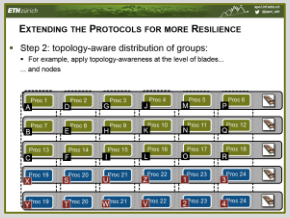


Logging accesses

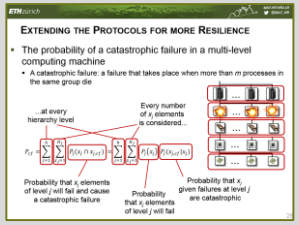
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

Overview of our Research

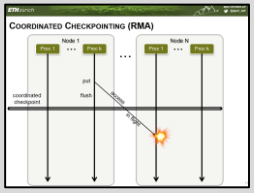
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

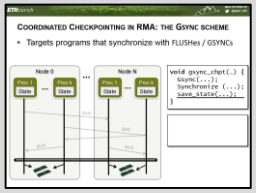
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

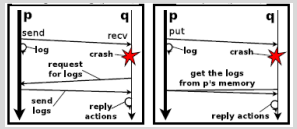


MP vs. RMA

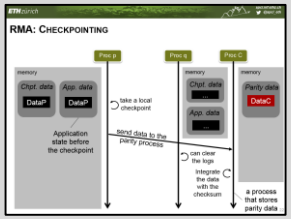


Schemes

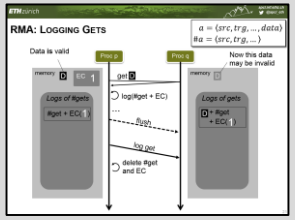
UC in RMA



MP vs. RMA

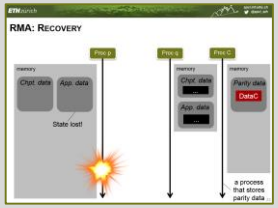


Checkpointing schemes



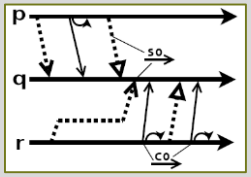
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics

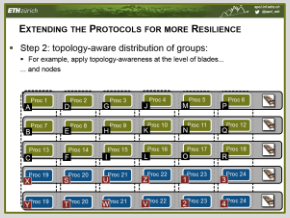


Proofs

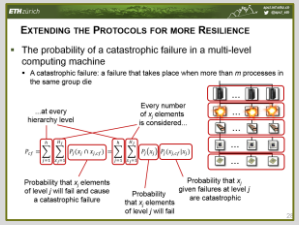
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

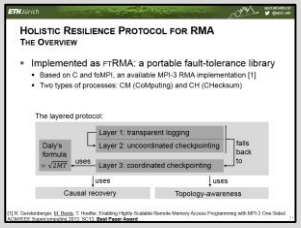


Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

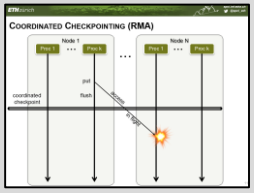
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

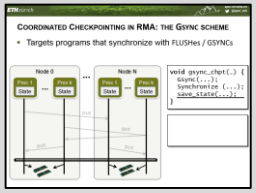
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA

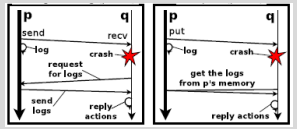


MP vs. RMA

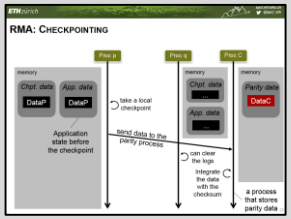


Schemes

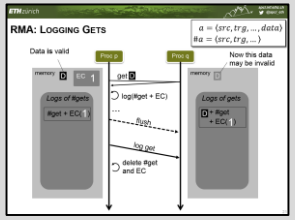
UC in RMA



MP vs. RMA

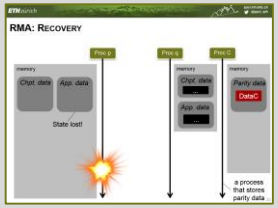


Checkpointing schemes



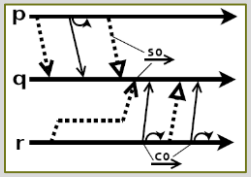
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

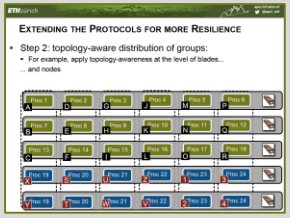
Theorem 4.2. *The recovery algorithm 2 preserves the* $\xrightarrow{cohb}$ *order (referred to as the* gsync order*).*

Deadlock freedom
Correct recovery

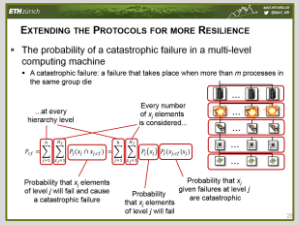
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

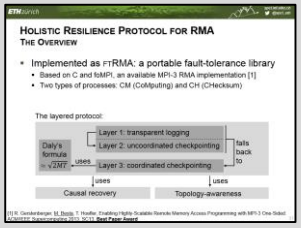


Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

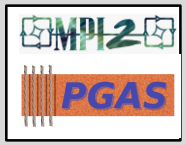
Performance

Overview of our Research

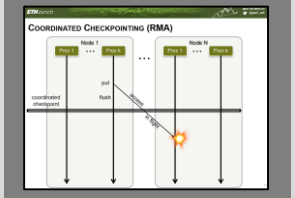
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

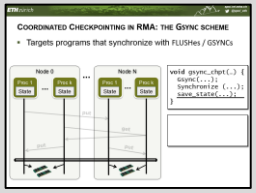
$$\mathrm{PUT}(p \rightrightarrows q)$$
$$\mathrm{GET}(p \leftrightarrows q)$$



CC in RMA

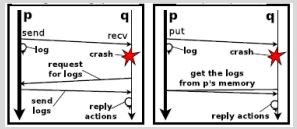


MP vs. RMA

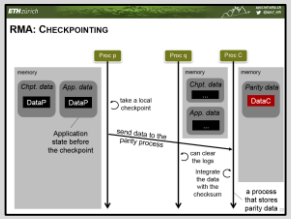


Schemes

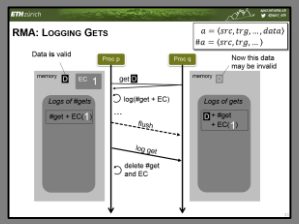
UC in RMA



MP vs. RMA

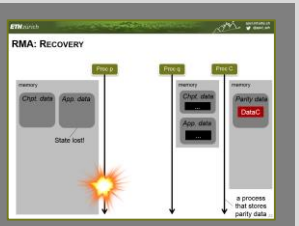


Checkpointing schemes



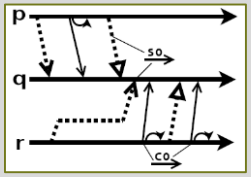
Logging accesses

Recovery in RMA

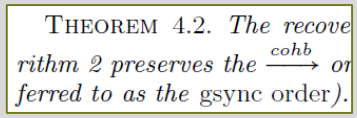


Basic scheme

Extended RMA semantics



Proofs



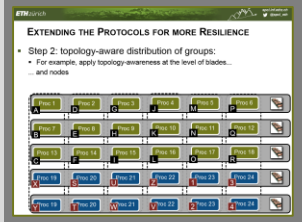
THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

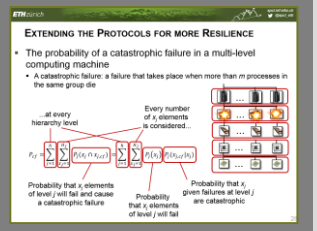
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

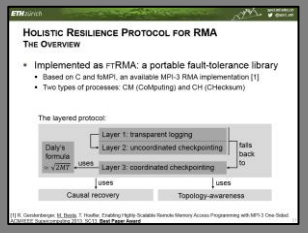


Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

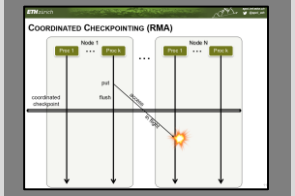
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

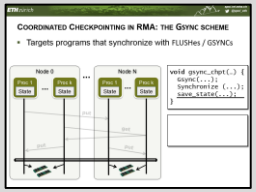
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA



MP vs. RMA



Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

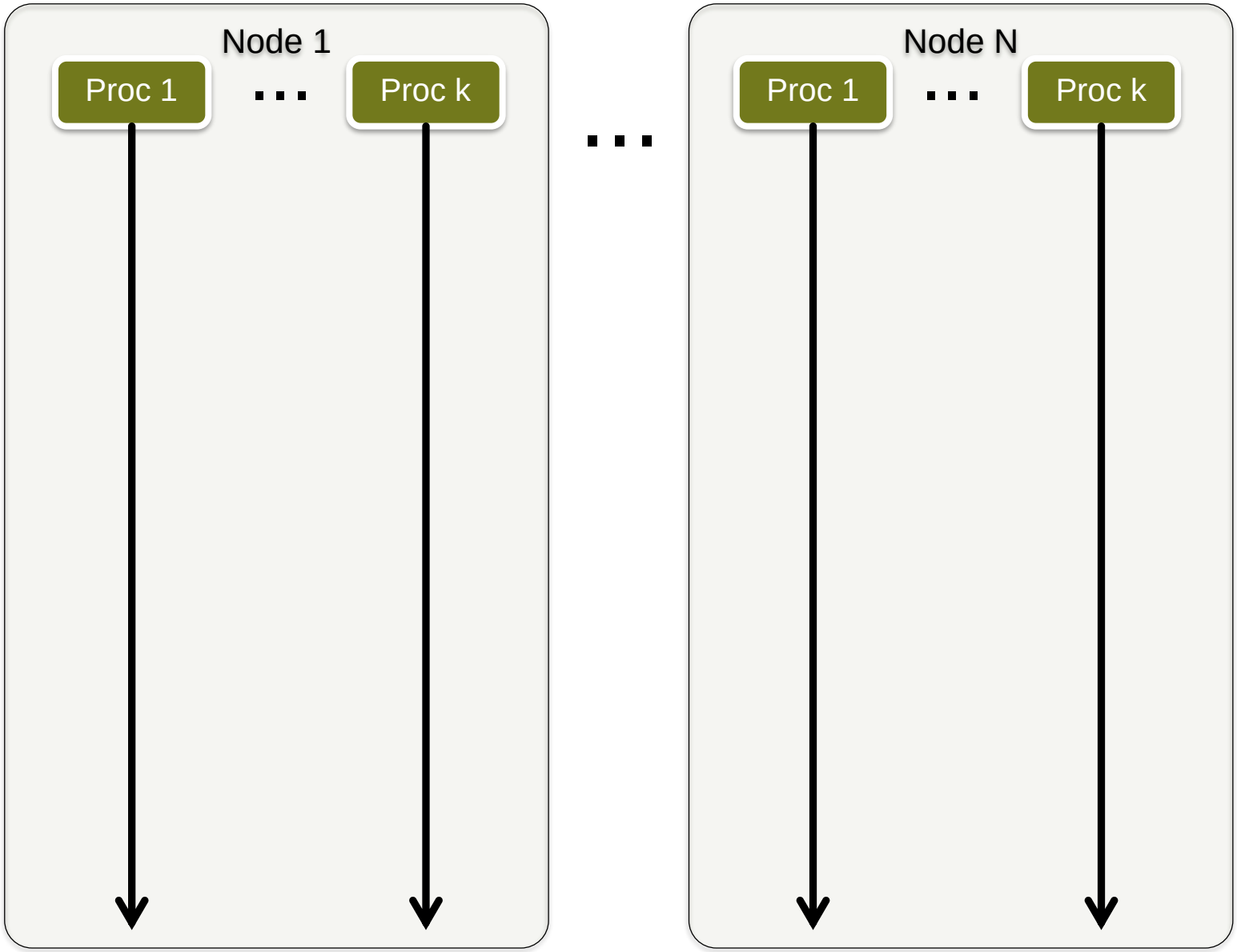
Performance

Distribution of processes

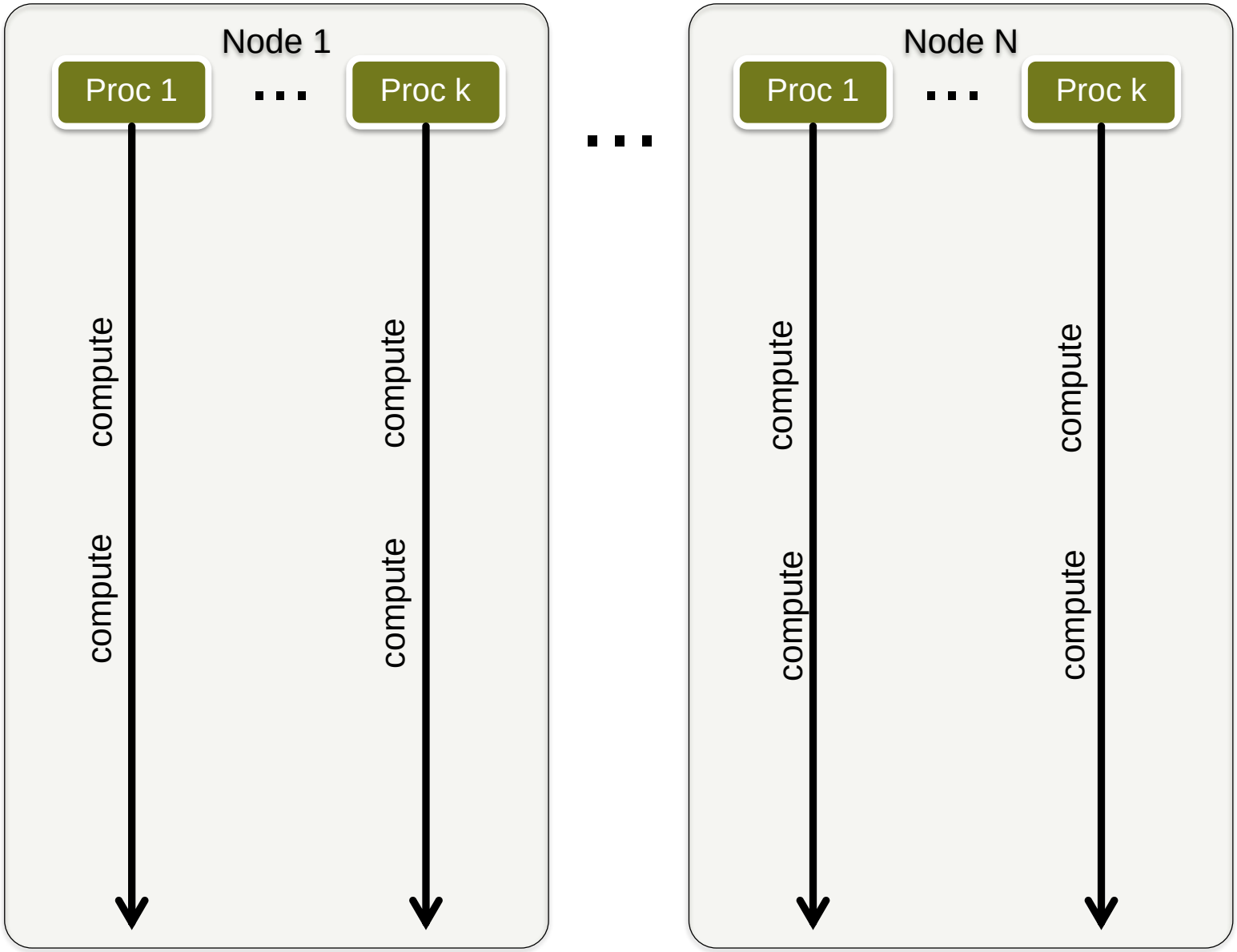
Decreasing failure prob.

COORDINATED CHECKPOINTING (MP)

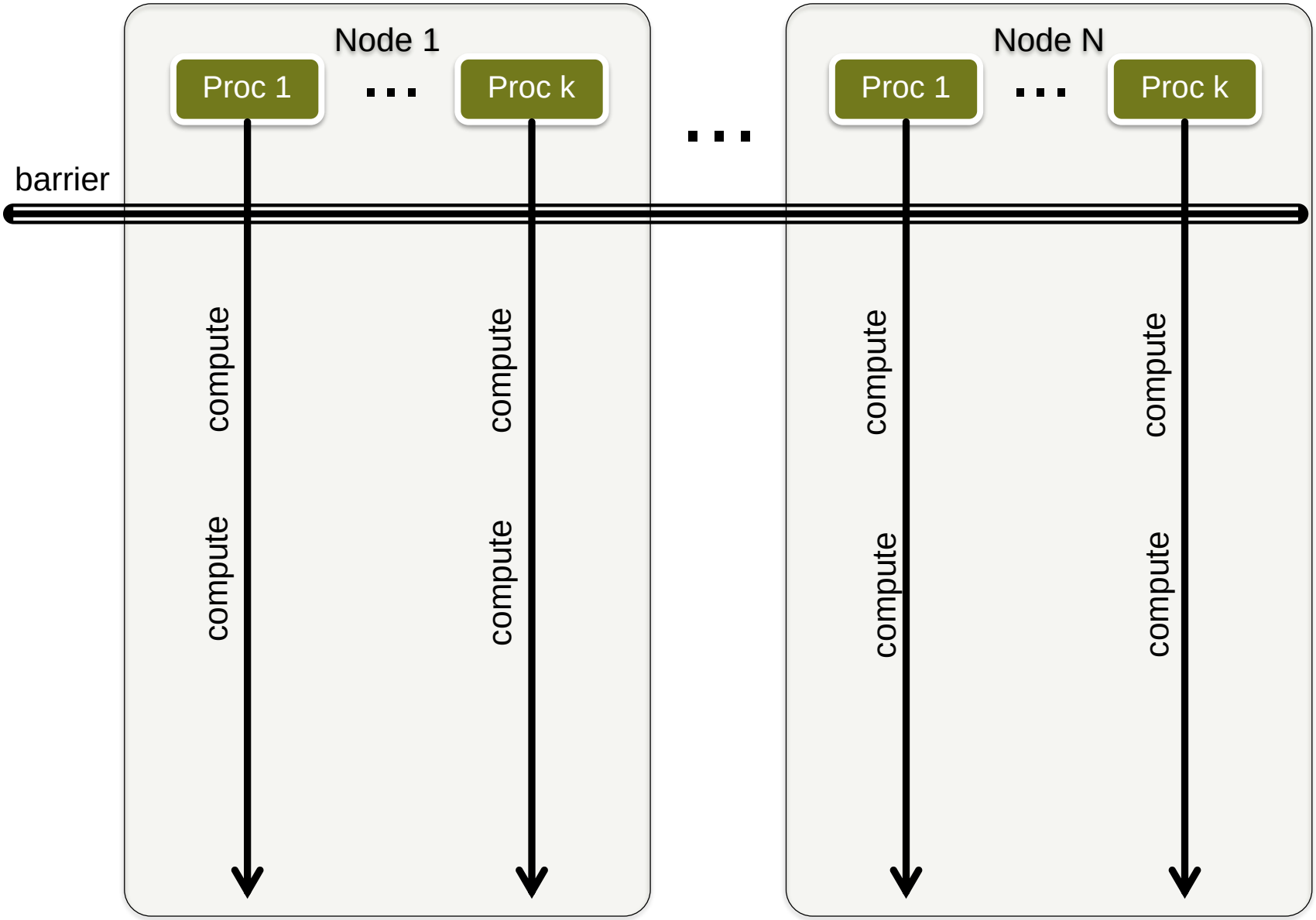
COORDINATED CHECKPOINTING (MP)



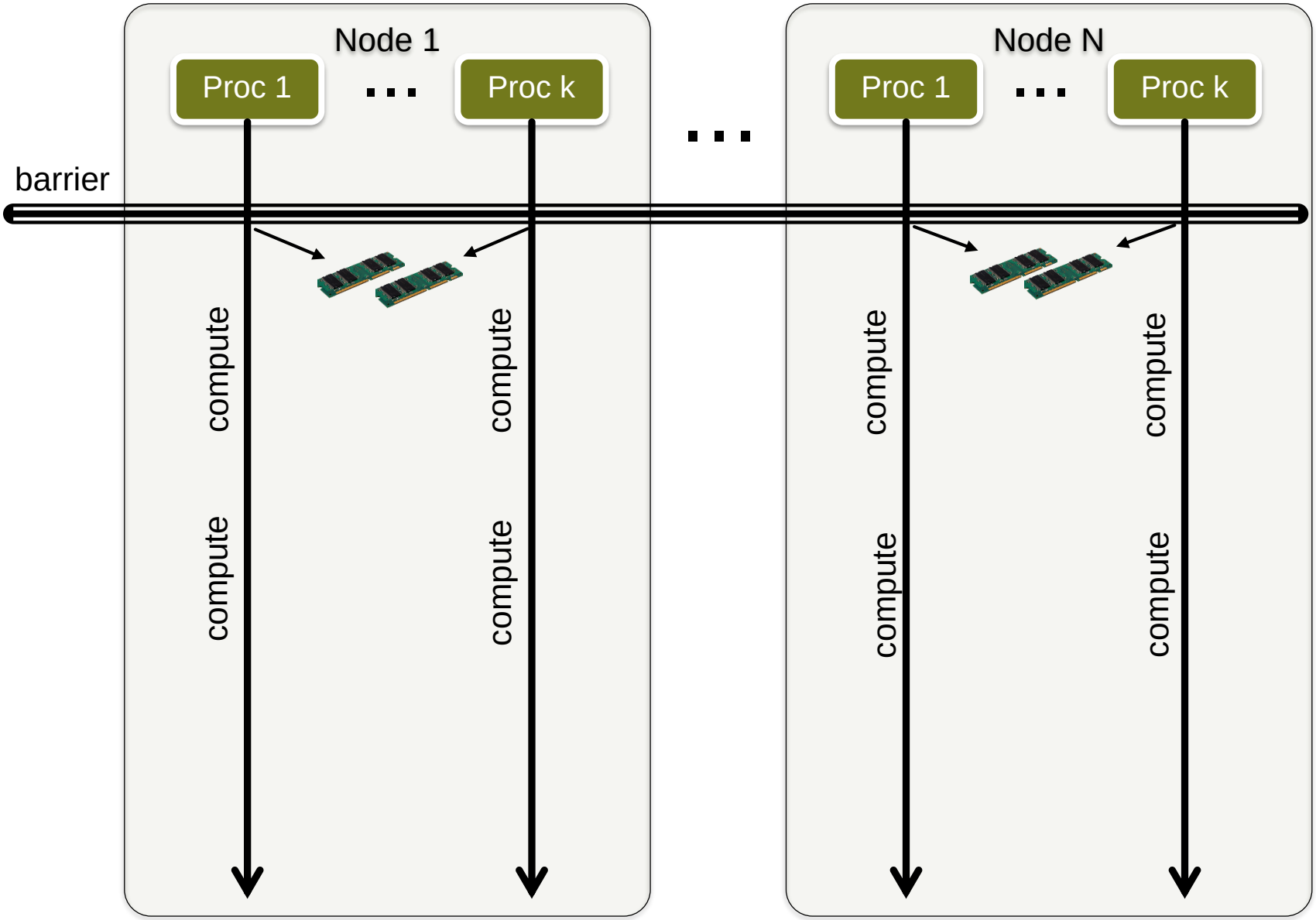
COORDINATED CHECKPOINTING (MP)



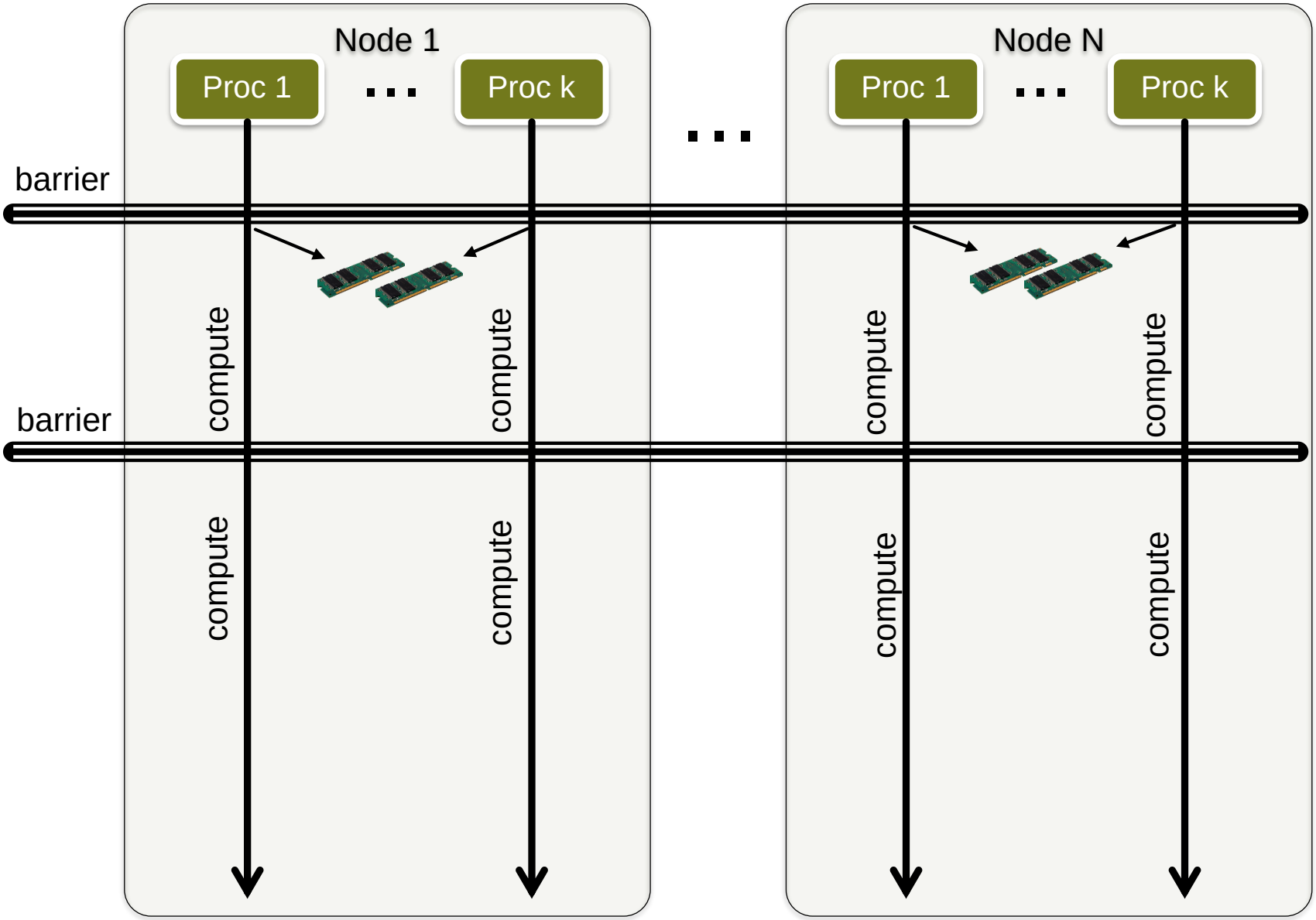
COORDINATED CHECKPOINTING (MP)



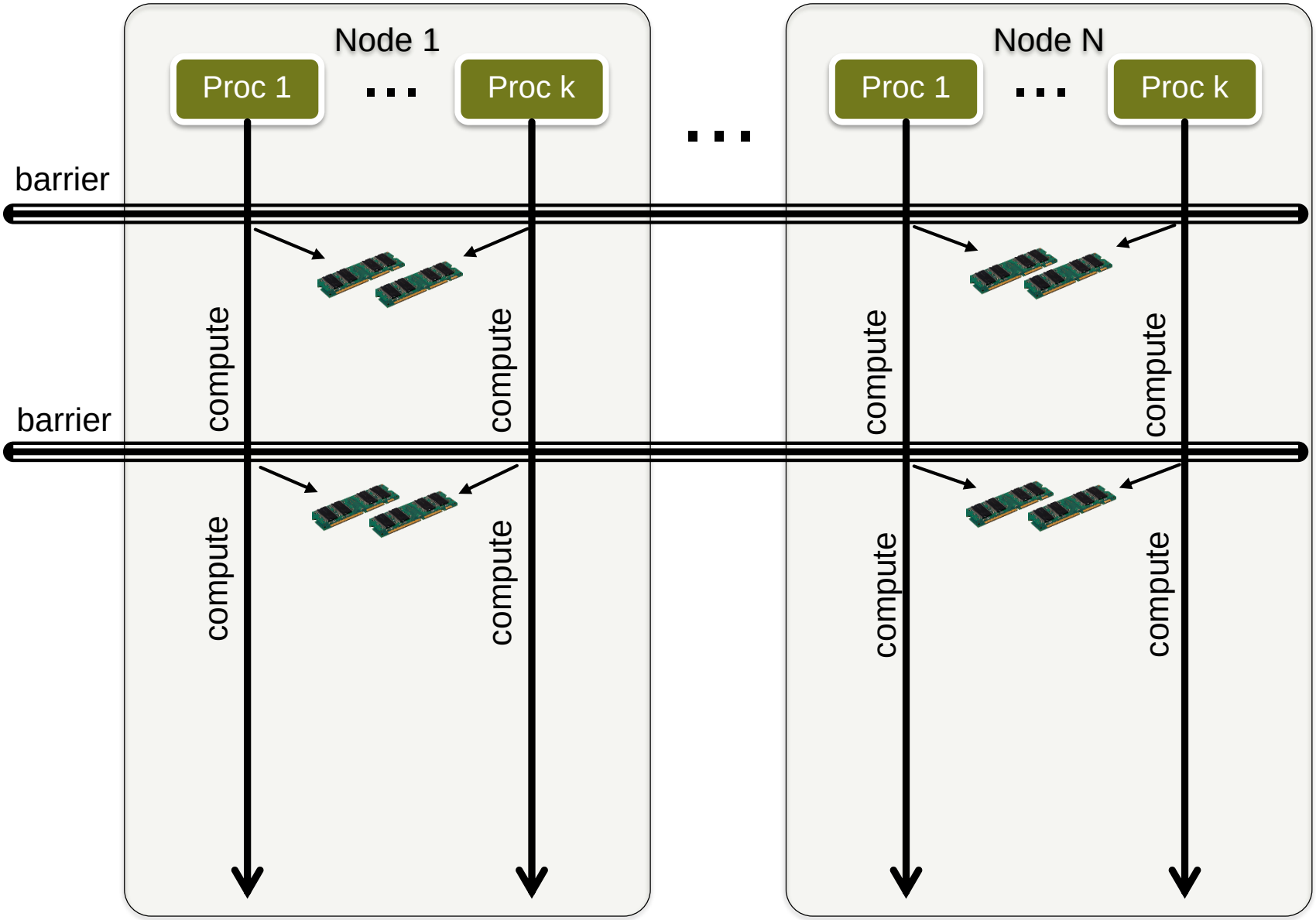
COORDINATED CHECKPOINTING (MP)



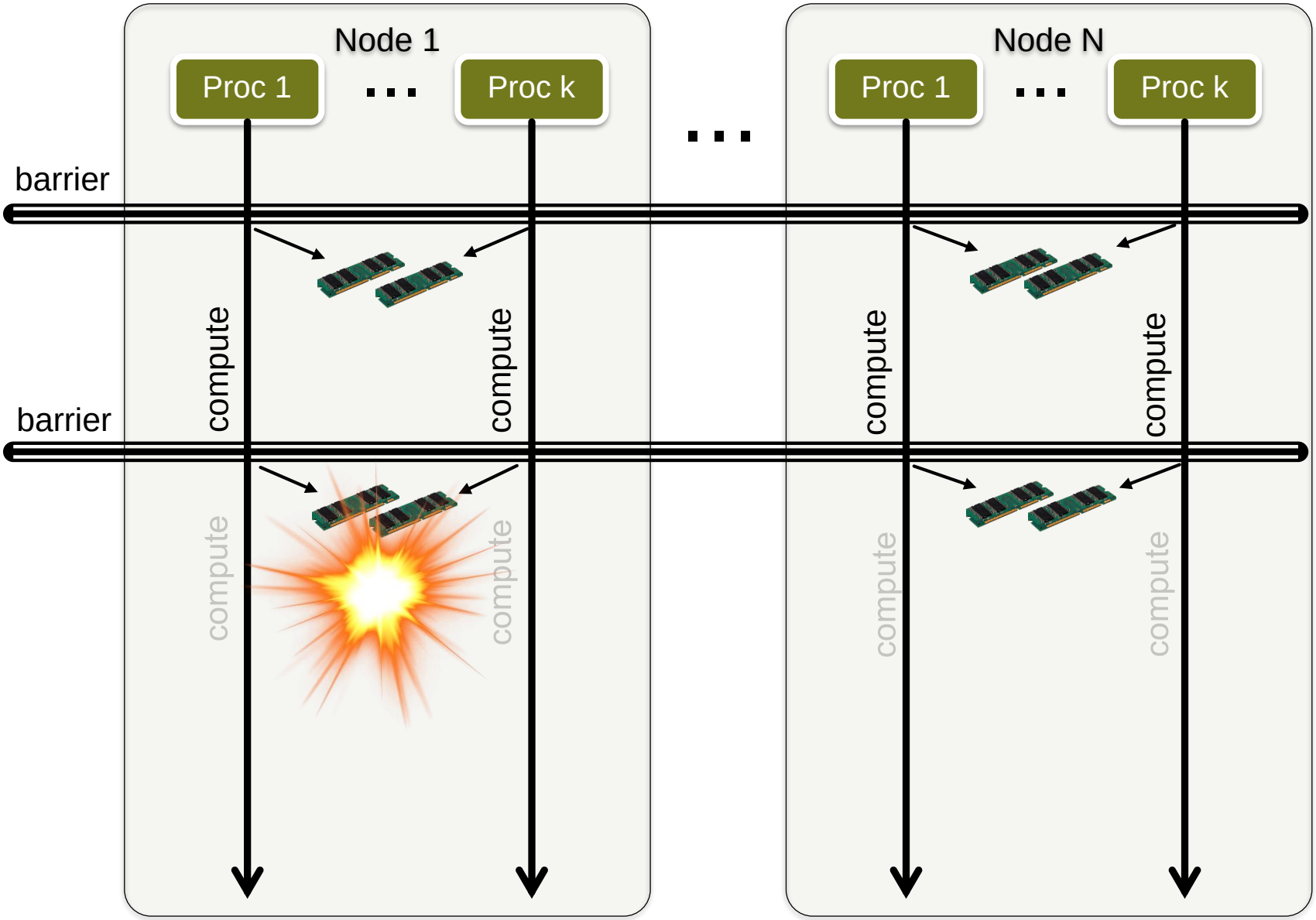
COORDINATED CHECKPOINTING (MP)



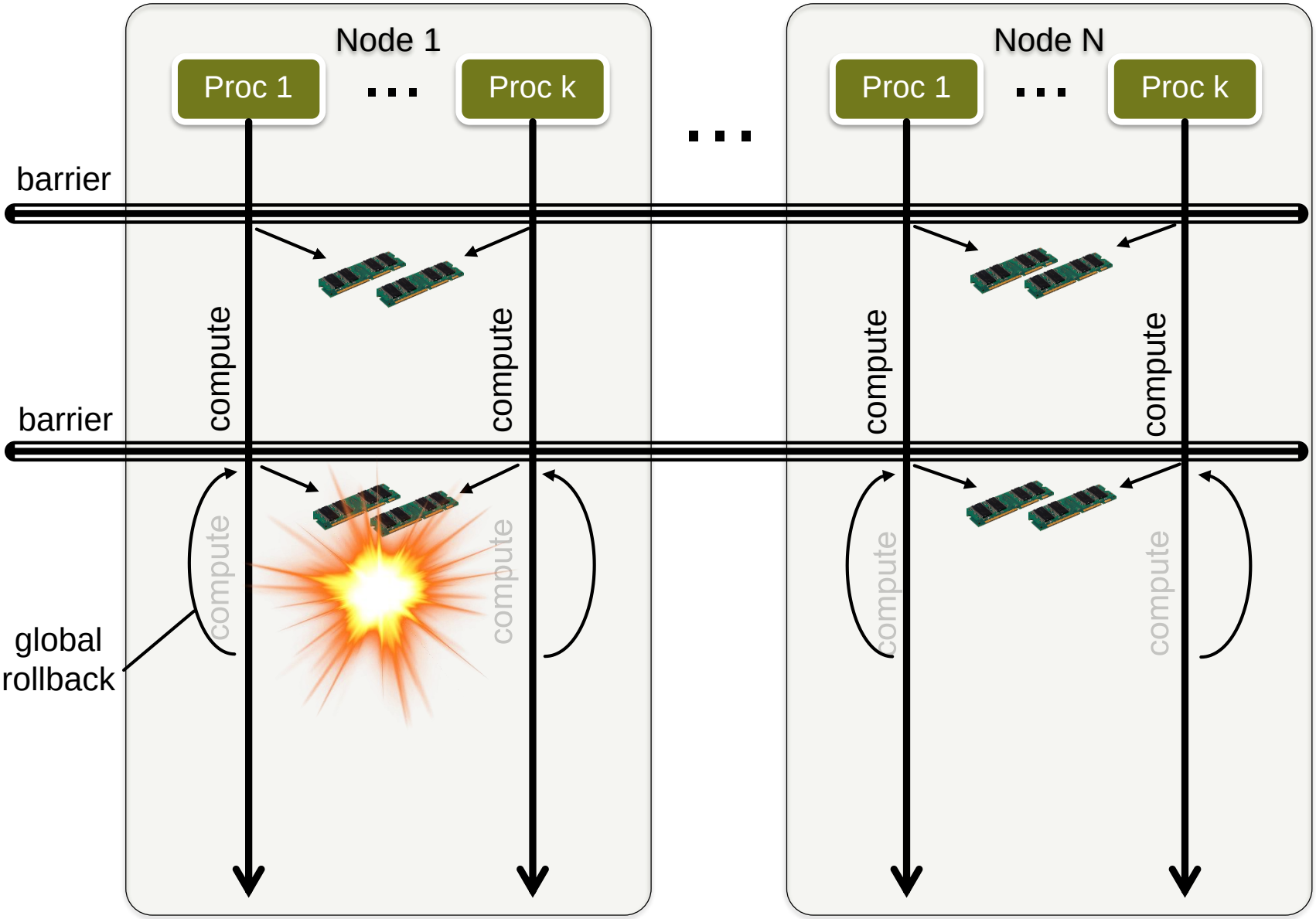
COORDINATED CHECKPOINTING (MP)



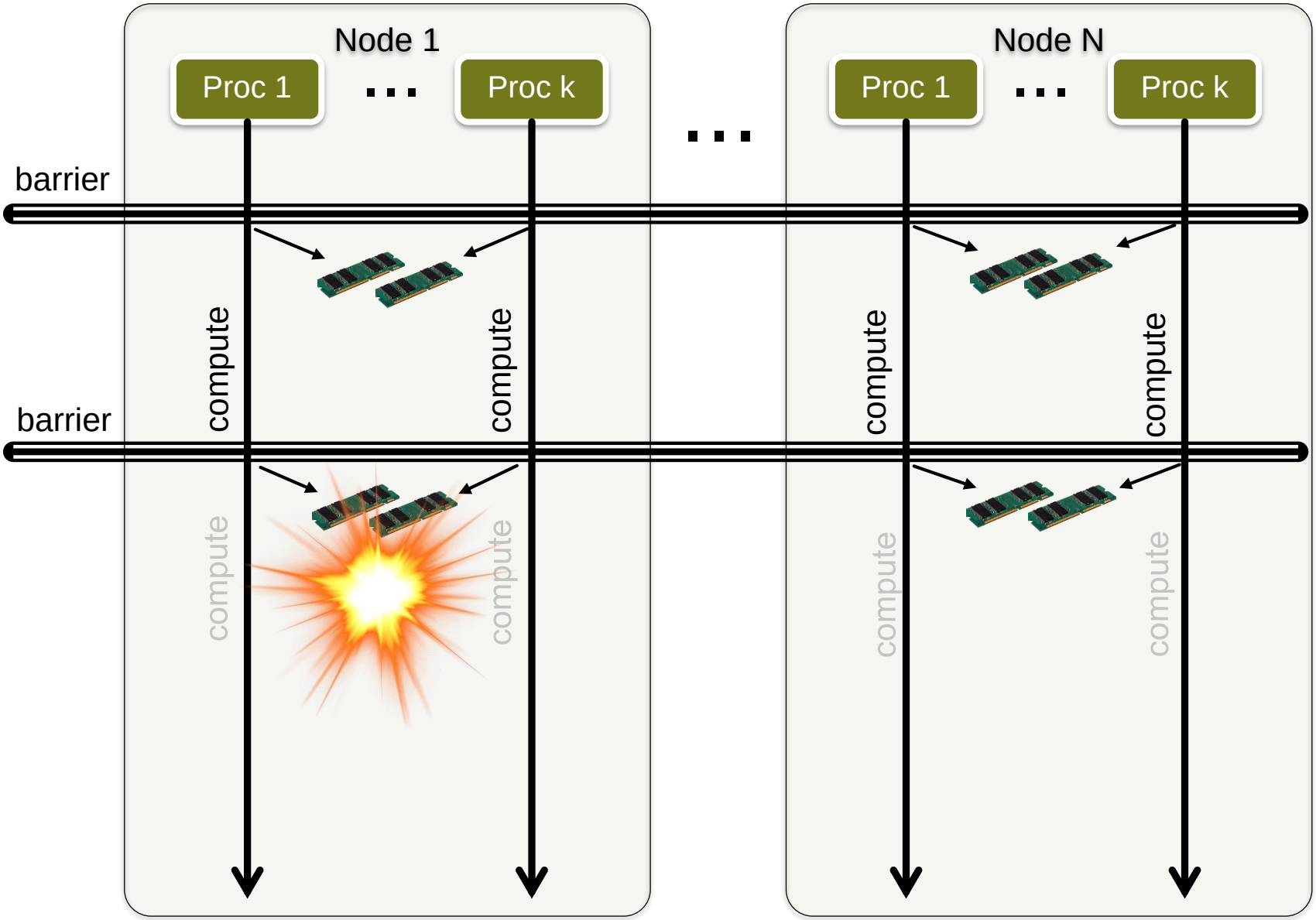
COORDINATED CHECKPOINTING (MP)



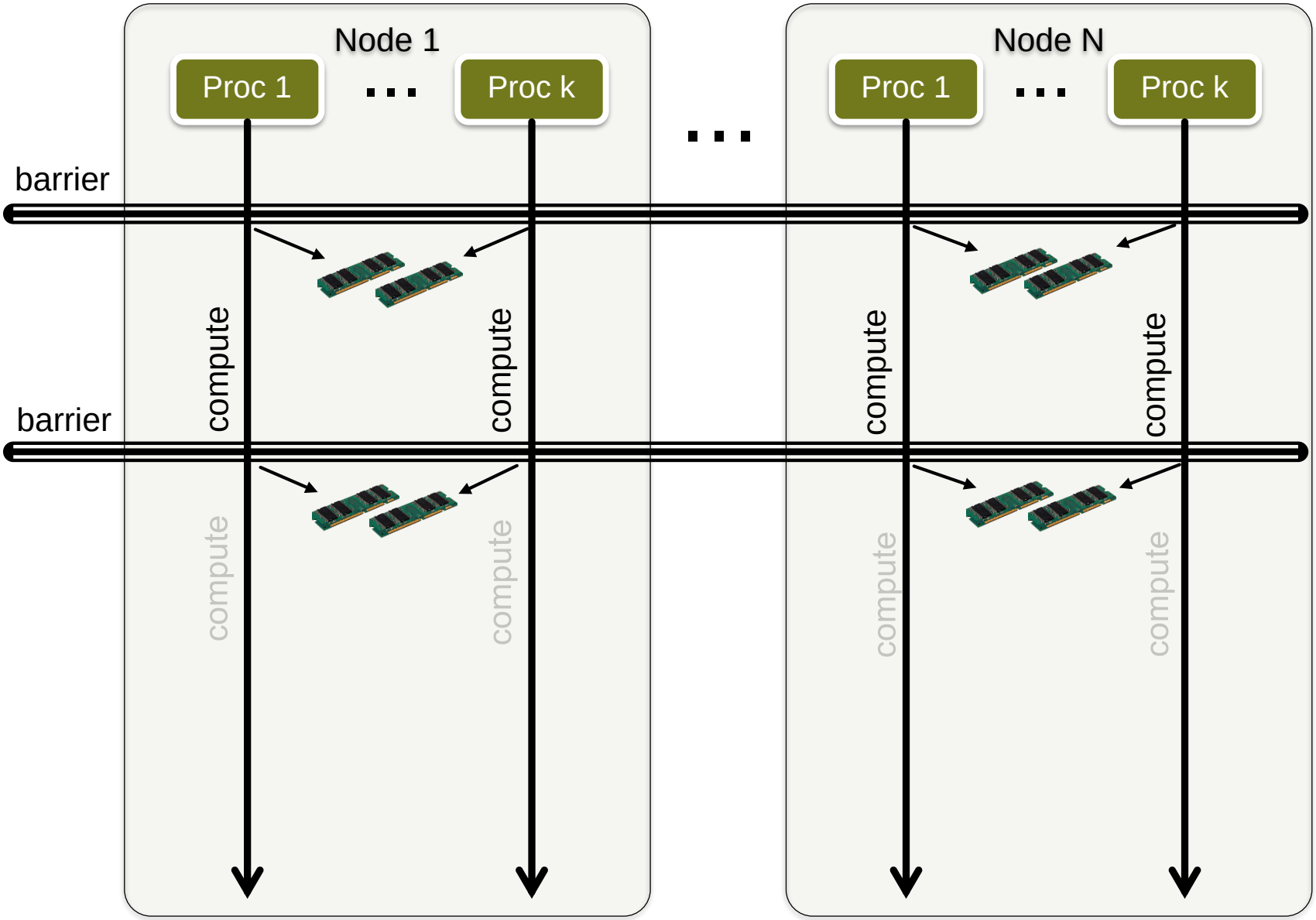
COORDINATED CHECKPOINTING (MP)



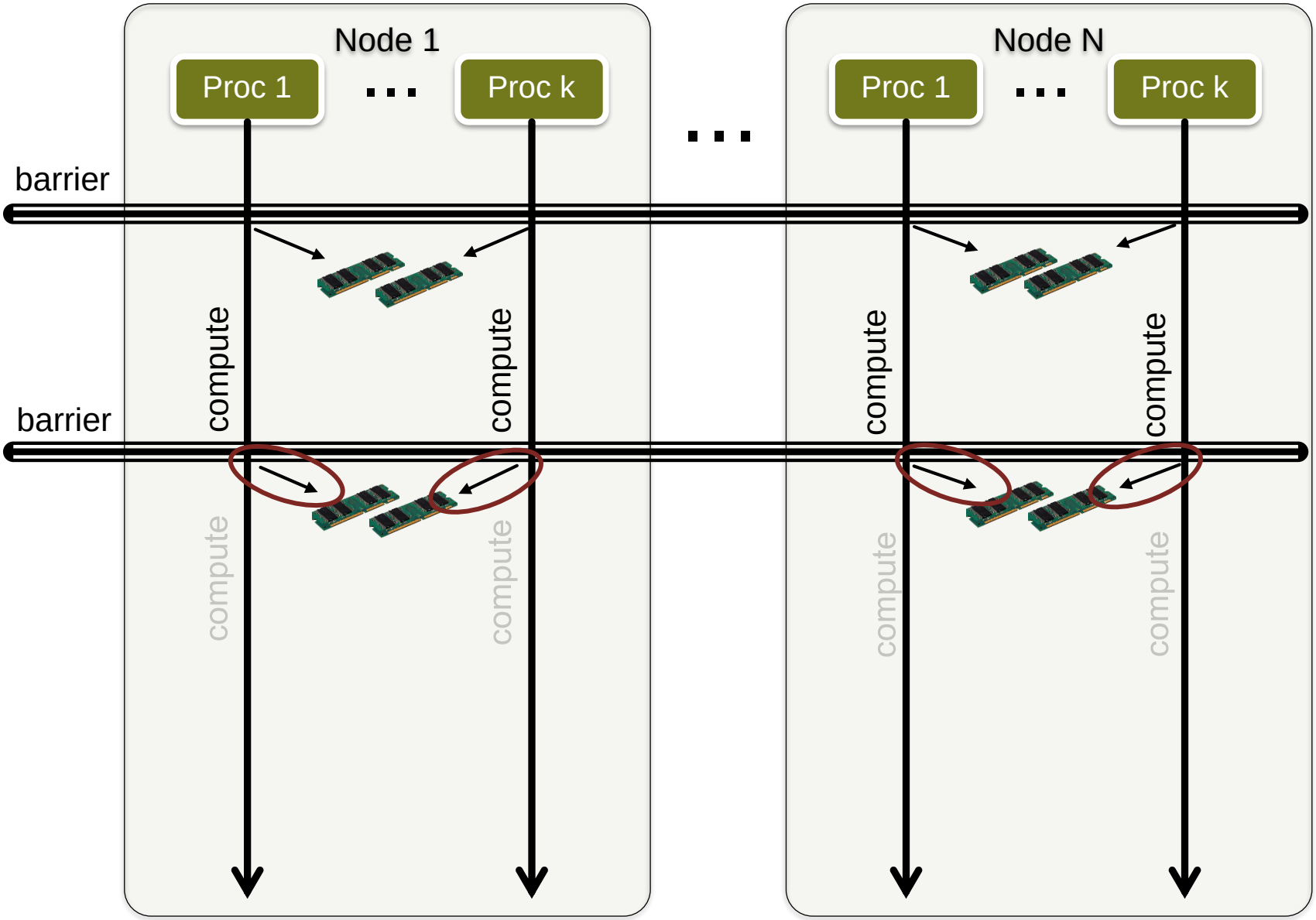
COORDINATED CHECKPOINTING (MP)



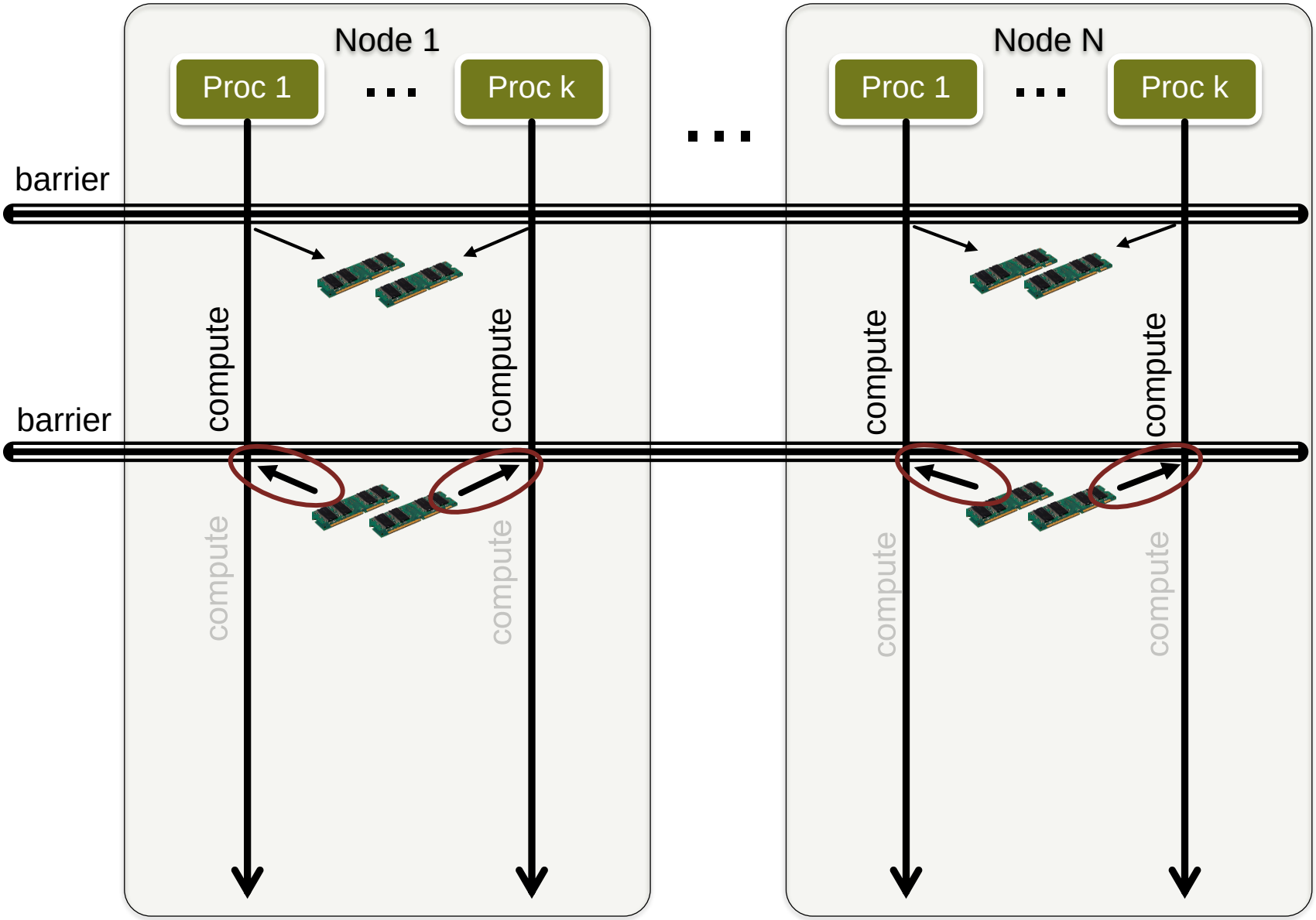
COORDINATED CHECKPOINTING (MP)



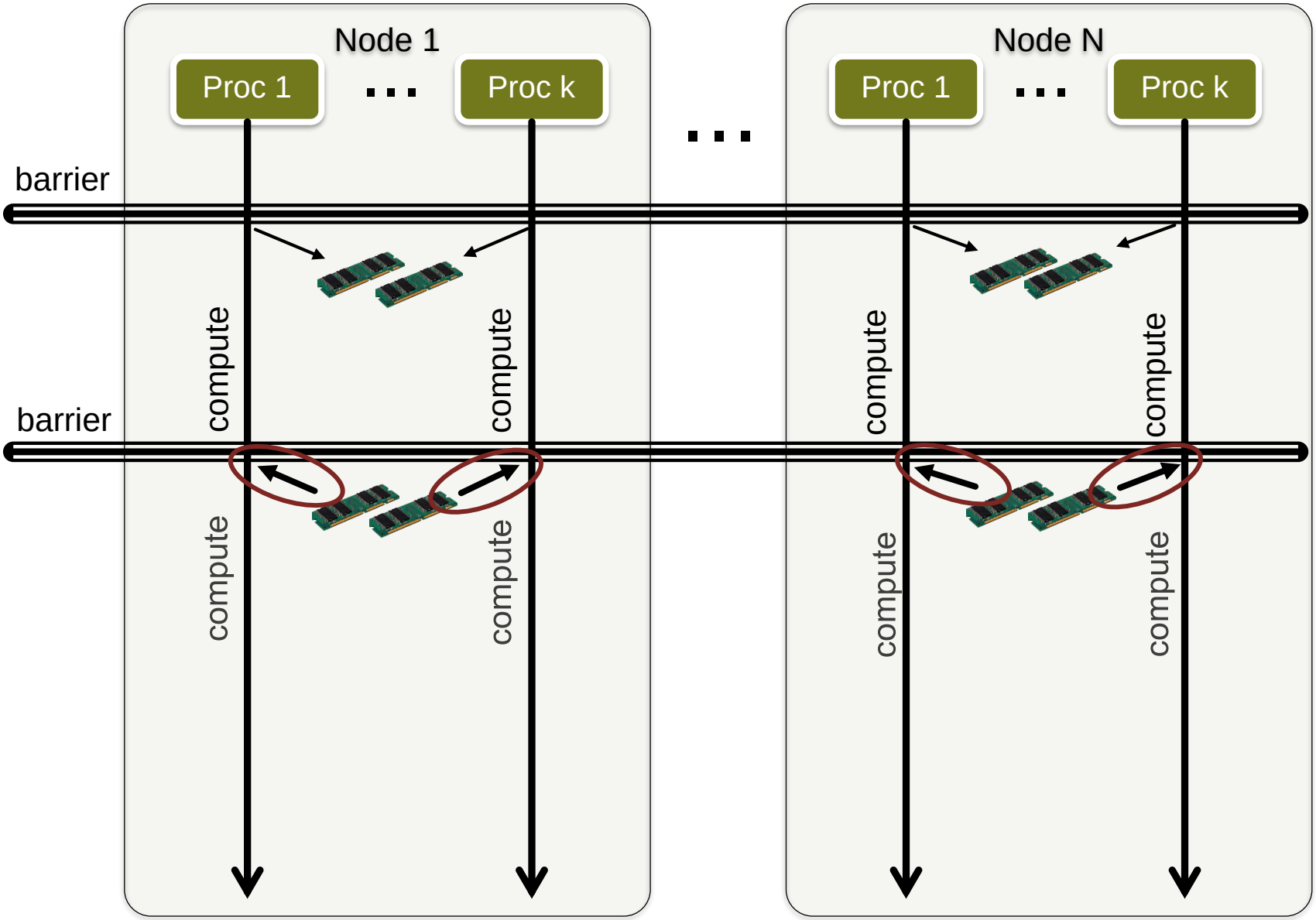
COORDINATED CHECKPOINTING (MP)



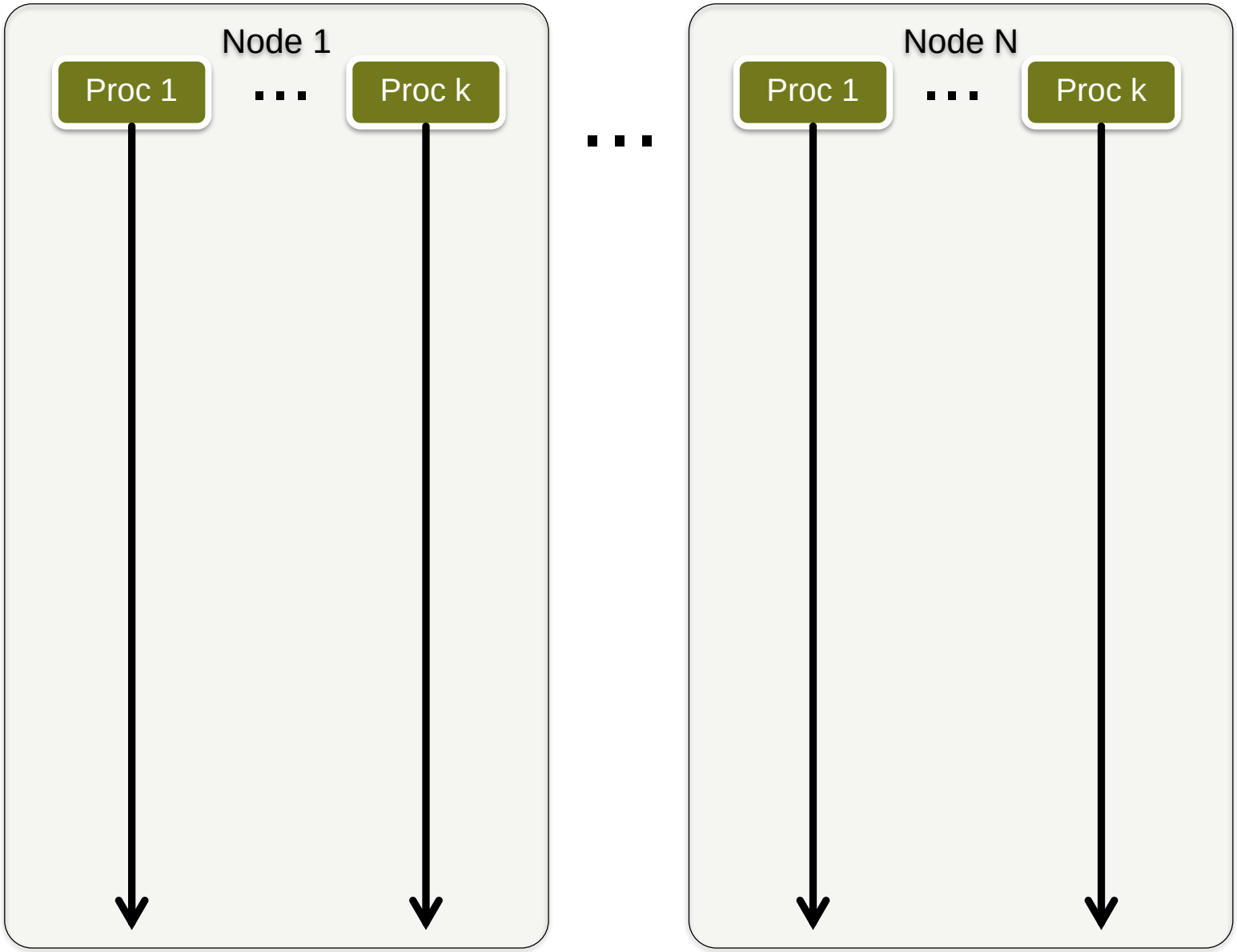
COORDINATED CHECKPOINTING (MP)



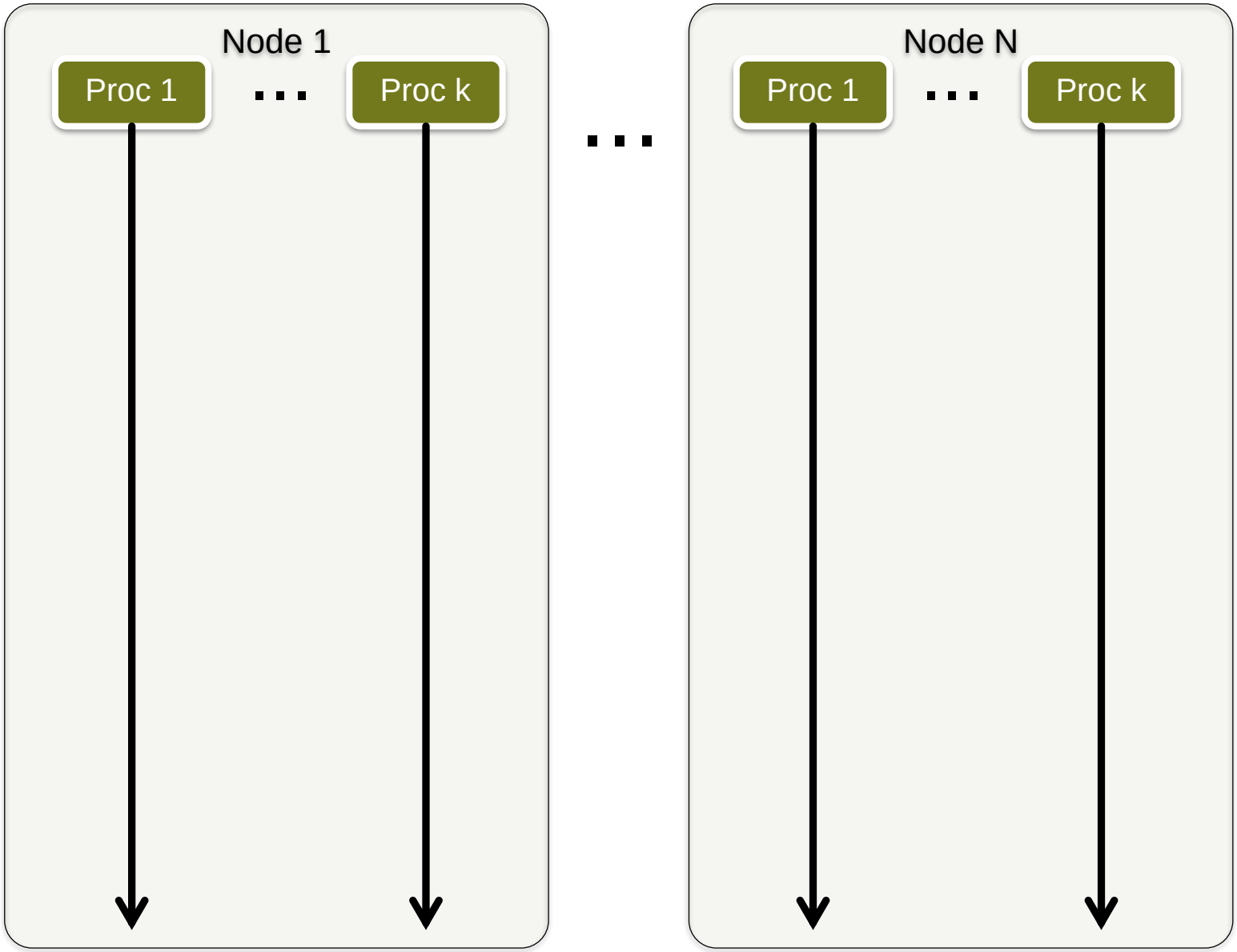
COORDINATED CHECKPOINTING (MP)



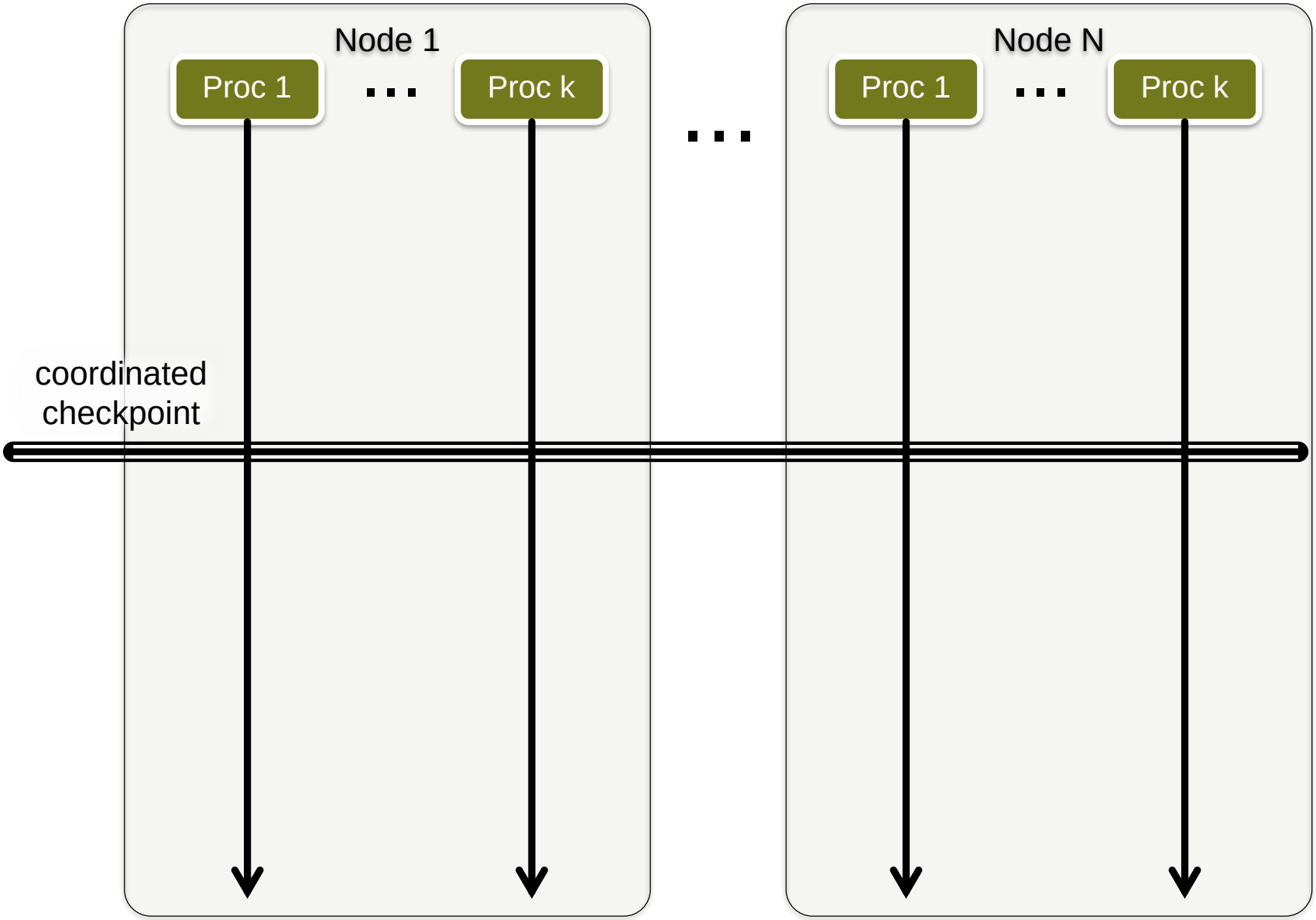
COORDINATED CHECKPOINTING (MP)



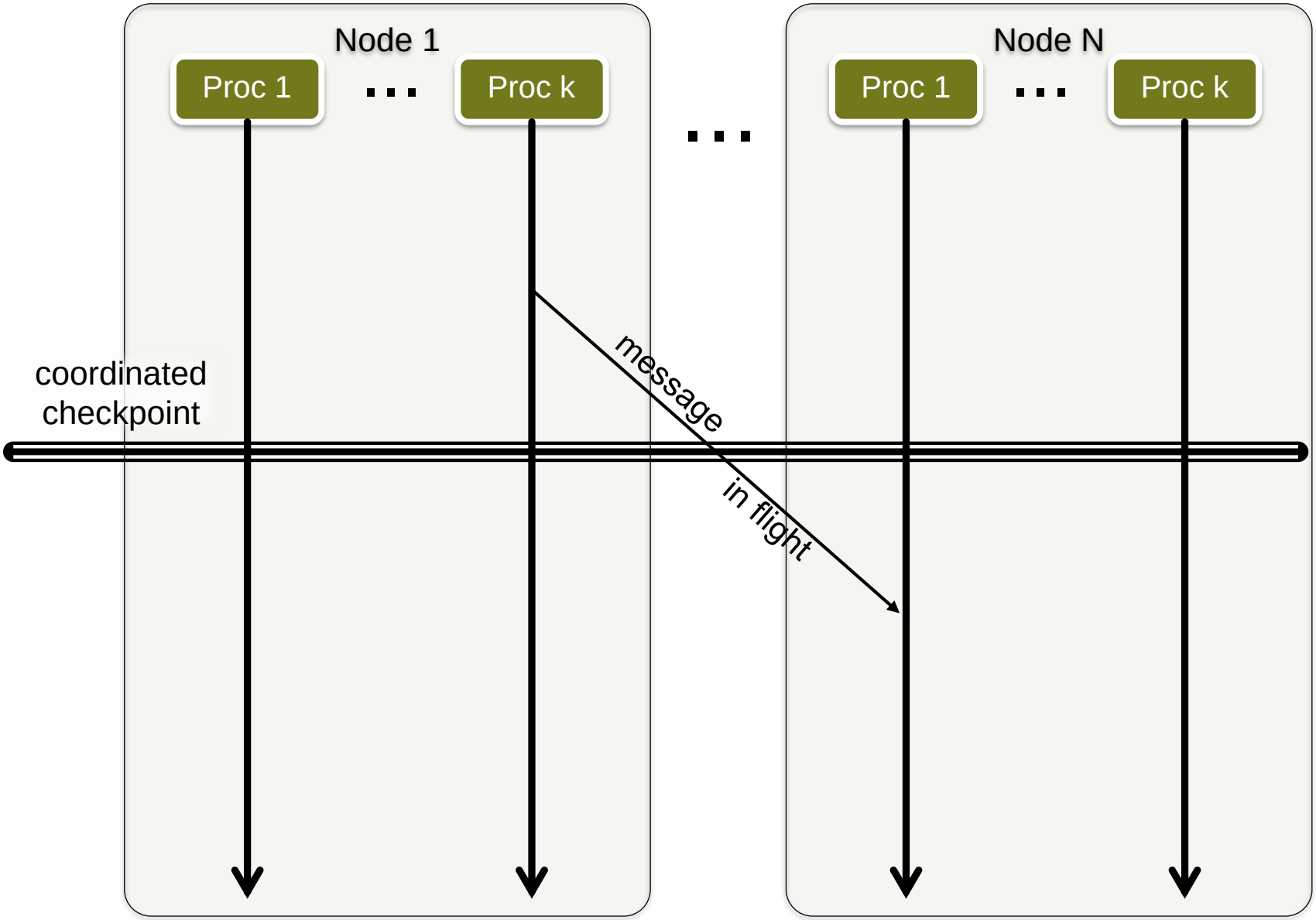
COORDINATED CHECKPOINTING (MP)



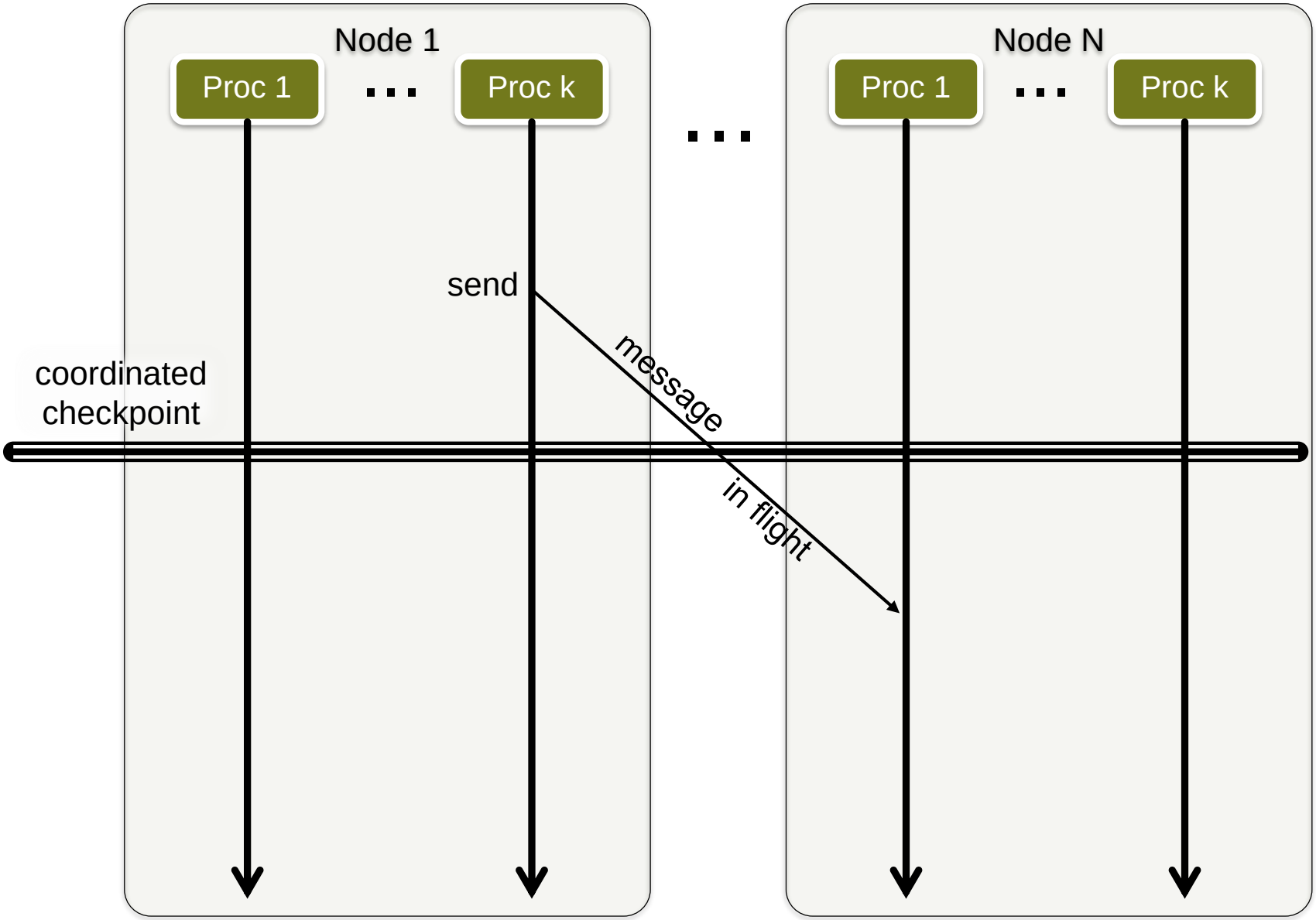
COORDINATED CHECKPOINTING (MP)



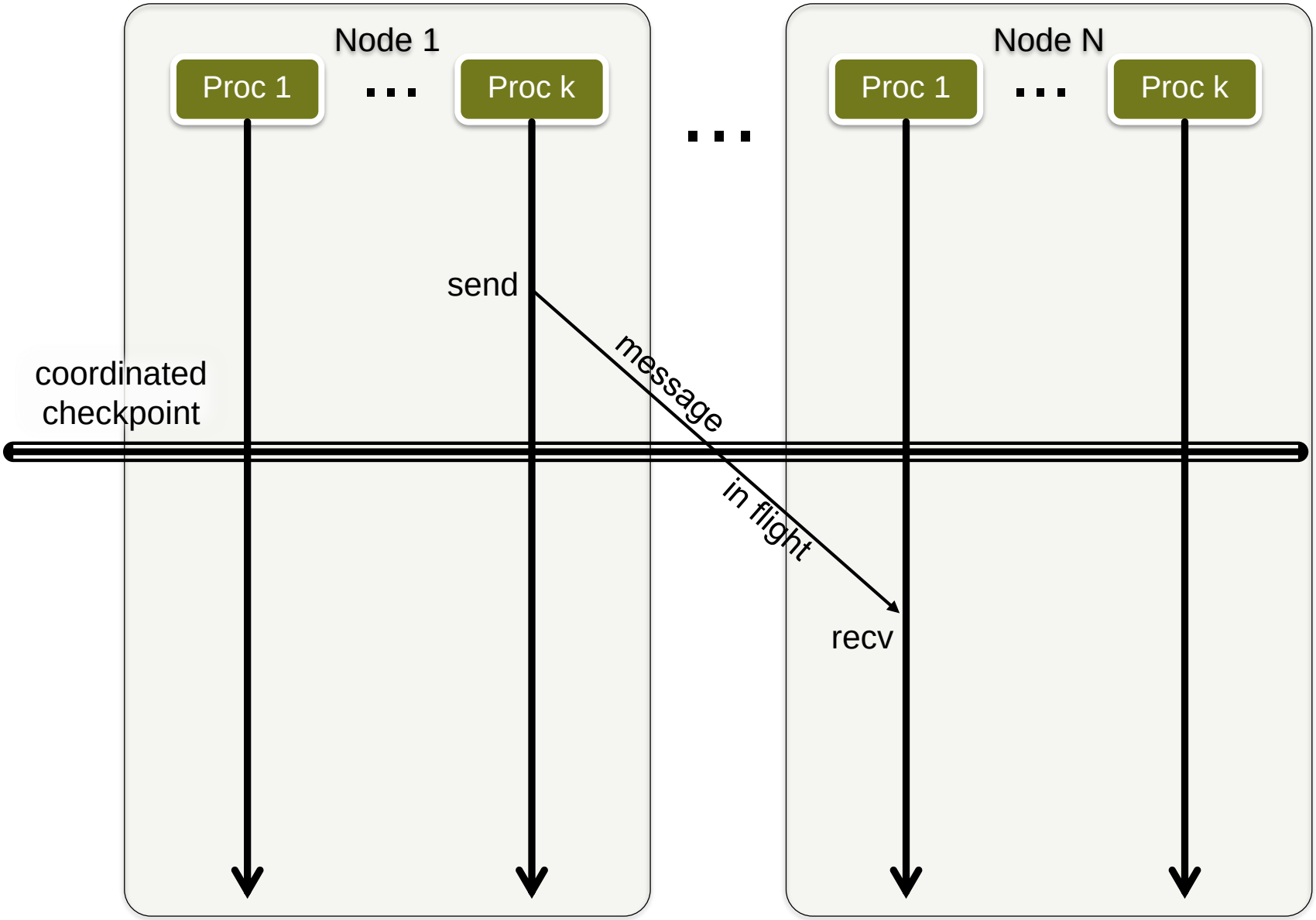
COORDINATED CHECKPOINTING (MP)



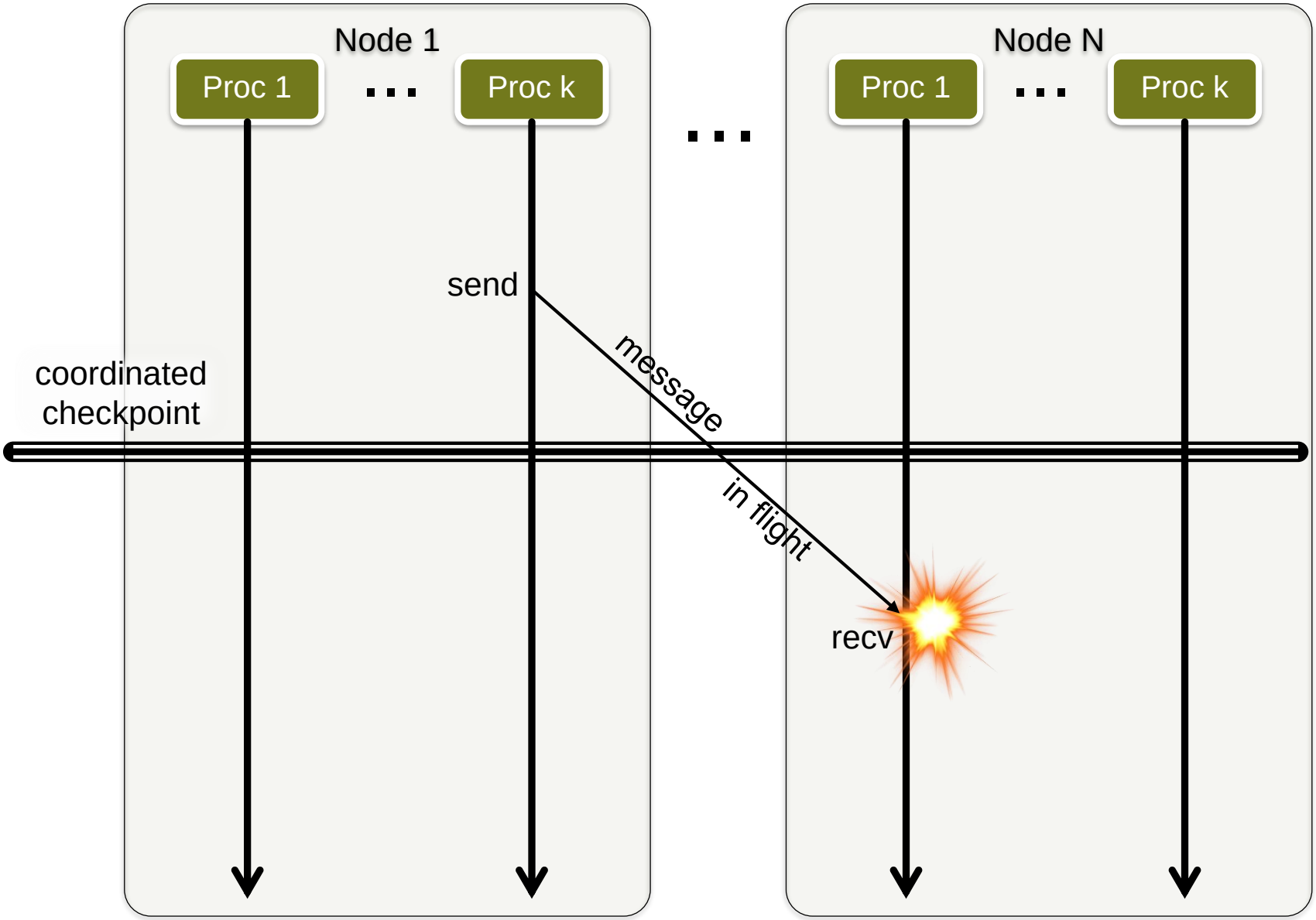
COORDINATED CHECKPOINTING (MP)



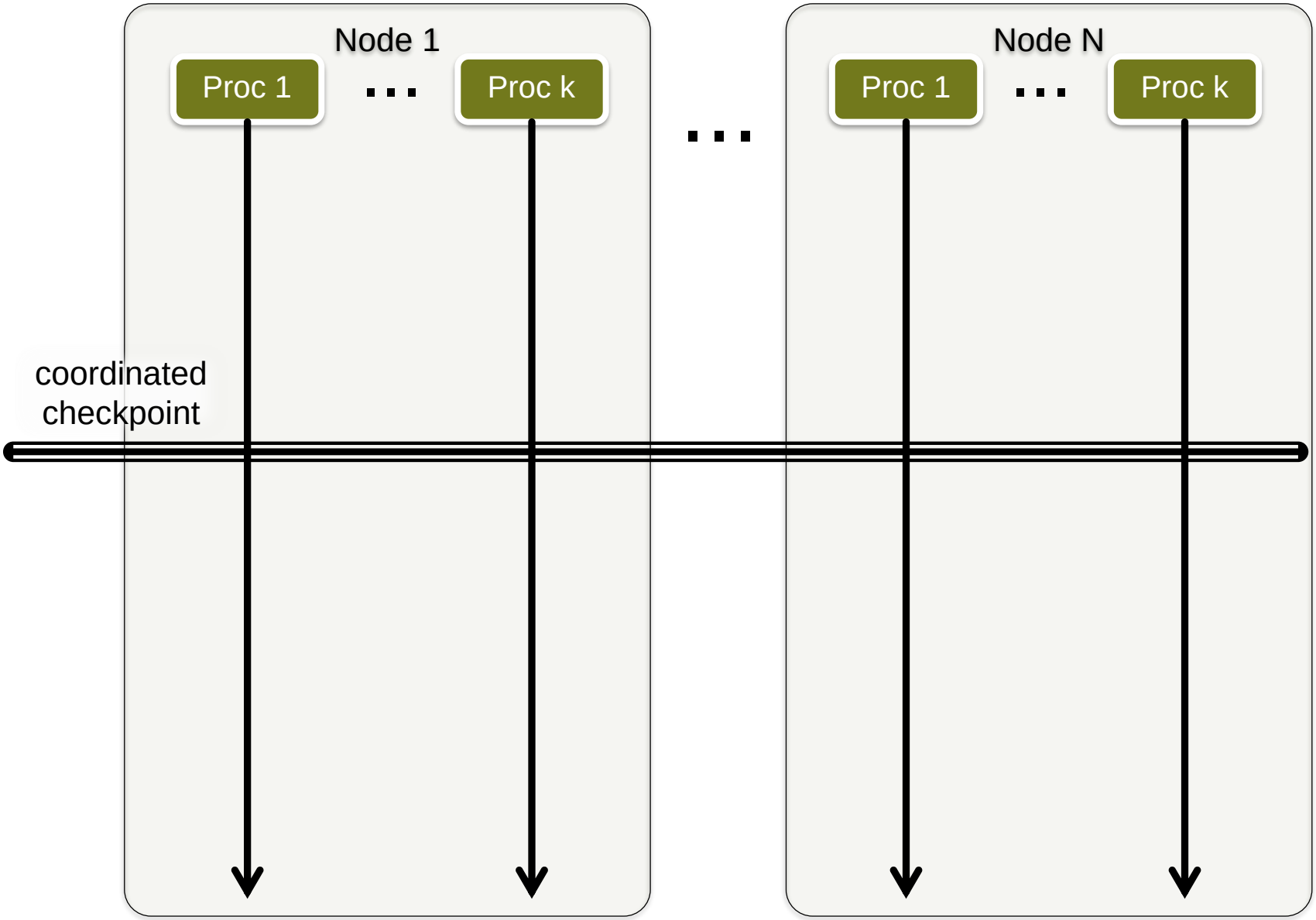
COORDINATED CHECKPOINTING (MP)



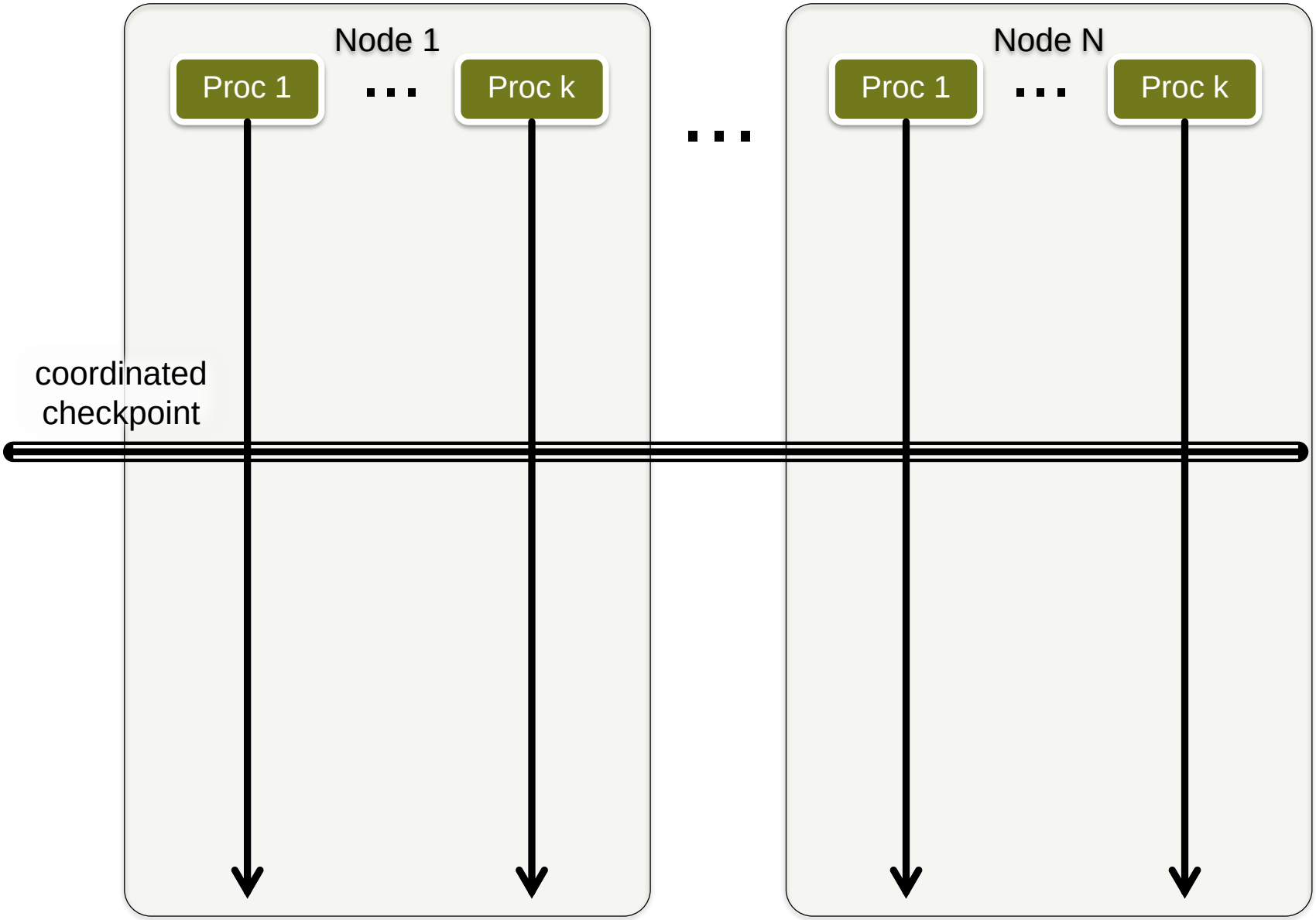
COORDINATED CHECKPOINTING (MP)



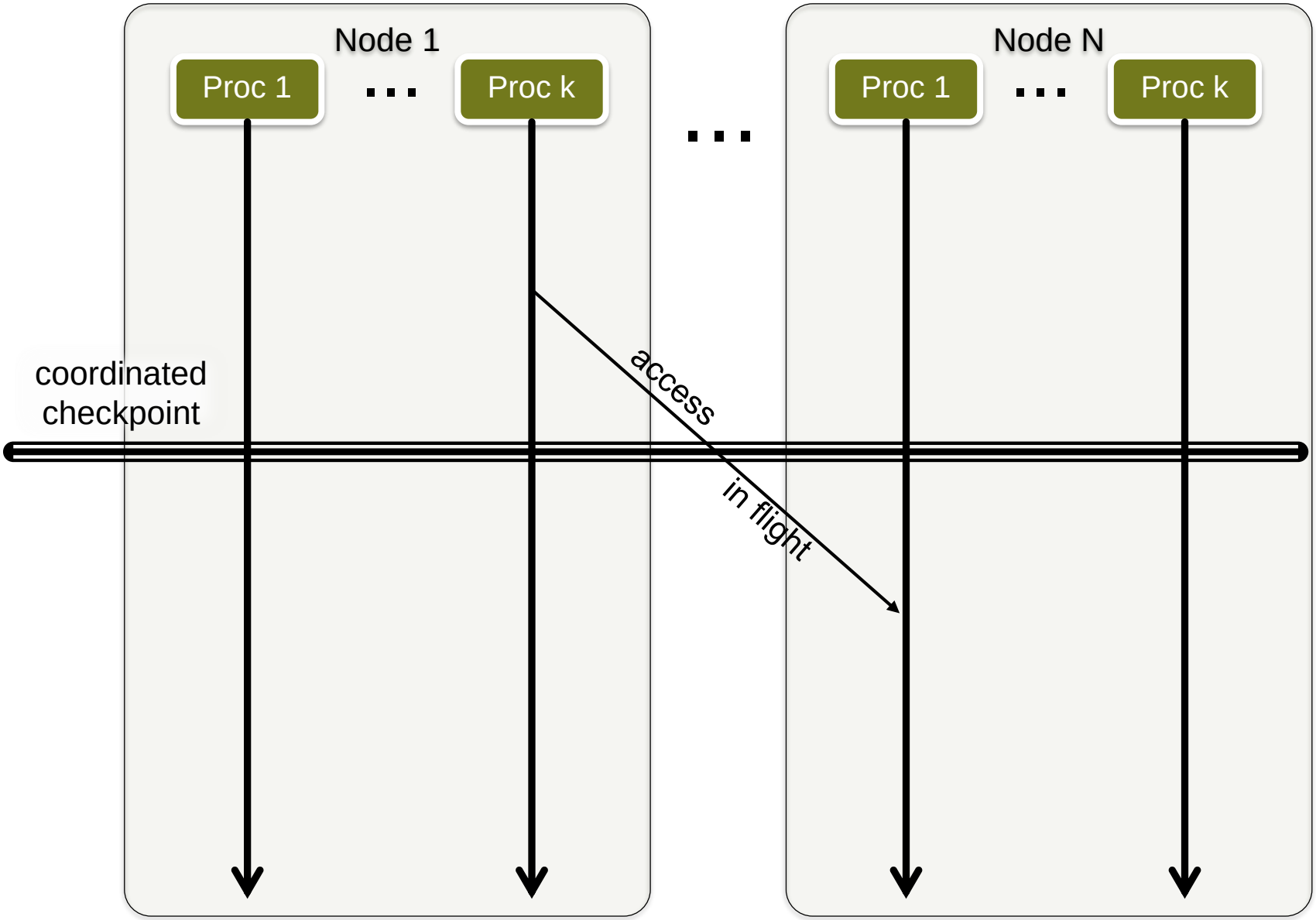
COORDINATED CHECKPOINTING (MP)



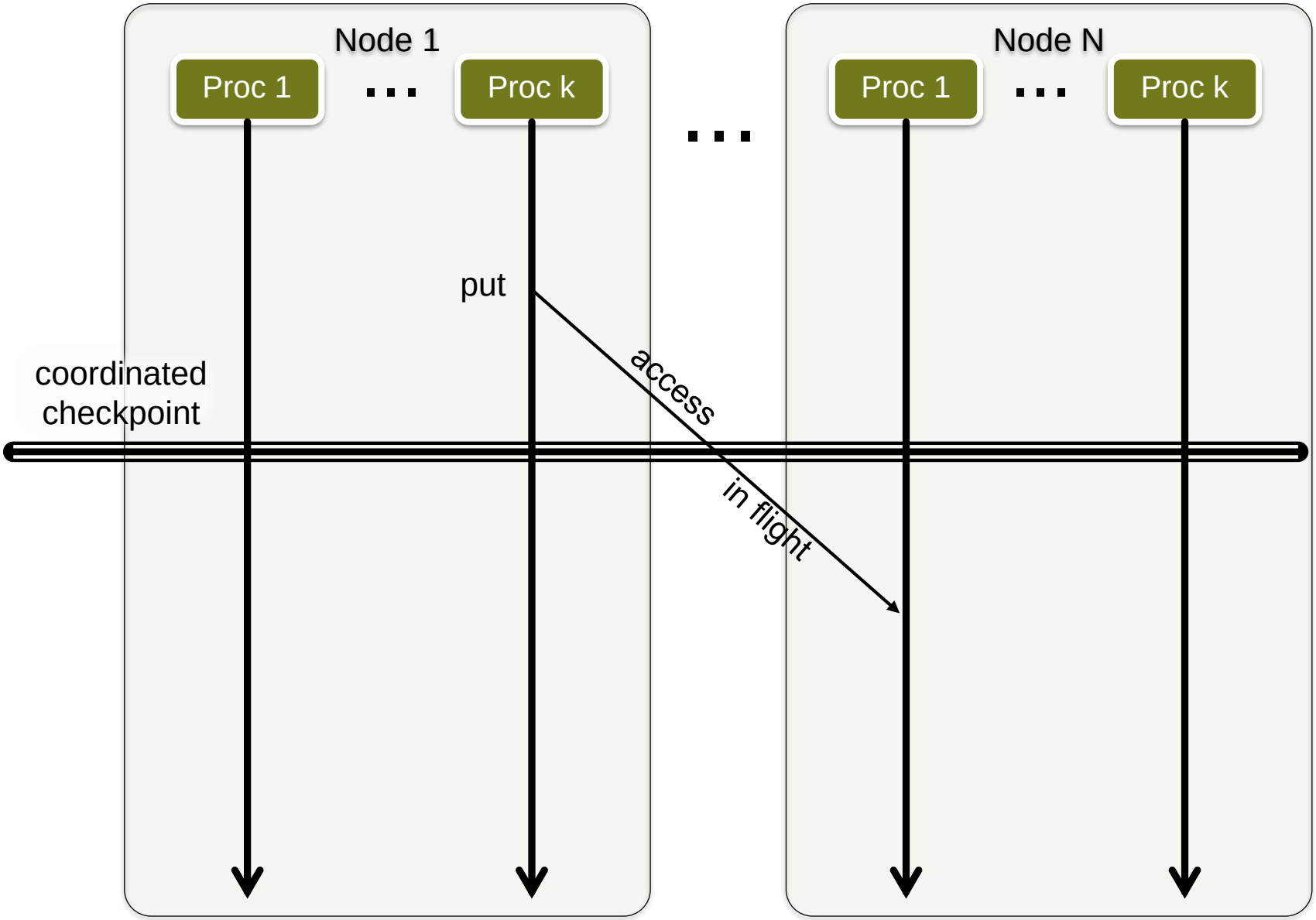
COORDINATED CHECKPOINTING (RMA)



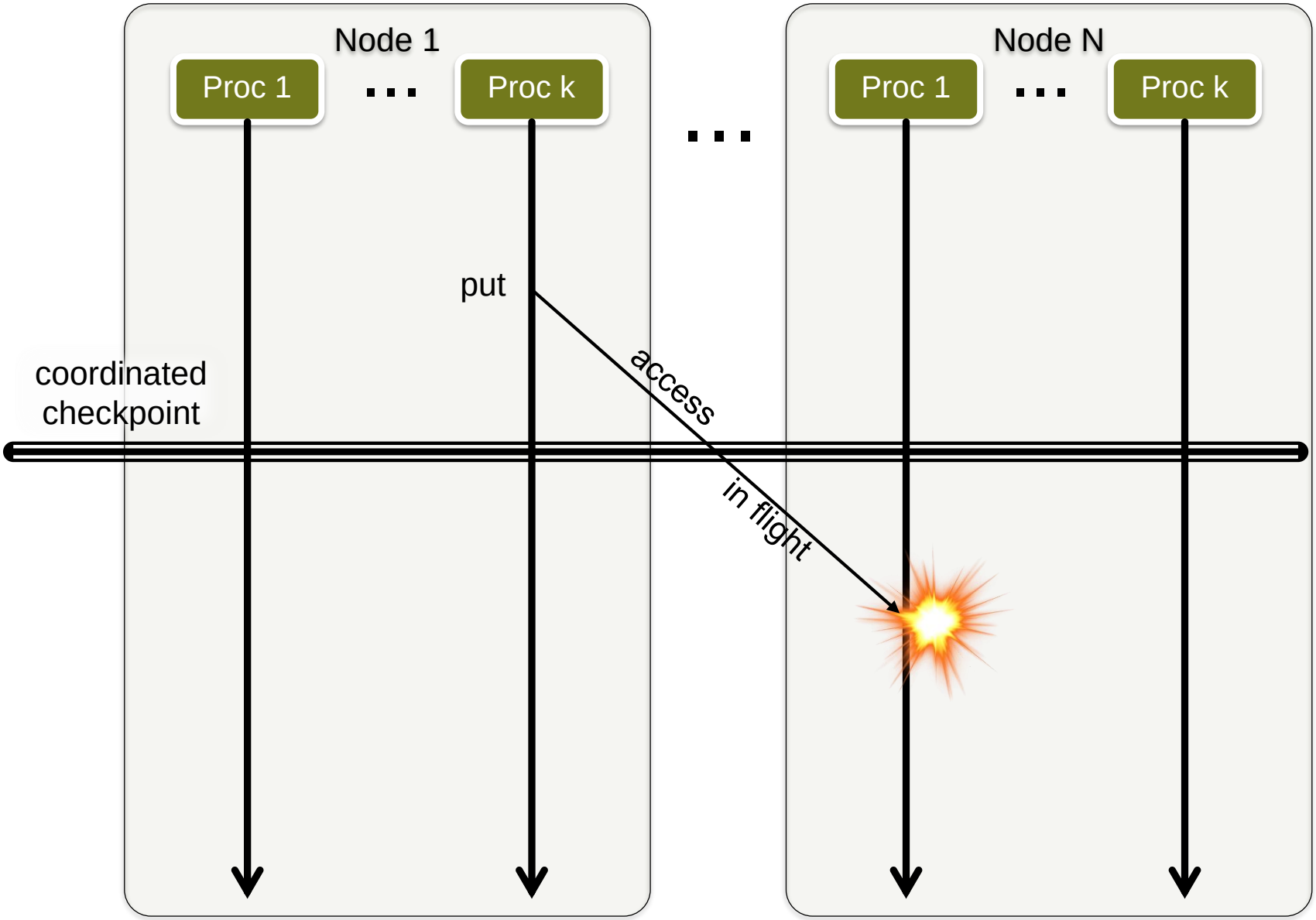
COORDINATED CHECKPOINTING (RMA)



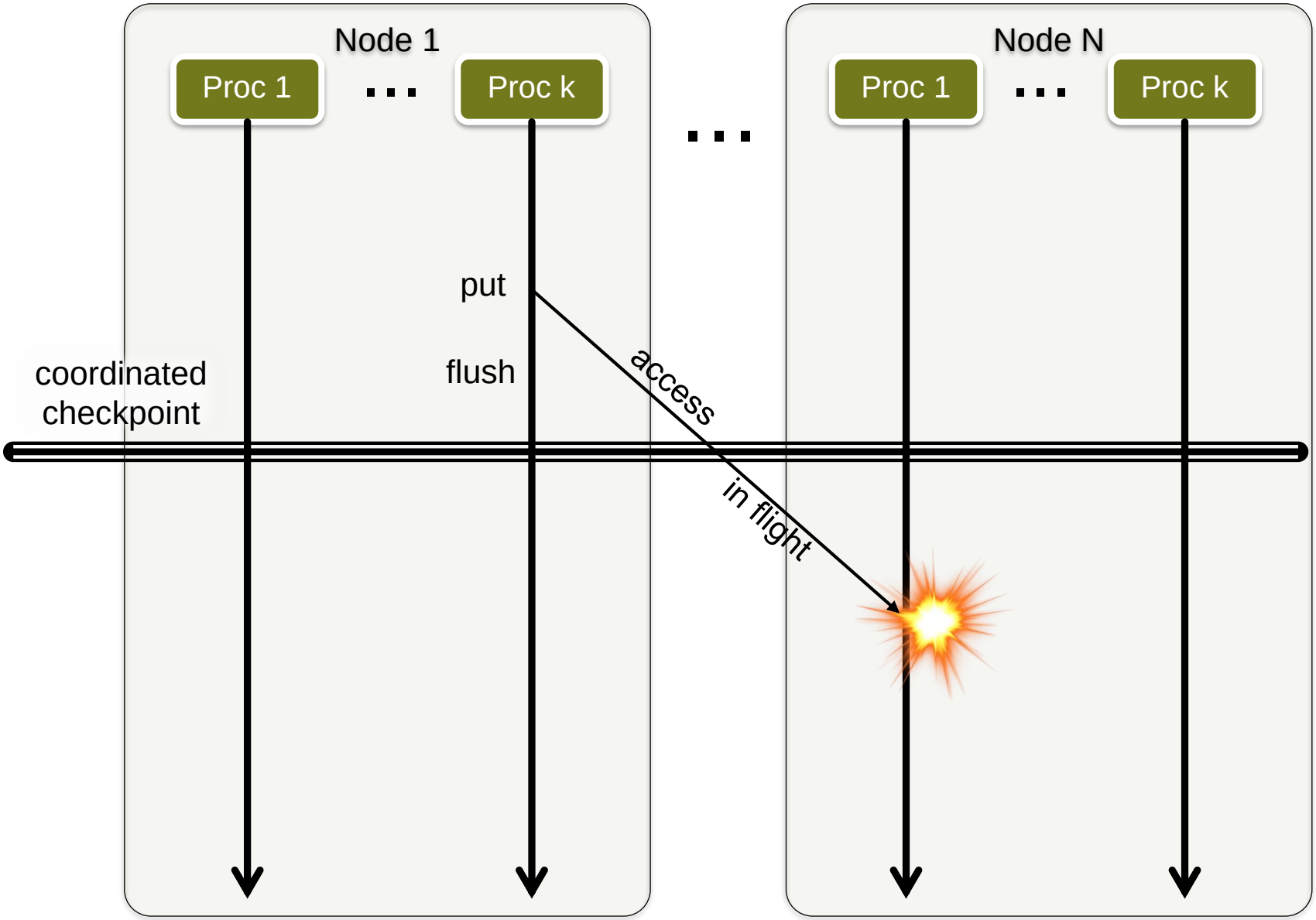
COORDINATED CHECKPOINTING (RMA)



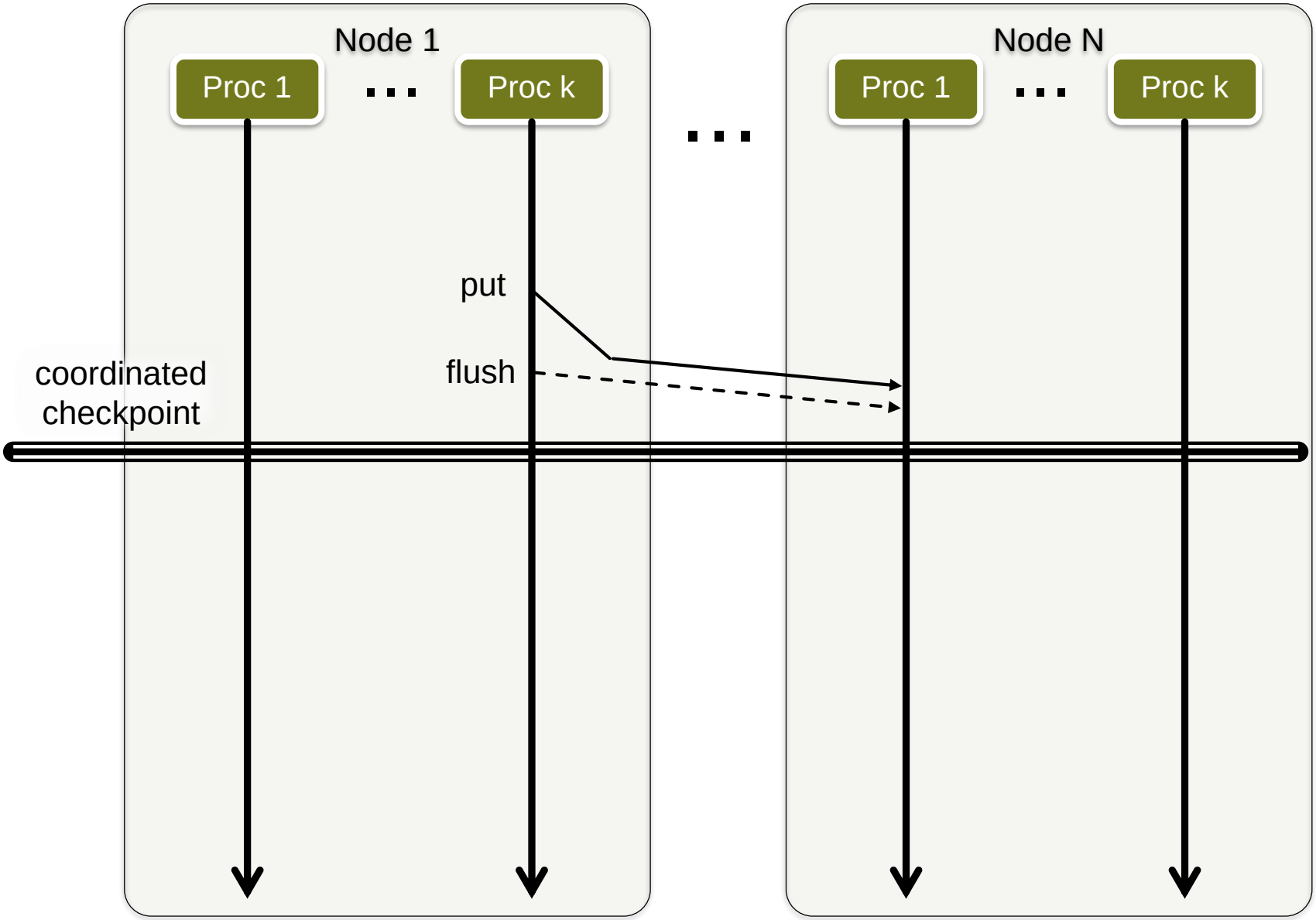
COORDINATED CHECKPOINTING (RMA)



COORDINATED CHECKPOINTING (RMA)



COORDINATED CHECKPOINTING (RMA)



Overview of our Research

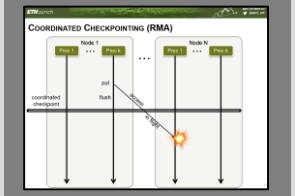
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

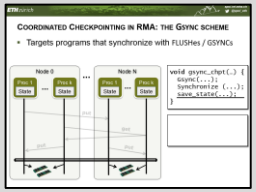
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA



MP vs. RMA



Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Distribution of processes

Decreasing failure prob.

Overview of our Research

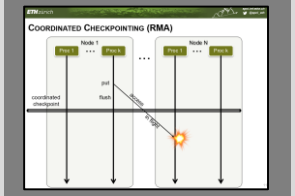
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

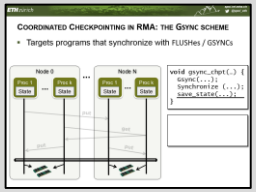
$$\text{PUT}(p \rightrightarrows q)$$
$$\text{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA

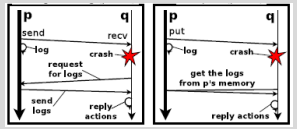


MP vs. RMA

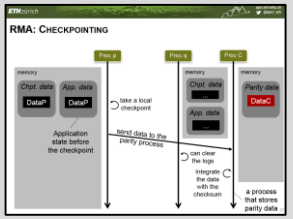


Schemes

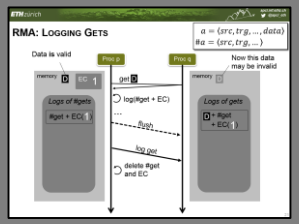
UC in RMA



MP vs. RMA

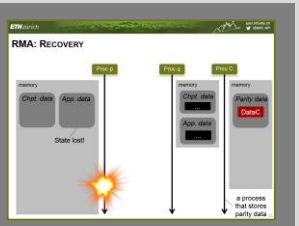


Checkpointing schemes



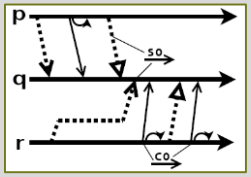
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

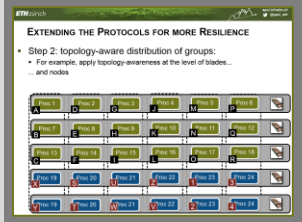
THEOREM 4.2. *The recovery algorithm 2 preserves the* $\xrightarrow{cohb}$ *order (referred to as the* gsync order*).*

Deadlock freedom
Correct recovery

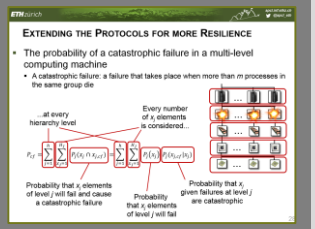
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

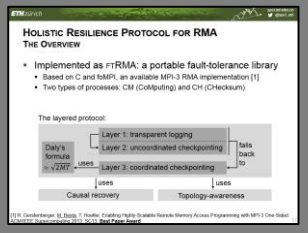


Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\text{PUT}(p \rightrightarrows q)$
$\text{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

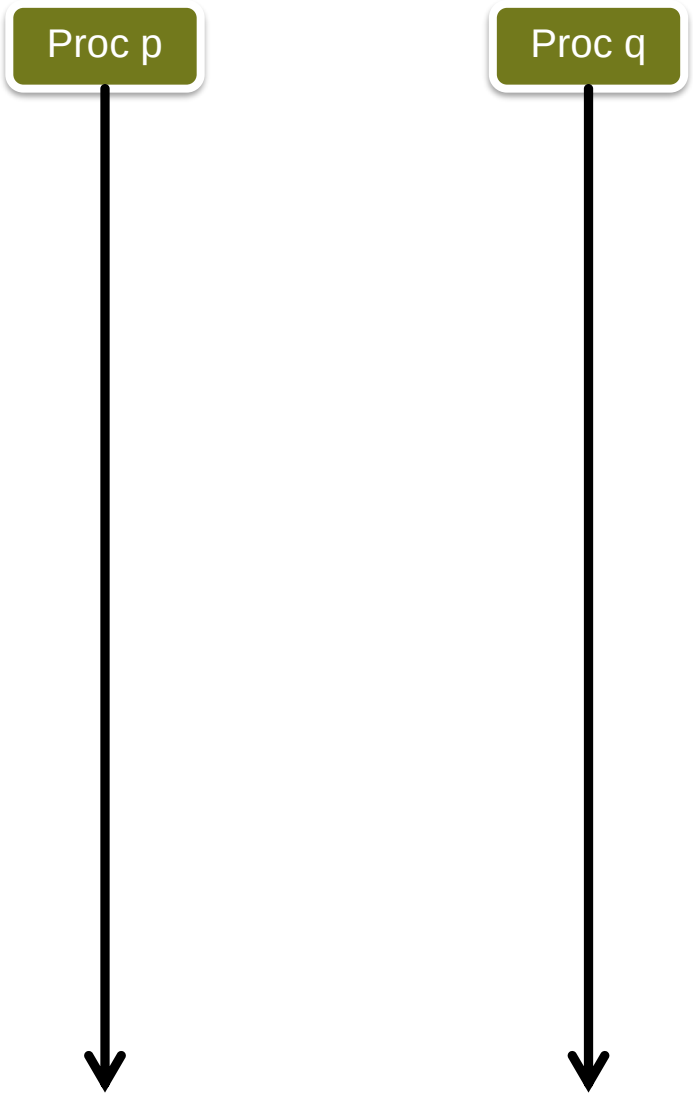
Design

Checkpoints on demand

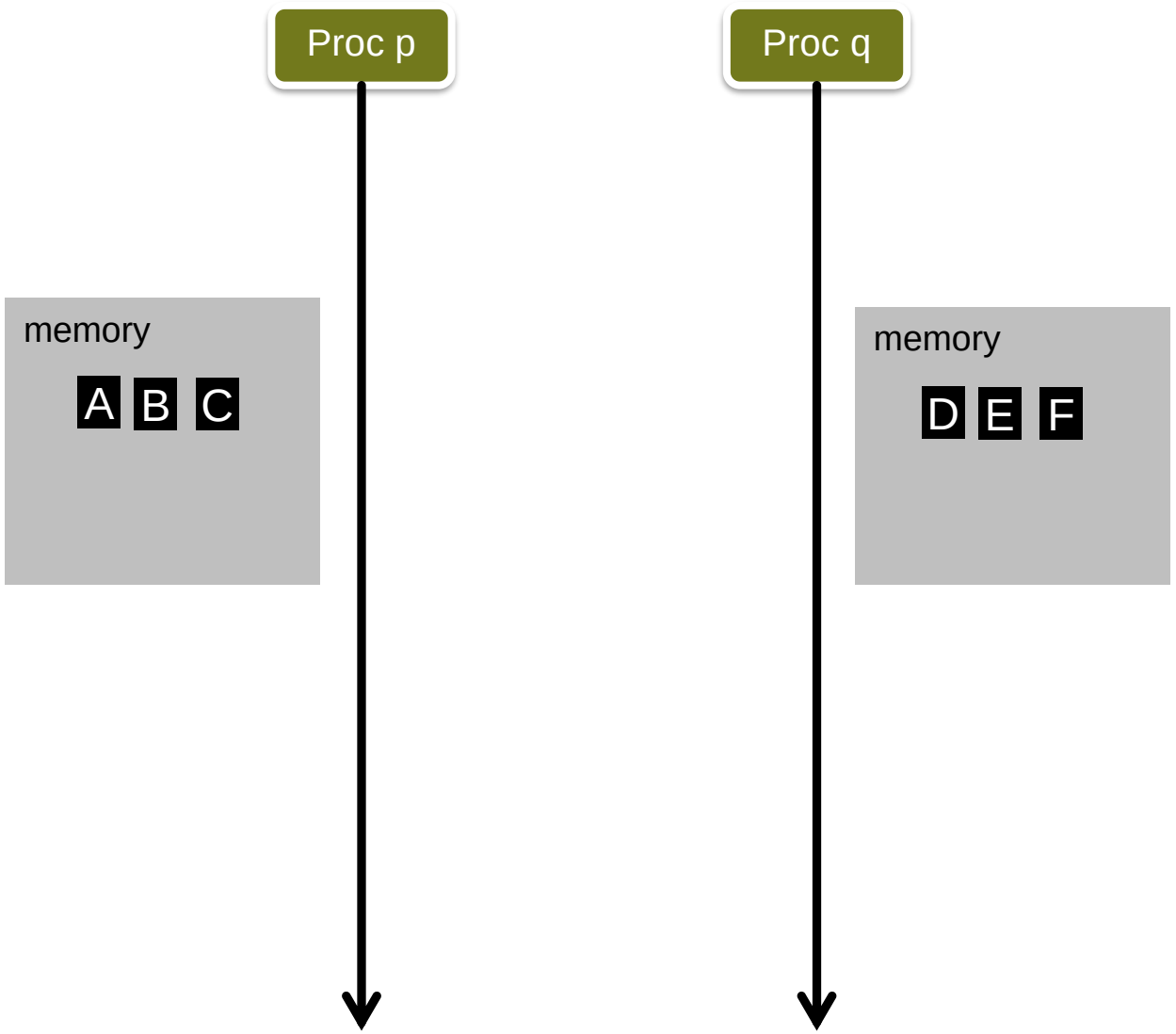
Optimizations

Performance

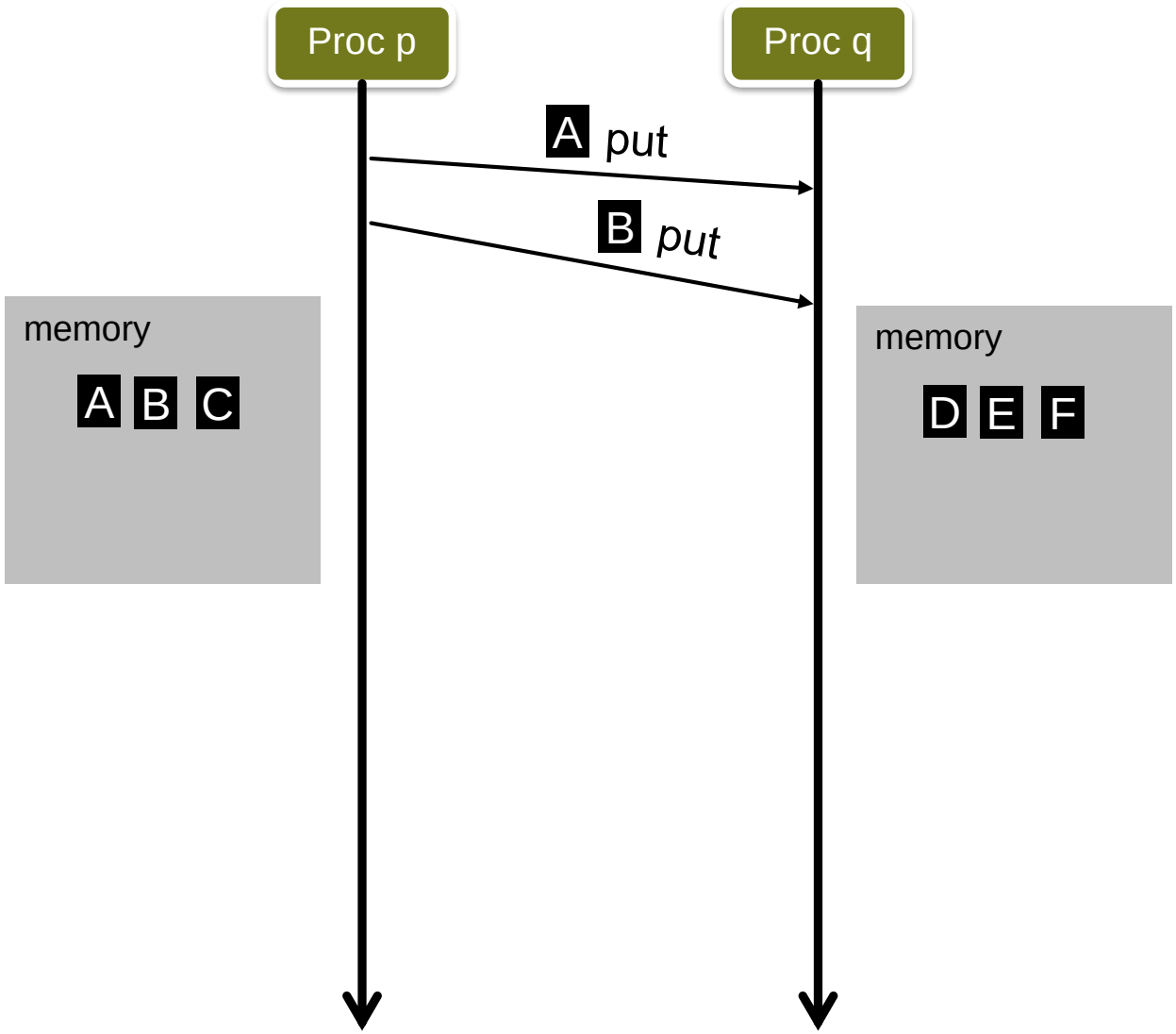
RMA: EPOCHS



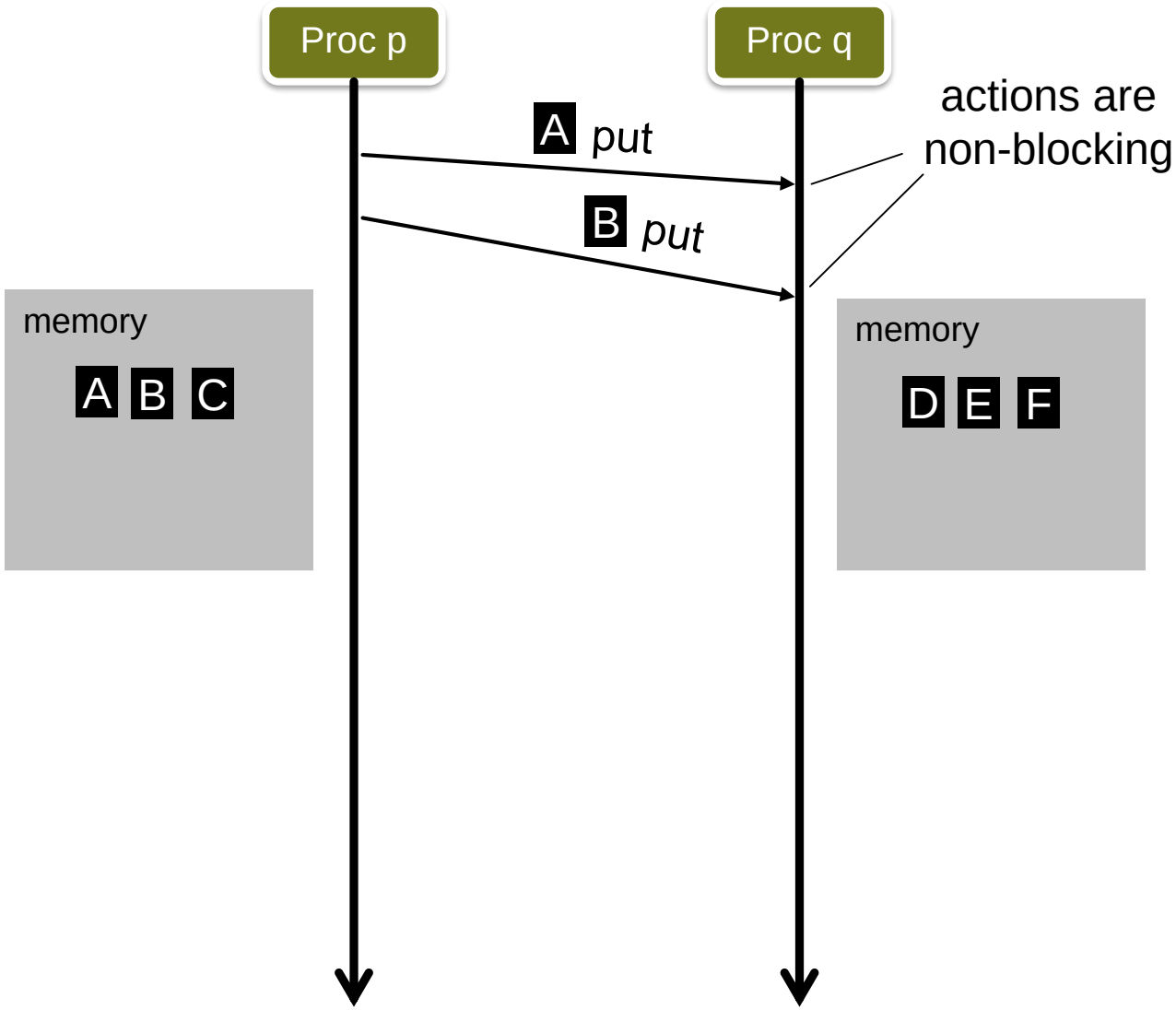
RMA: EPOCHS



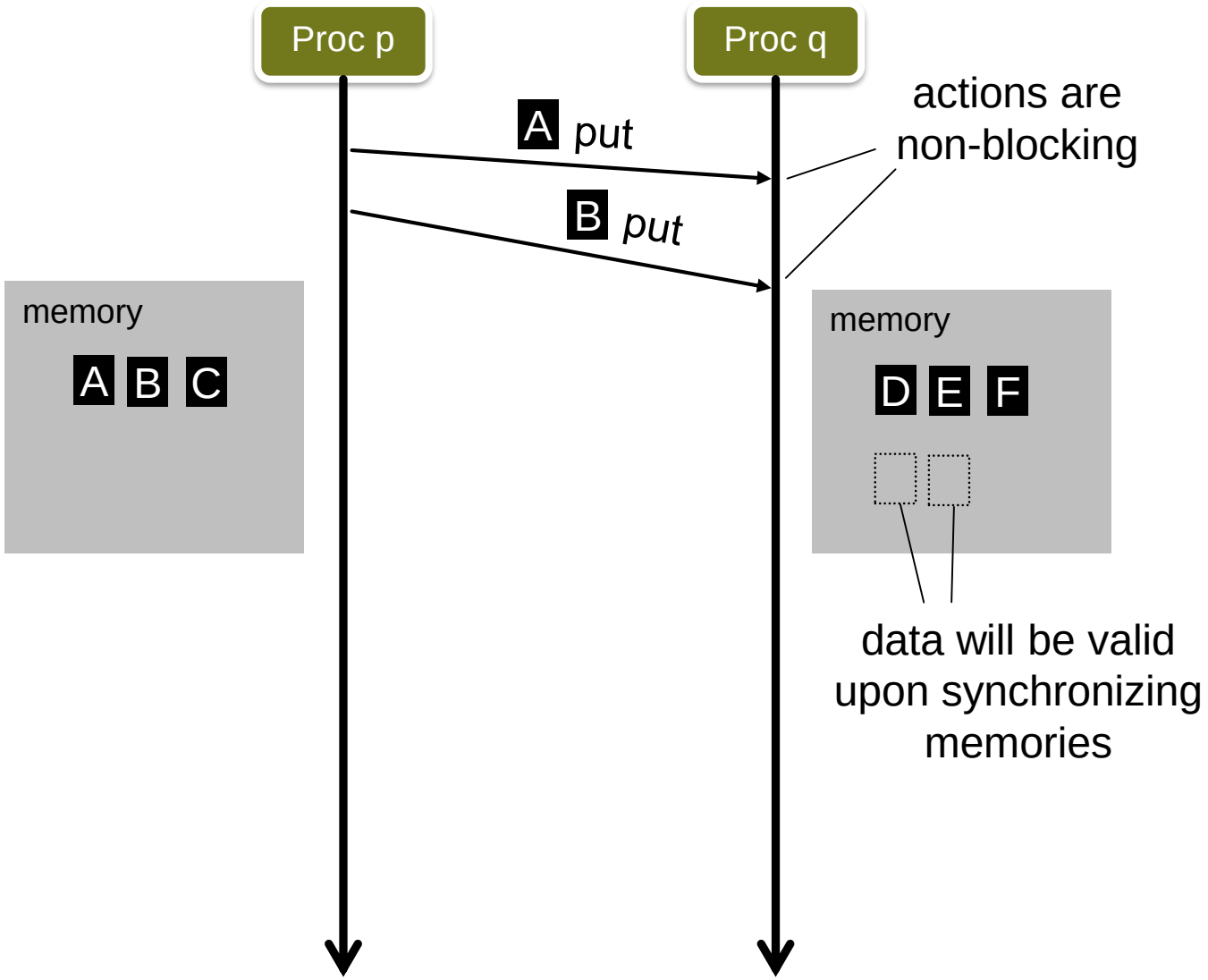
RMA: EPOCHS



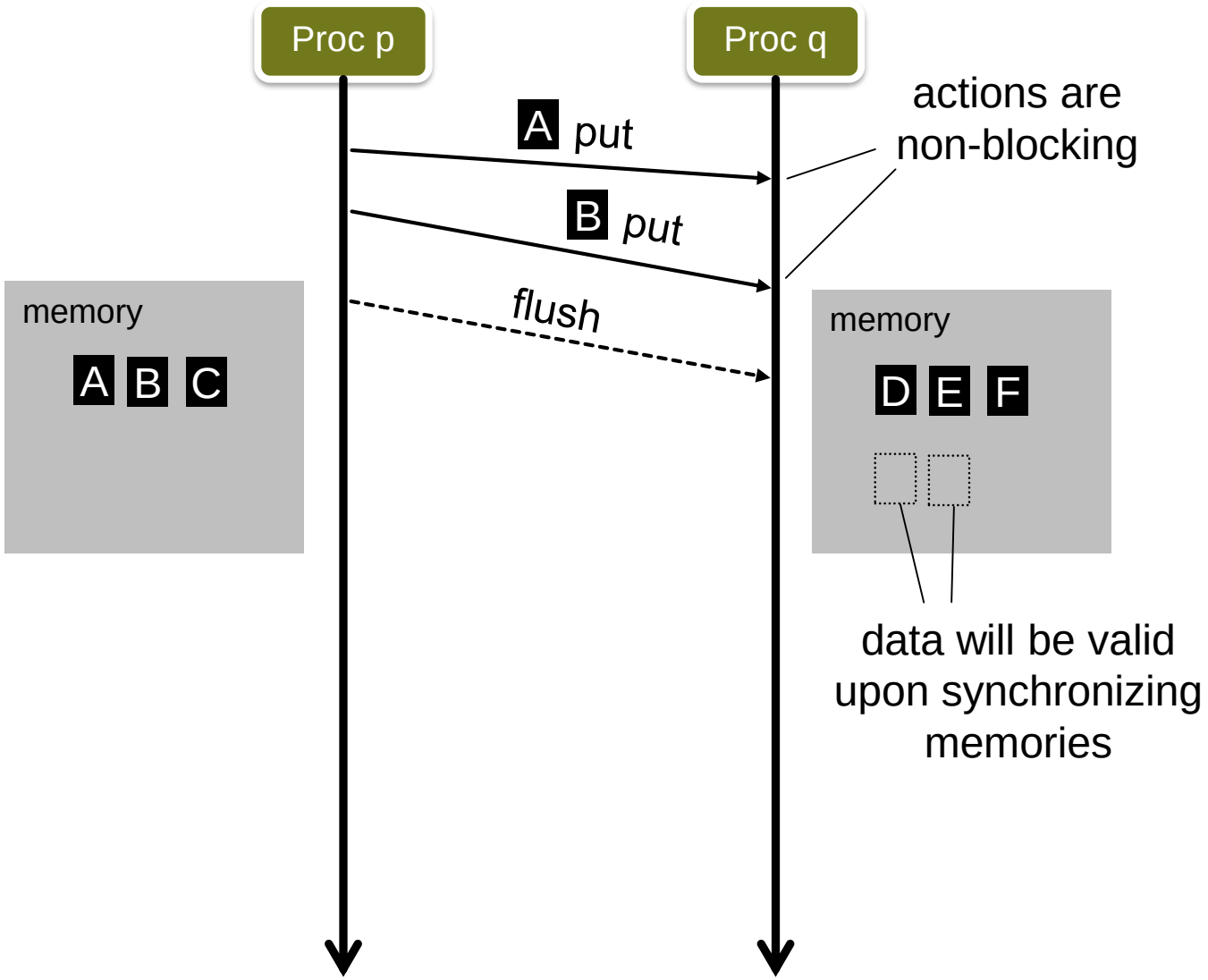
RMA: EPOCHS


Proc p
Proc q
A put
B put
actions are non-blocking
memory
A B C
memory
D E F

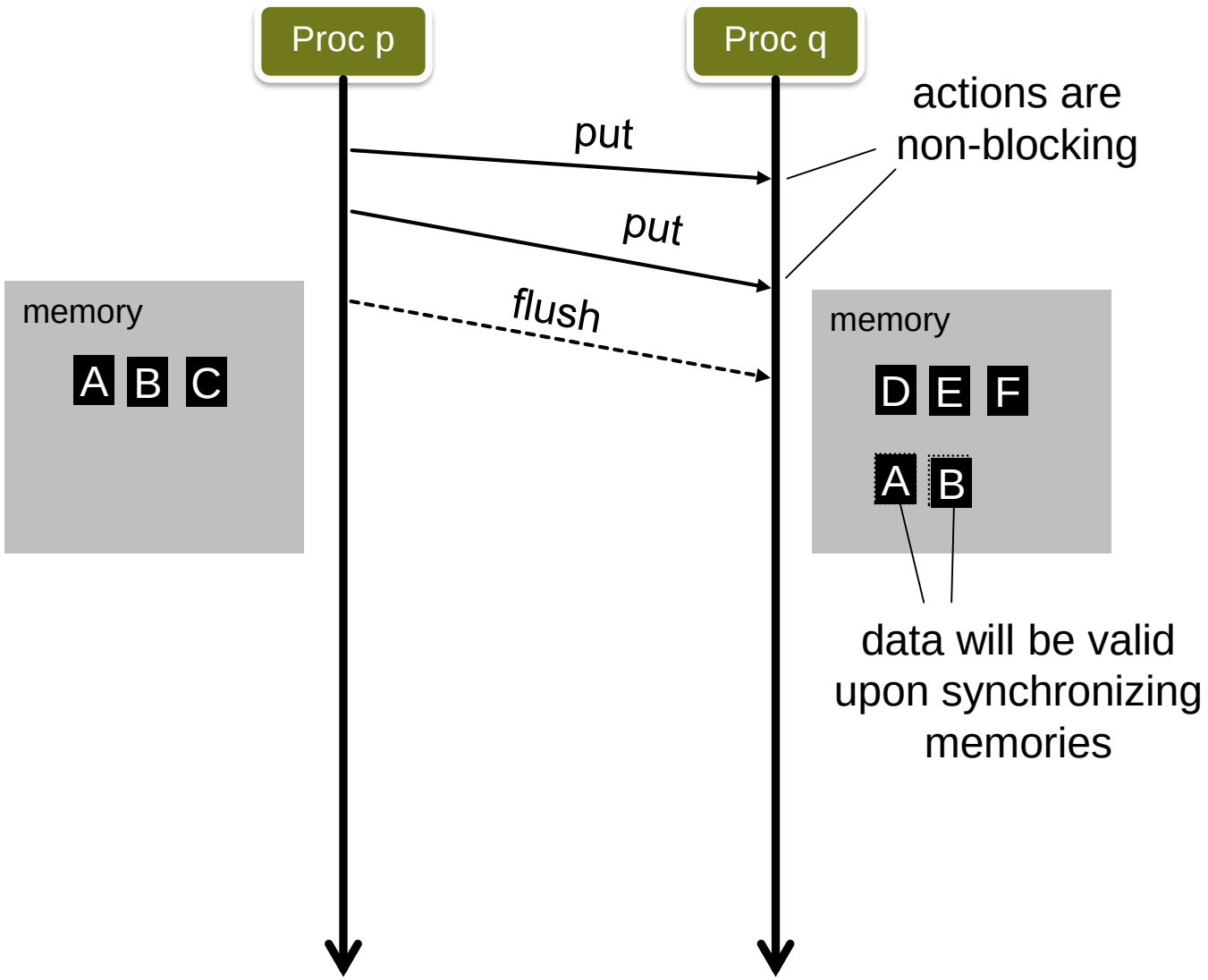
RMA: EPOCHS


Proc p
Proc q
A put
B put
actions are non-blocking
memory
A B C
memory
D E F
data will be valid upon synchronizing memories

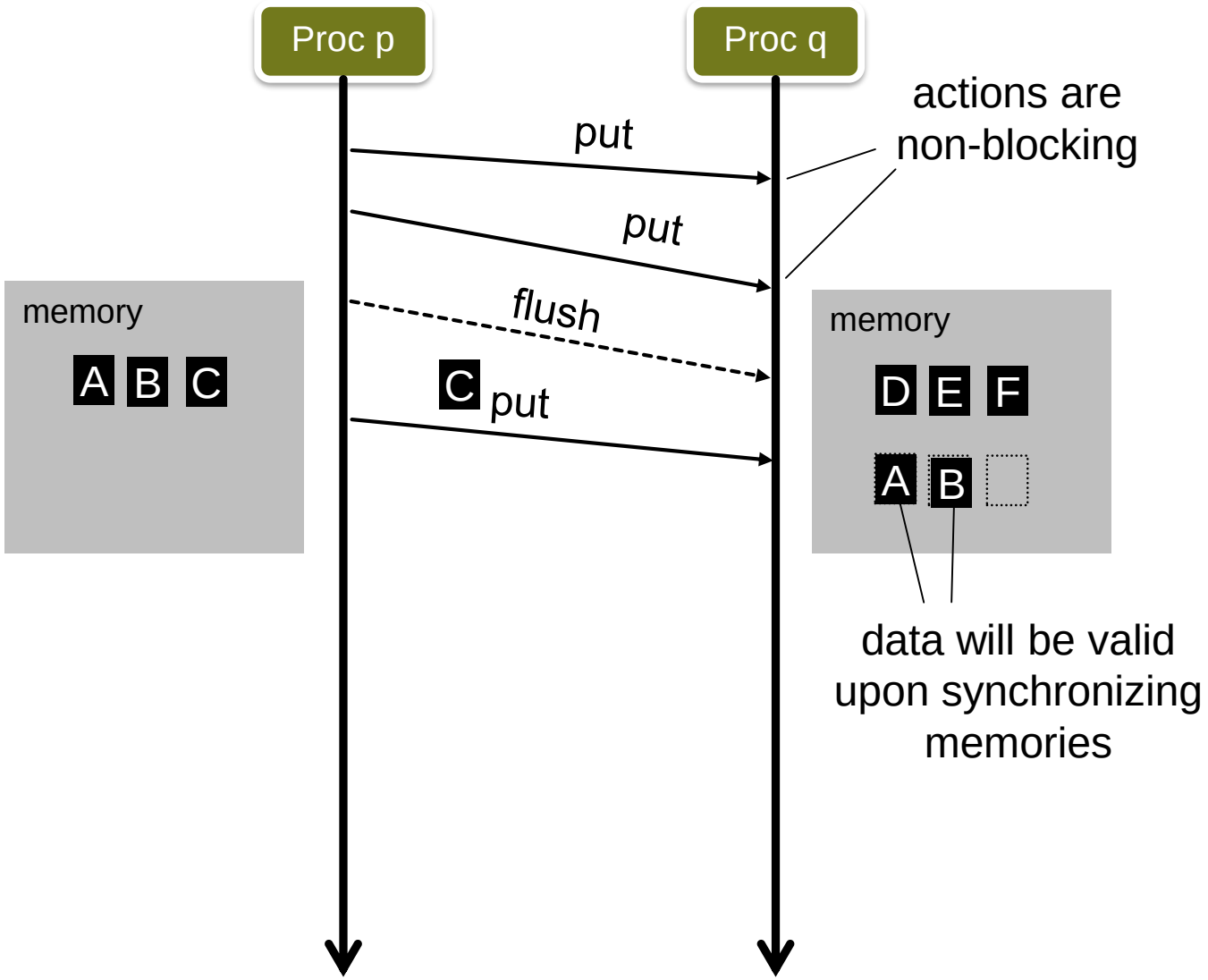
RMA: EPOCHS


Proc p
Proc q
A put
B put
flush
actions are non-blocking
memory
A B C
memory
D E F
data will be valid upon synchronizing memories

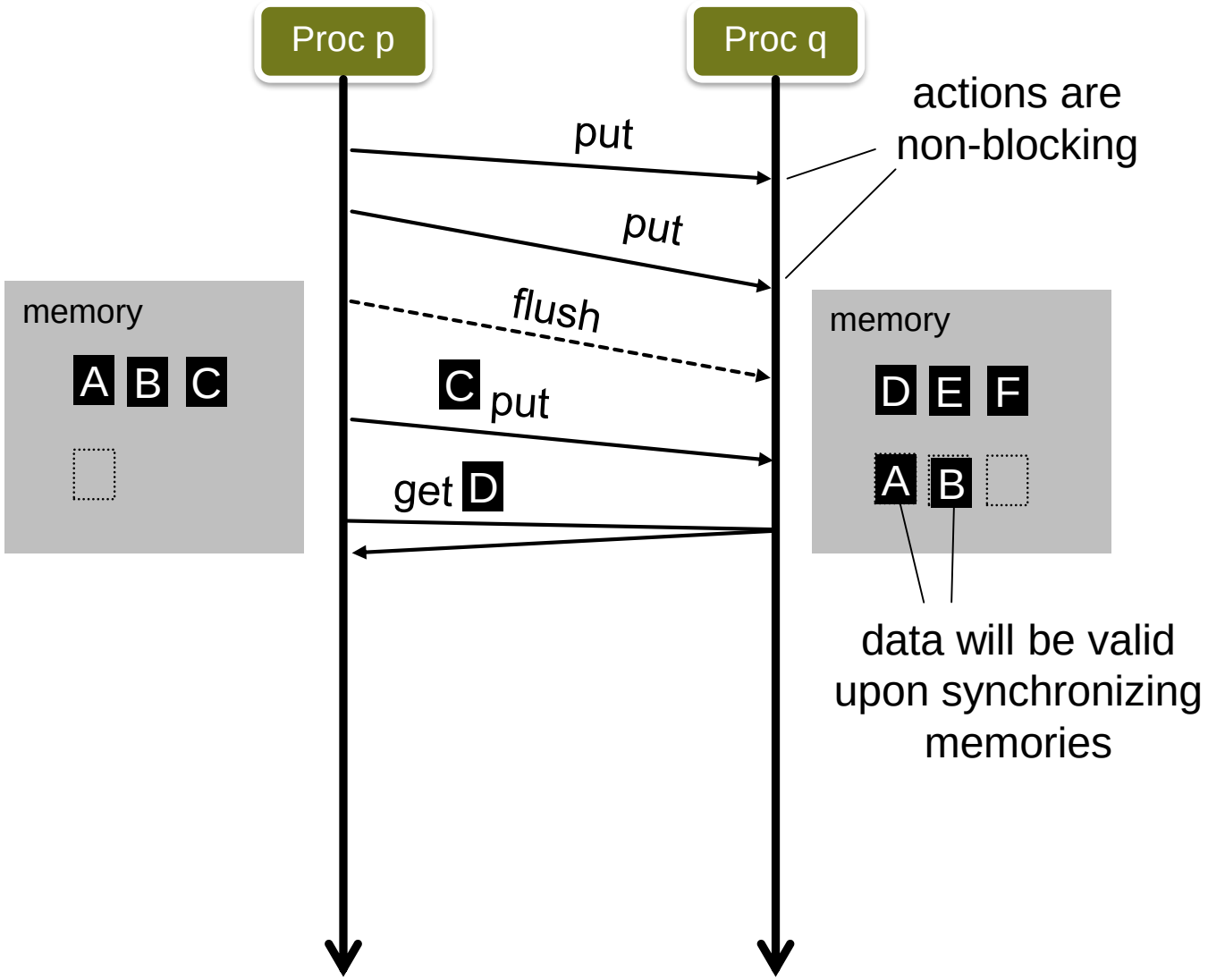
RMA: EPOCHS


Proc p
Proc q
put
put
flush
actions are non-blocking
memory
A B C
memory
D E F
A B
data will be valid upon synchronizing memories

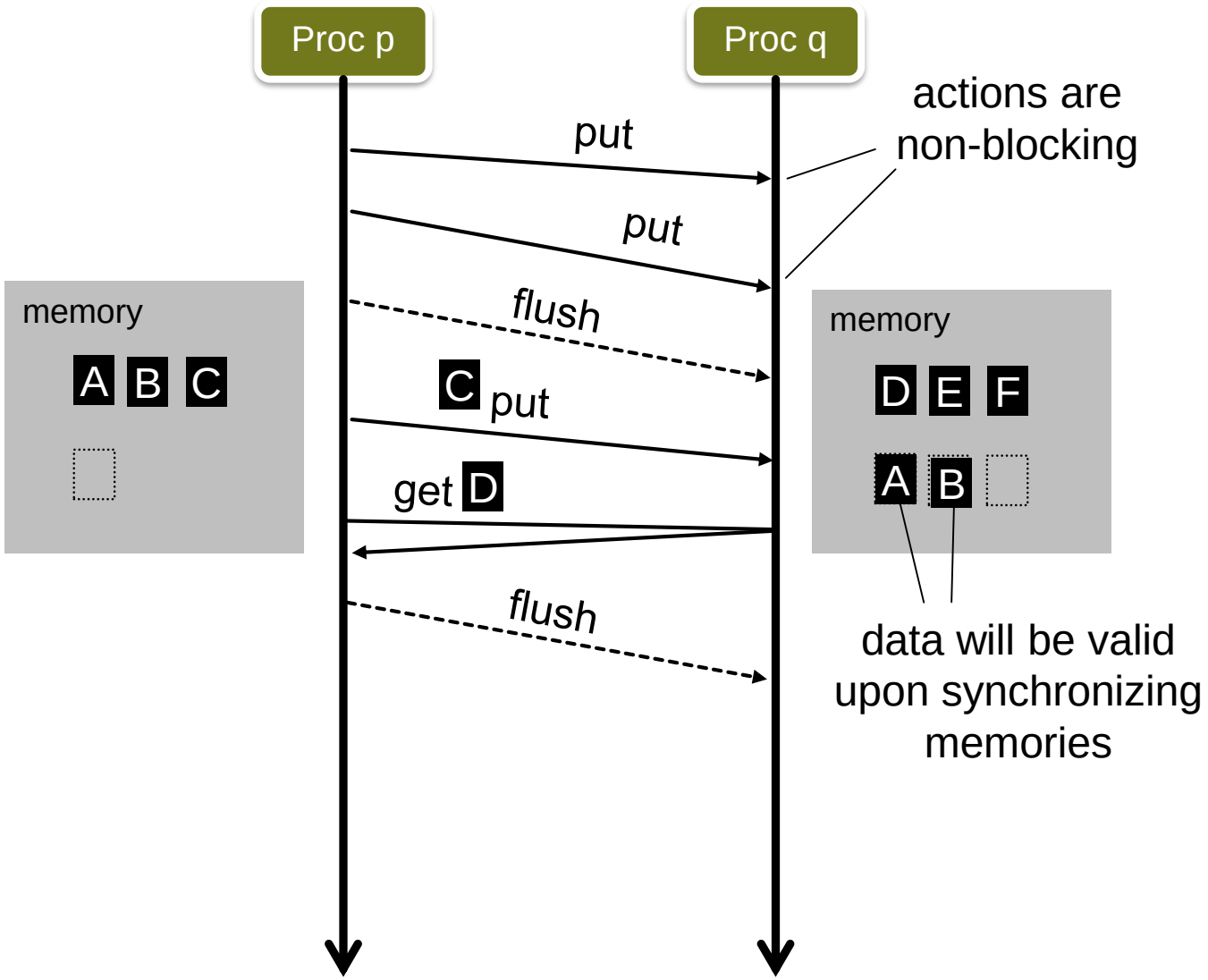
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
C put
memory
A B C
memory
D E F
A B
data will be valid upon synchronizing memories

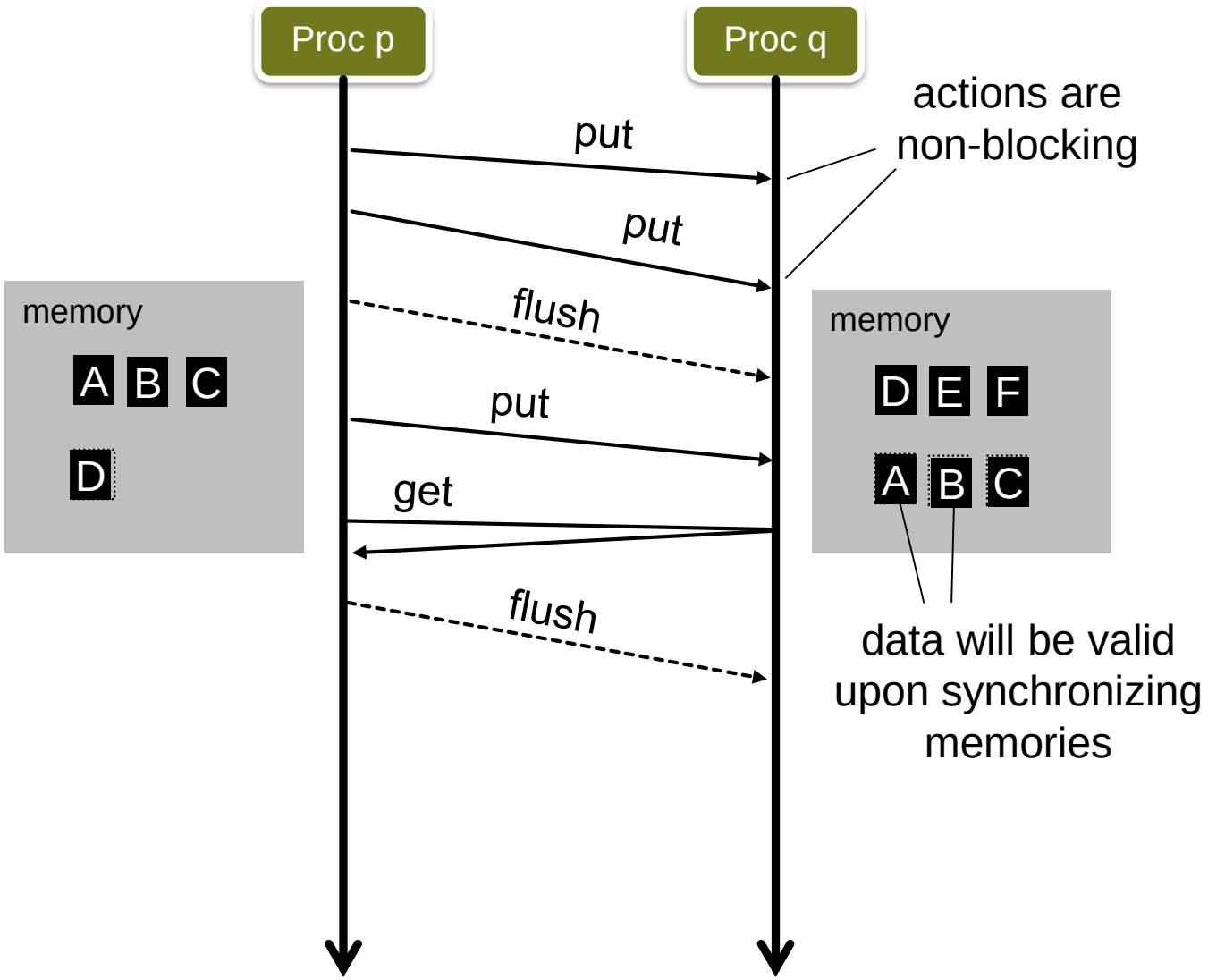
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
C put
get D
memory
A B C
memory
D E F
A B
data will be valid upon synchronizing memories

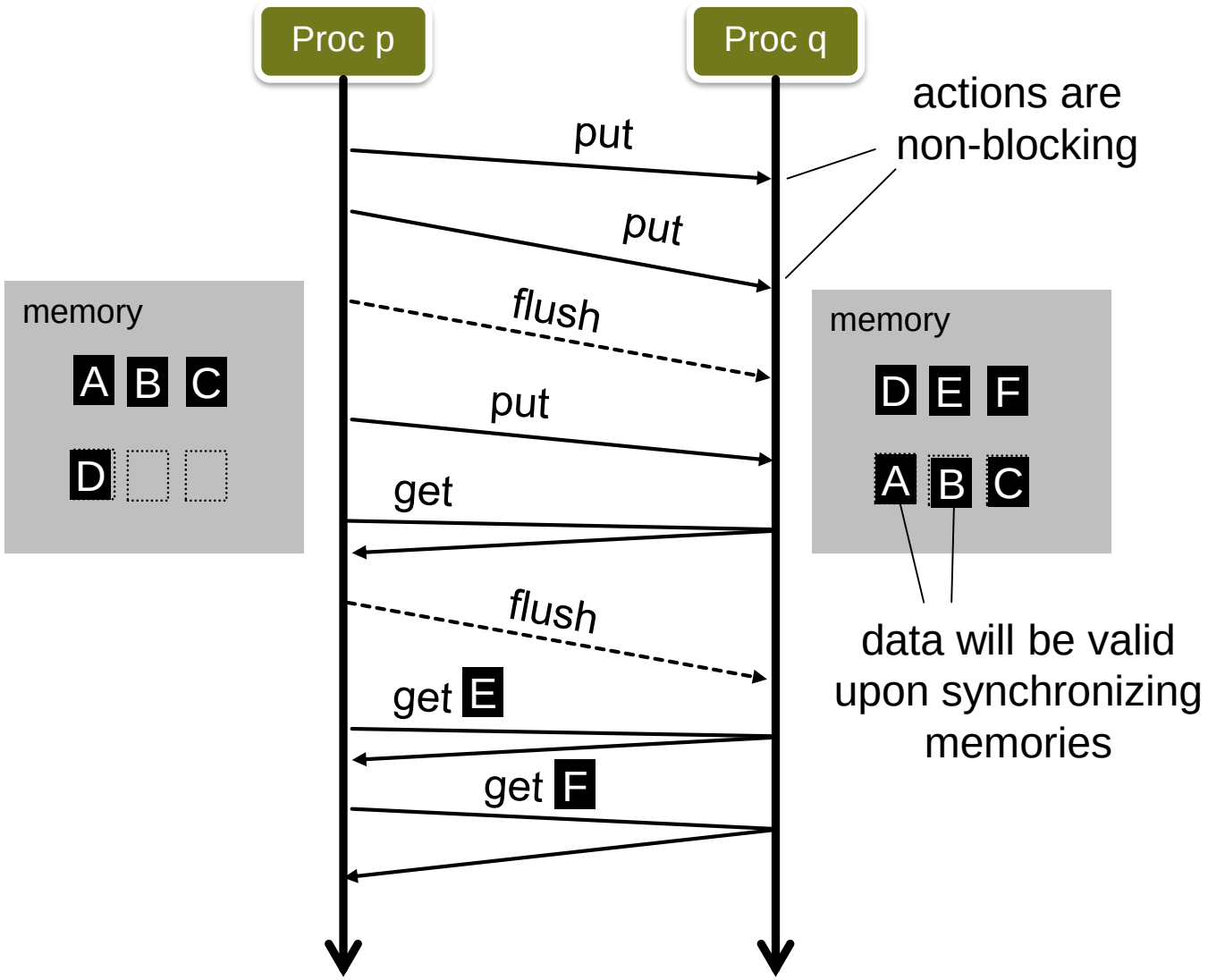
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
C put
get D
flush
memory
A B C
memory
D E F
A B
data will be valid upon synchronizing memories

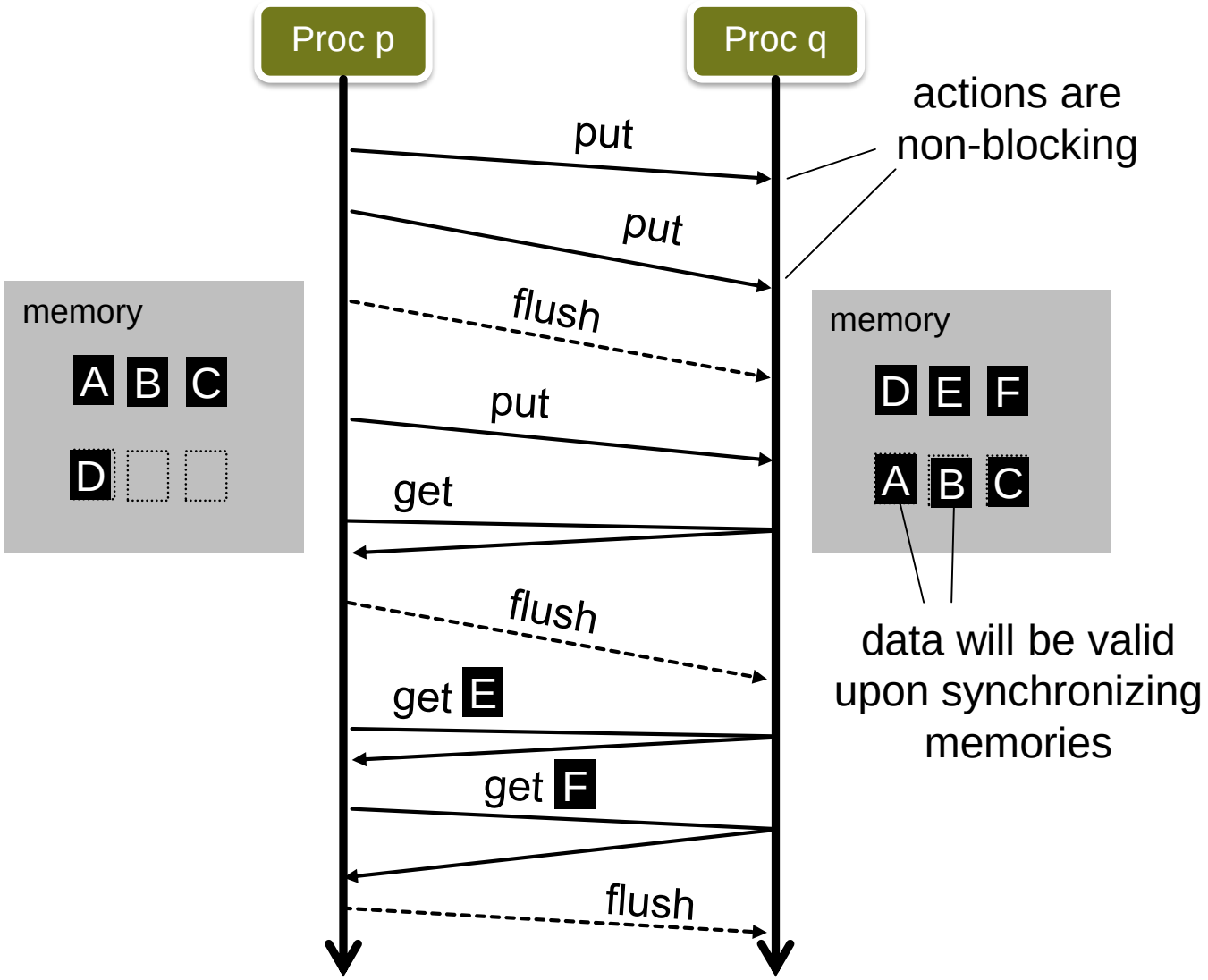
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
put
get
flush
memory
A B C
D
memory
D E F
A B C
data will be valid upon synchronizing memories

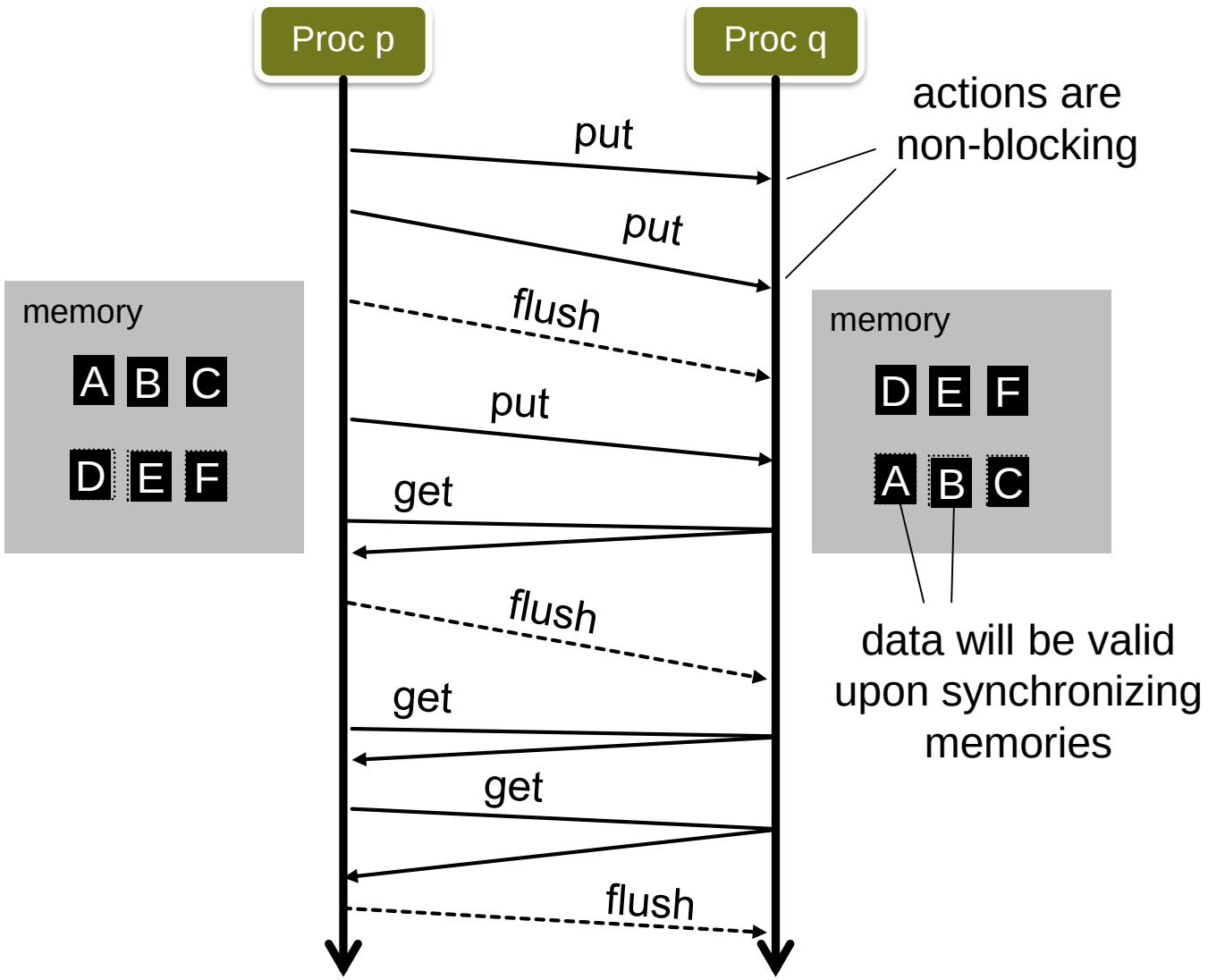
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
put
get
flush
get E
get F
memory
A B C
D
memory
D E F
A B C
data will be valid upon synchronizing memories

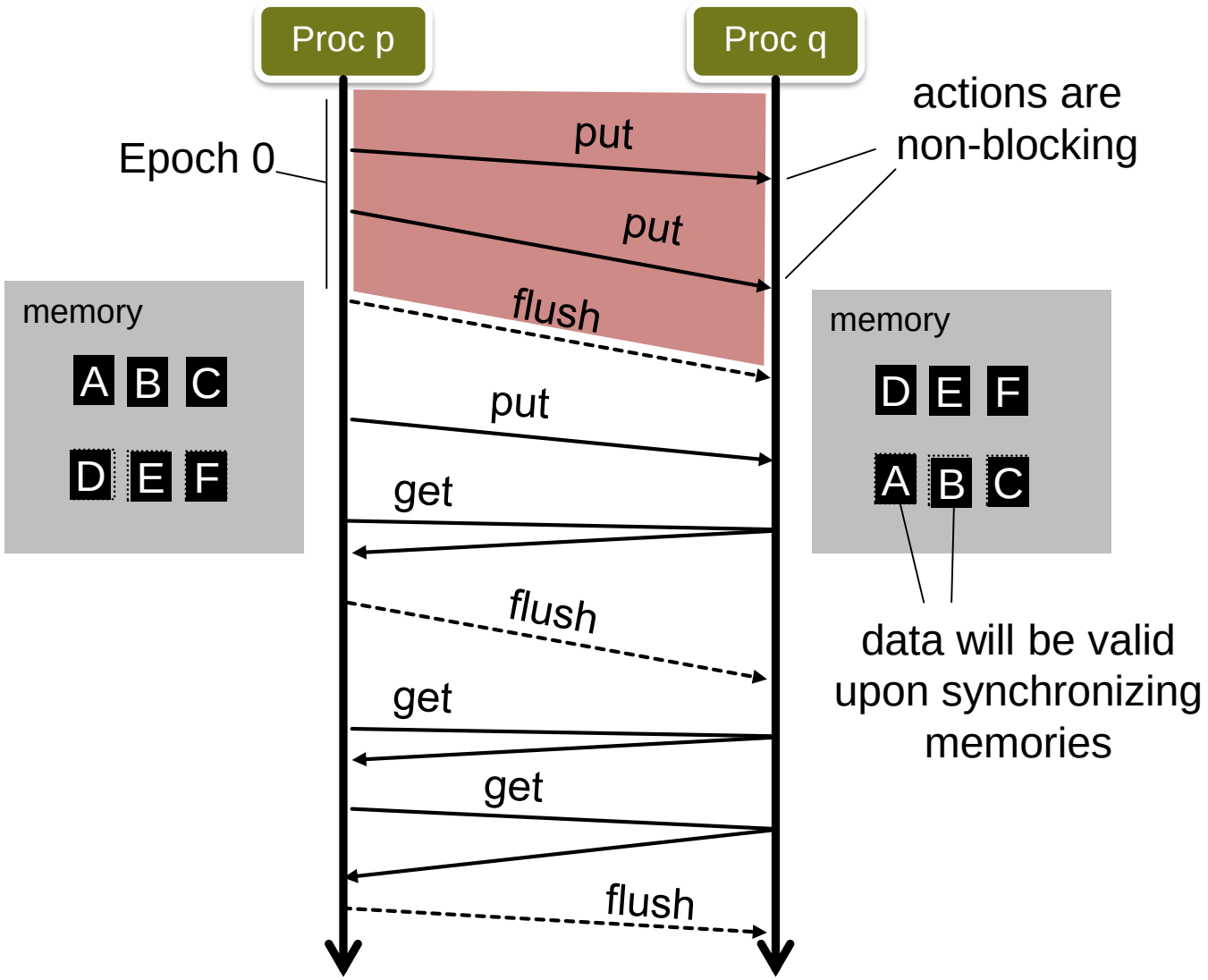
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
put
get
flush
get E
get F
flush
memory
A B C
D
memory
D E F
A B C
data will be valid upon synchronizing memories

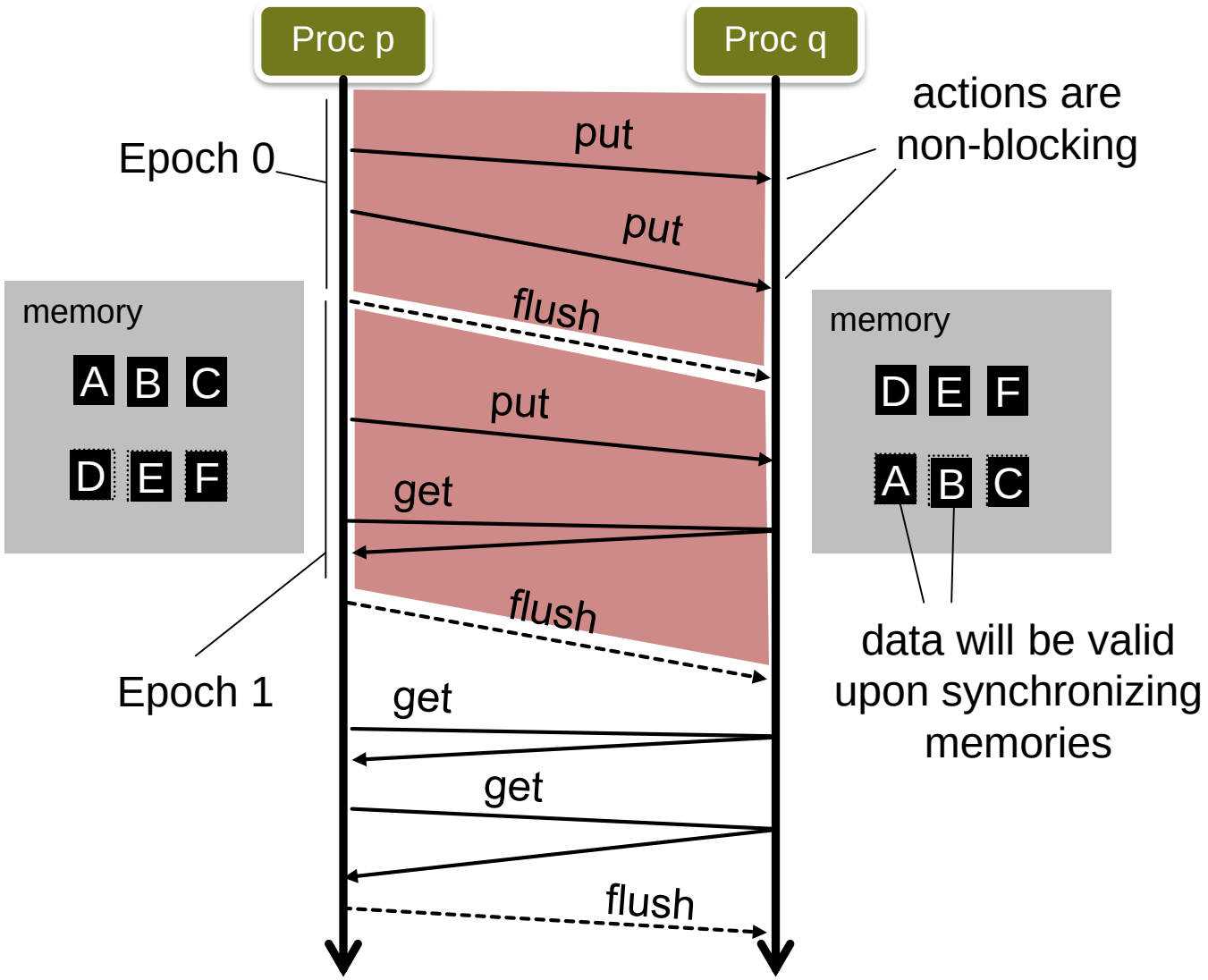
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
put
put
flush
put
get
flush
get
get
flush
memory
A B C
D E F
memory
D E F
A B C
data will be valid upon synchronizing memories

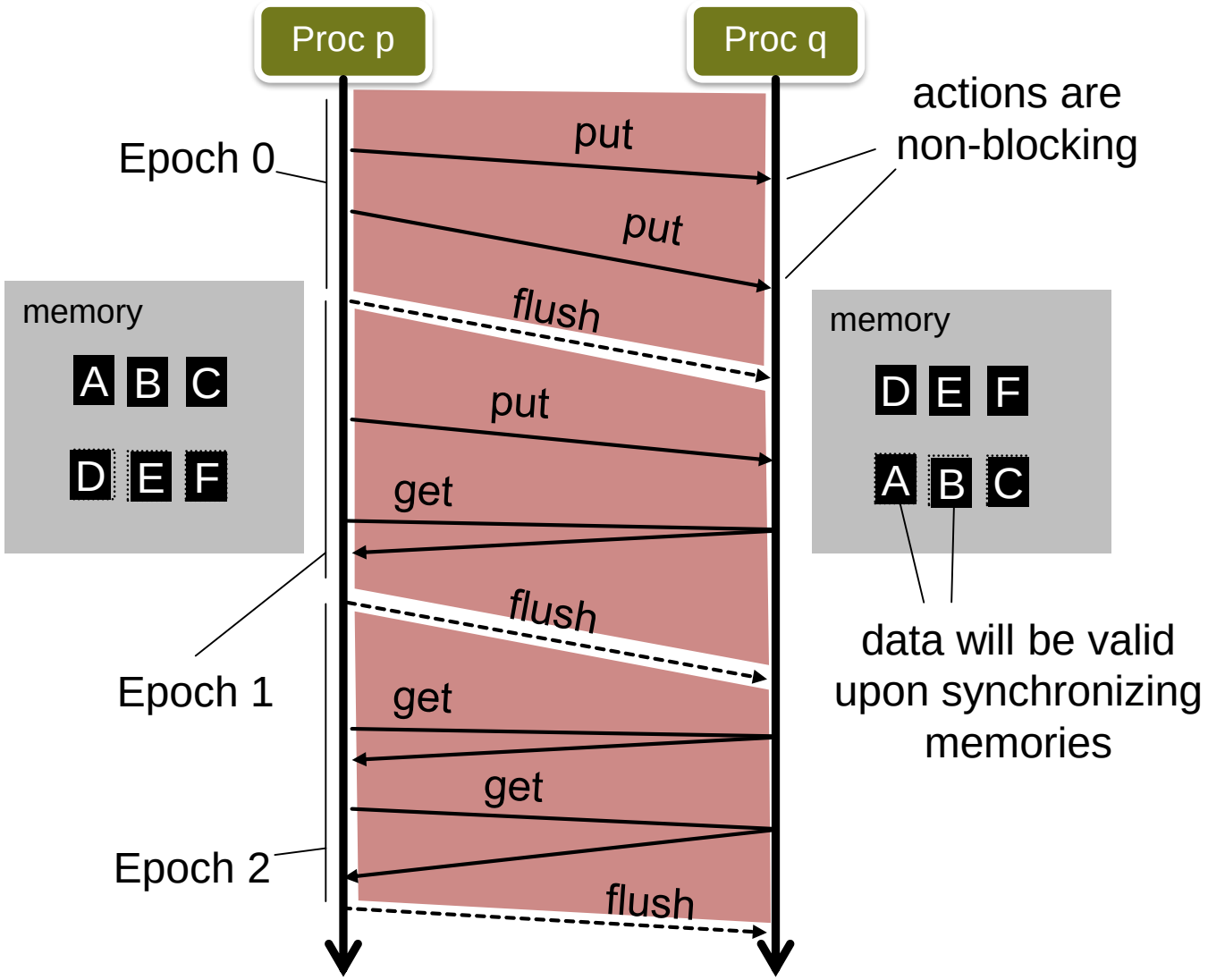
RMA: EPOCHS


Proc p
Proc q
Epoch 0
put
put
flush
actions are non-blocking
memory
A B C
D E F
memory
D E F
A B C
put
get
flush
get
get
flush
data will be valid upon synchronizing memories

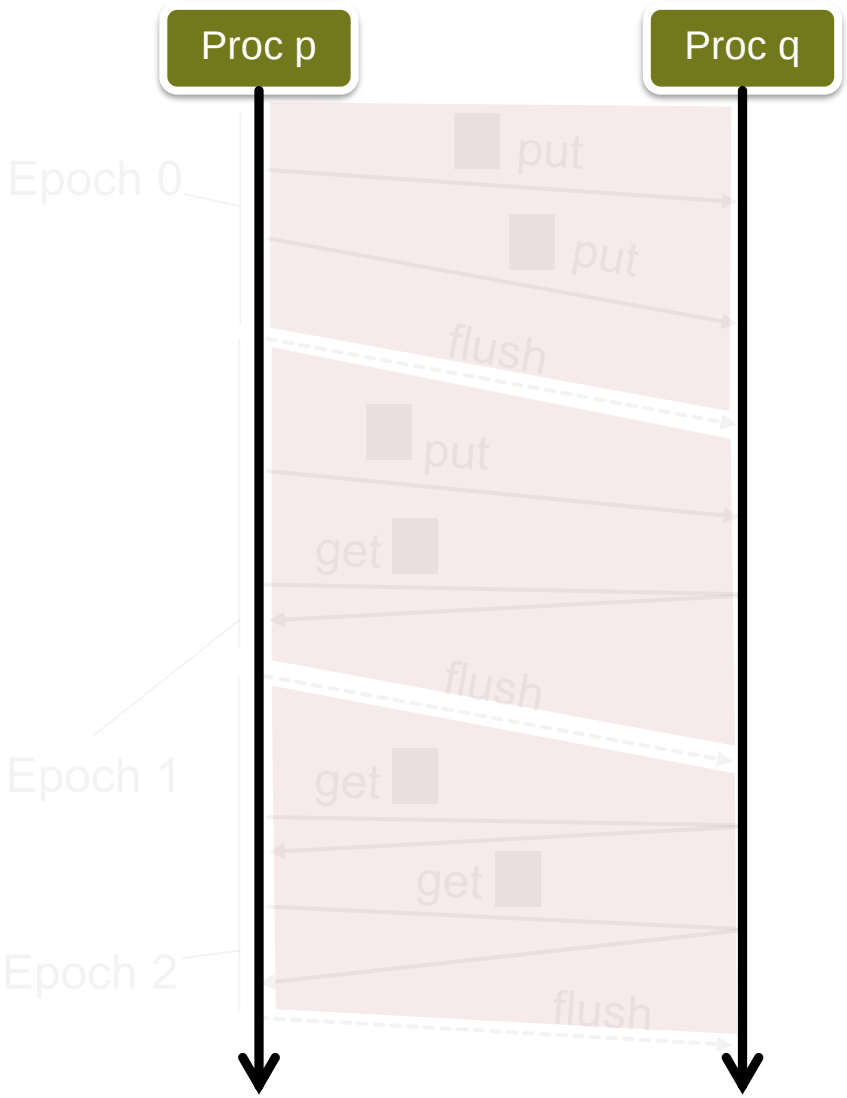
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
Epoch 0
put
put
flush
memory
A B C
D E F
put
get
flush
Epoch 1
get
get
flush
memory
D E F
A B C
data will be valid upon synchronizing memories

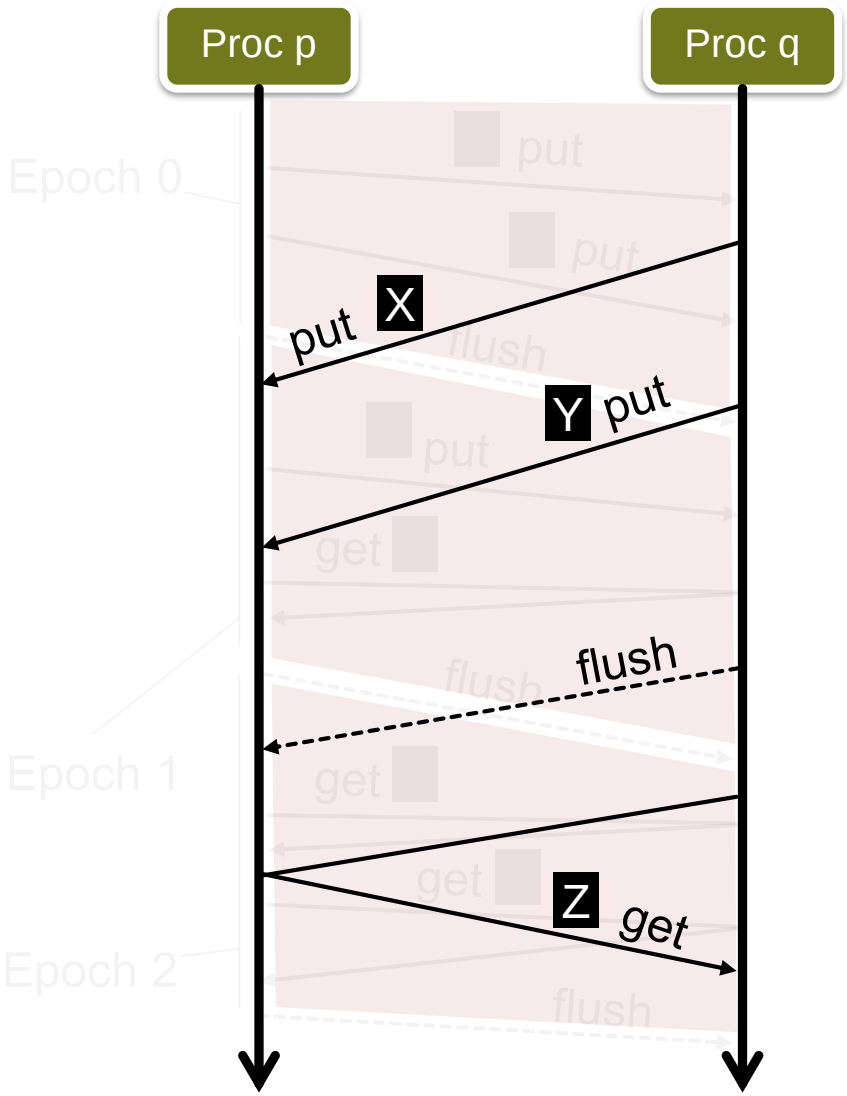
RMA: EPOCHS


Proc p
Proc q
actions are non-blocking
Epoch 0
put
put
flush
memory
A B C
D E F
put
get
memory
D E F
A B C
flush
Epoch 1
get
get
Epoch 2
flush
data will be valid upon synchronizing memories

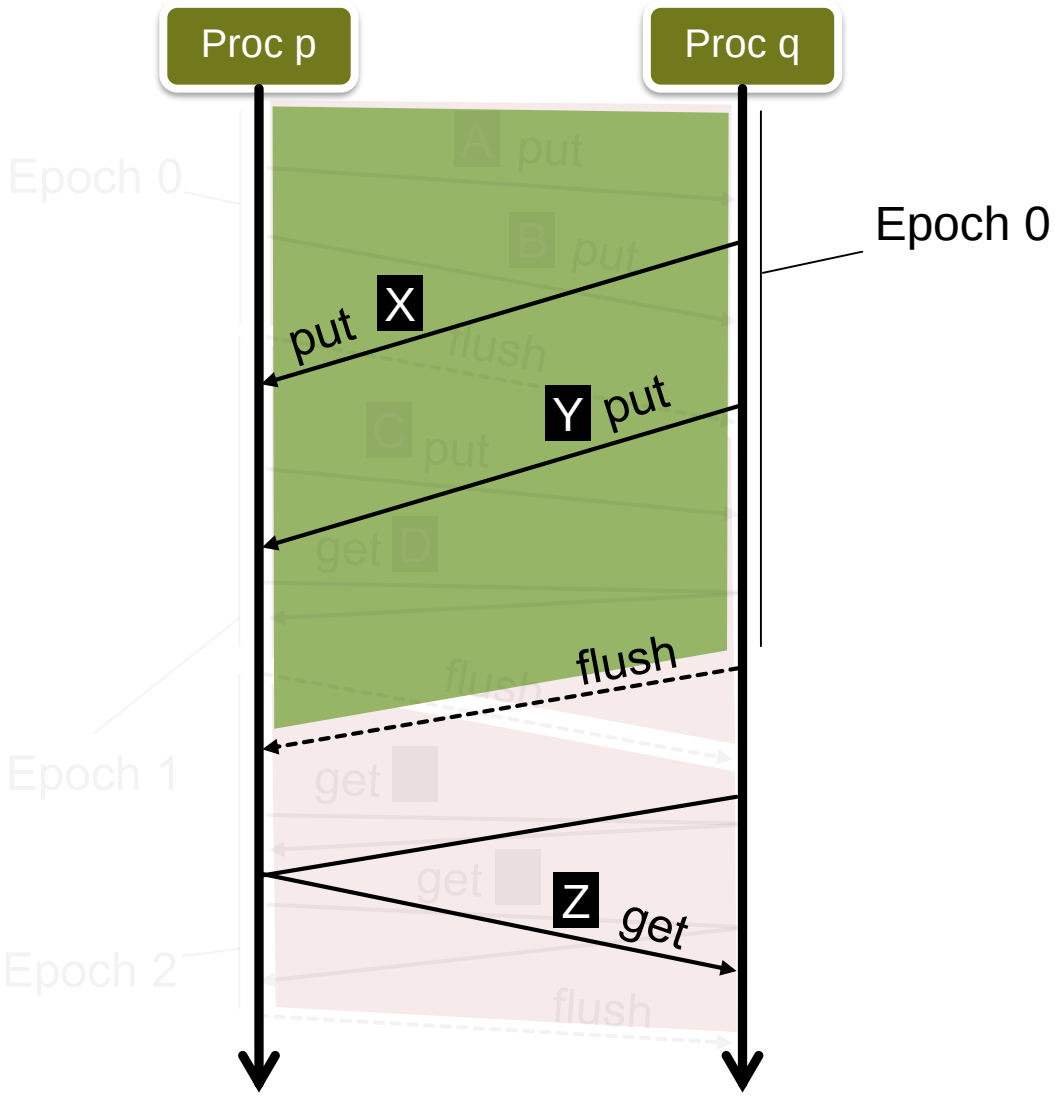
RMA: EPOCHS


Proc p
Proc q
Epoch 0
put
put
flush
put
get
Epoch 1
flush
get
get
Epoch 2
flush

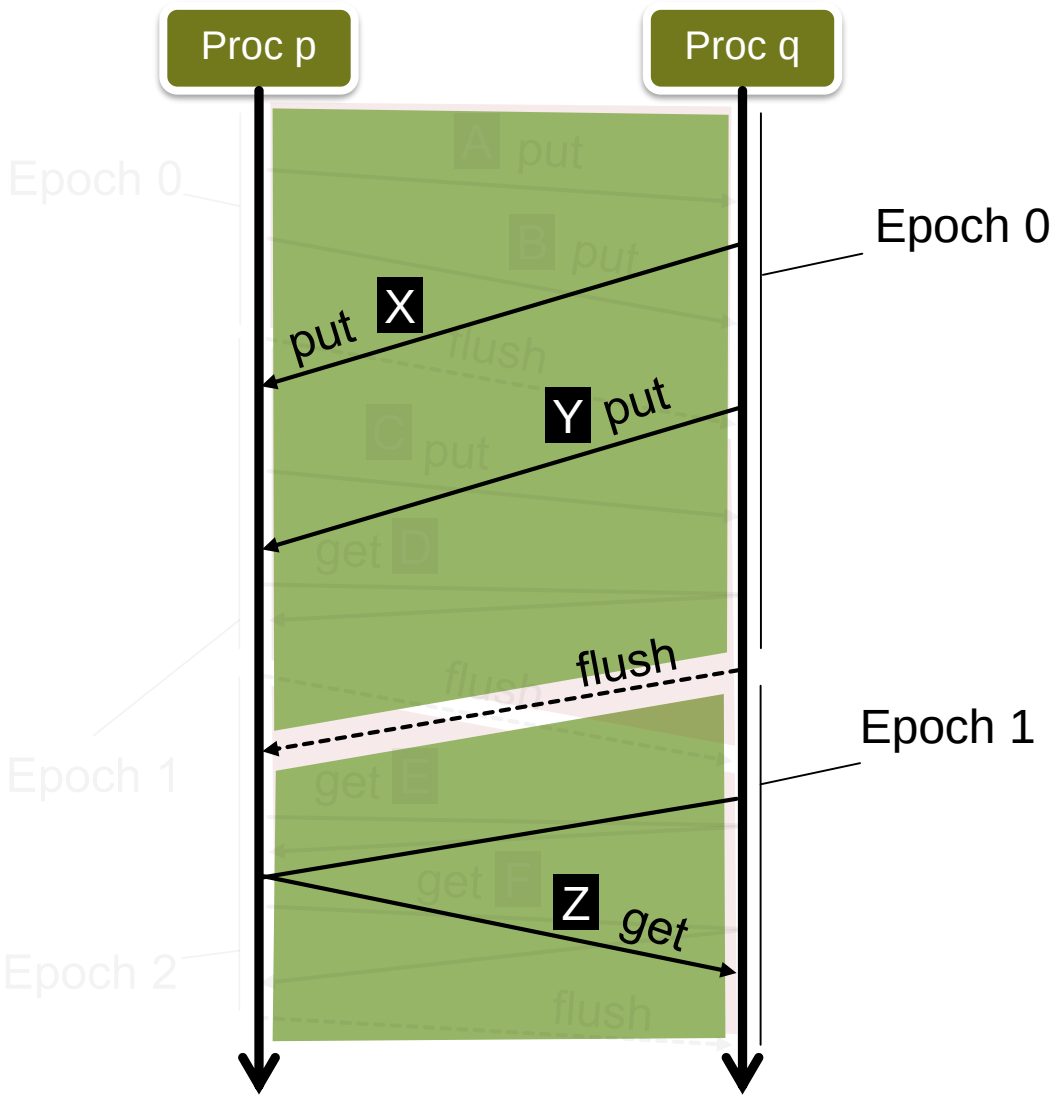
RMA: EPOCHS


Proc p
Proc q
Epoch 0
Epoch 1
Epoch 2
put
X
Y
put
flush
get
Z
get
flush

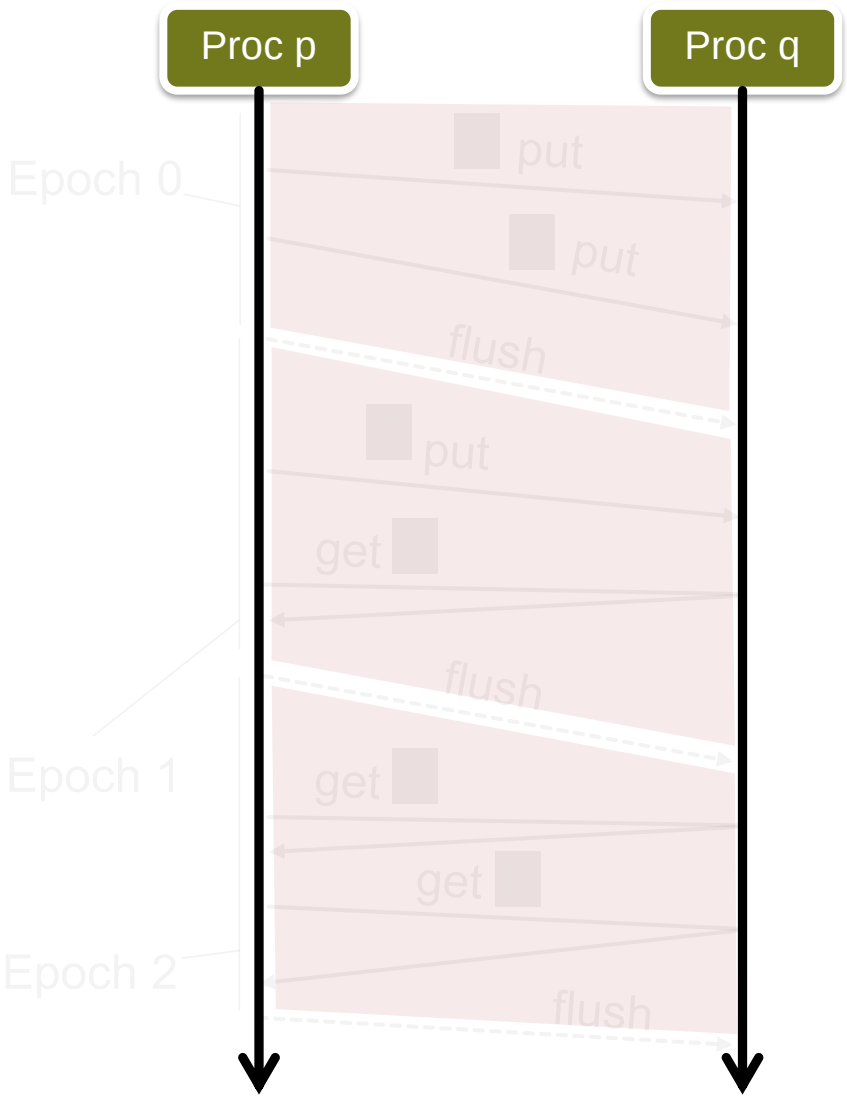
RMA: EPOCHS


Proc p
Proc q
Epoch 0
put X
Y put
flush
Epoch 1
get
Z get
Epoch 2
flush

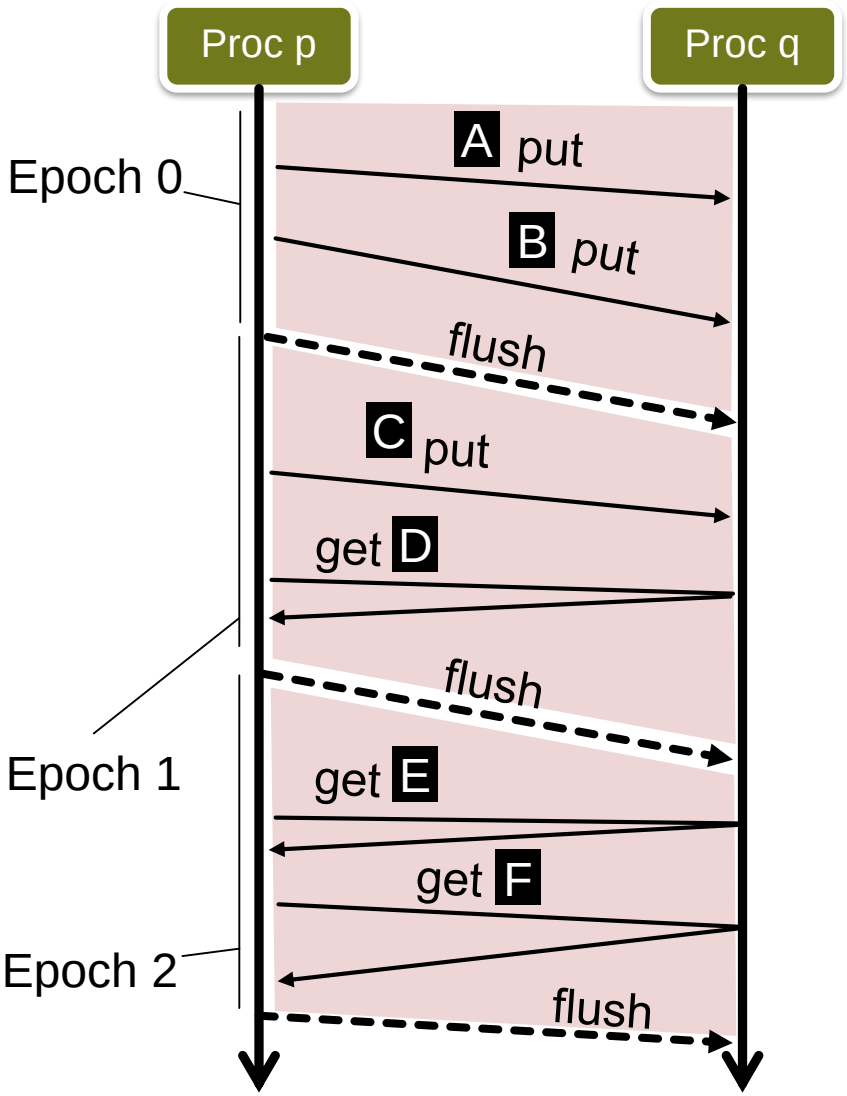
RMA: EPOCHS



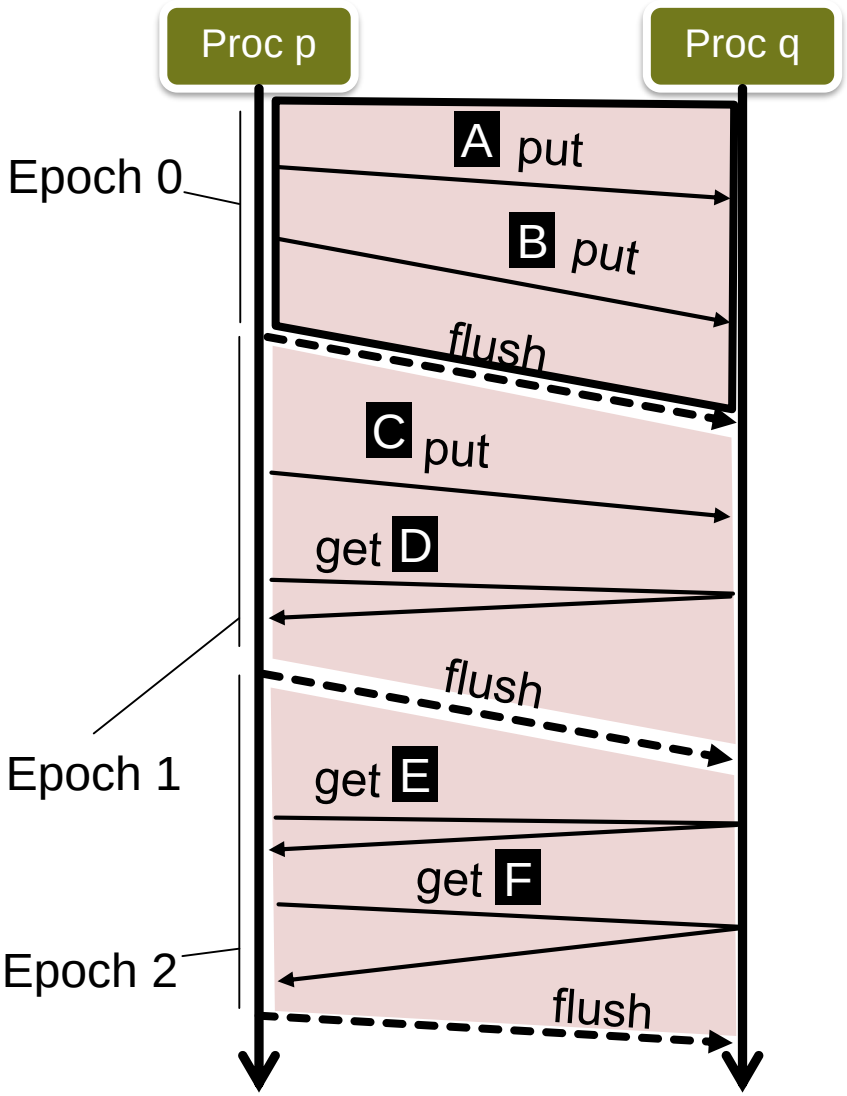
RMA: EPOCHS


Proc p
Proc q
Epoch 0
put
put
flush
put
get
Epoch 1
flush
get
get
Epoch 2
flush

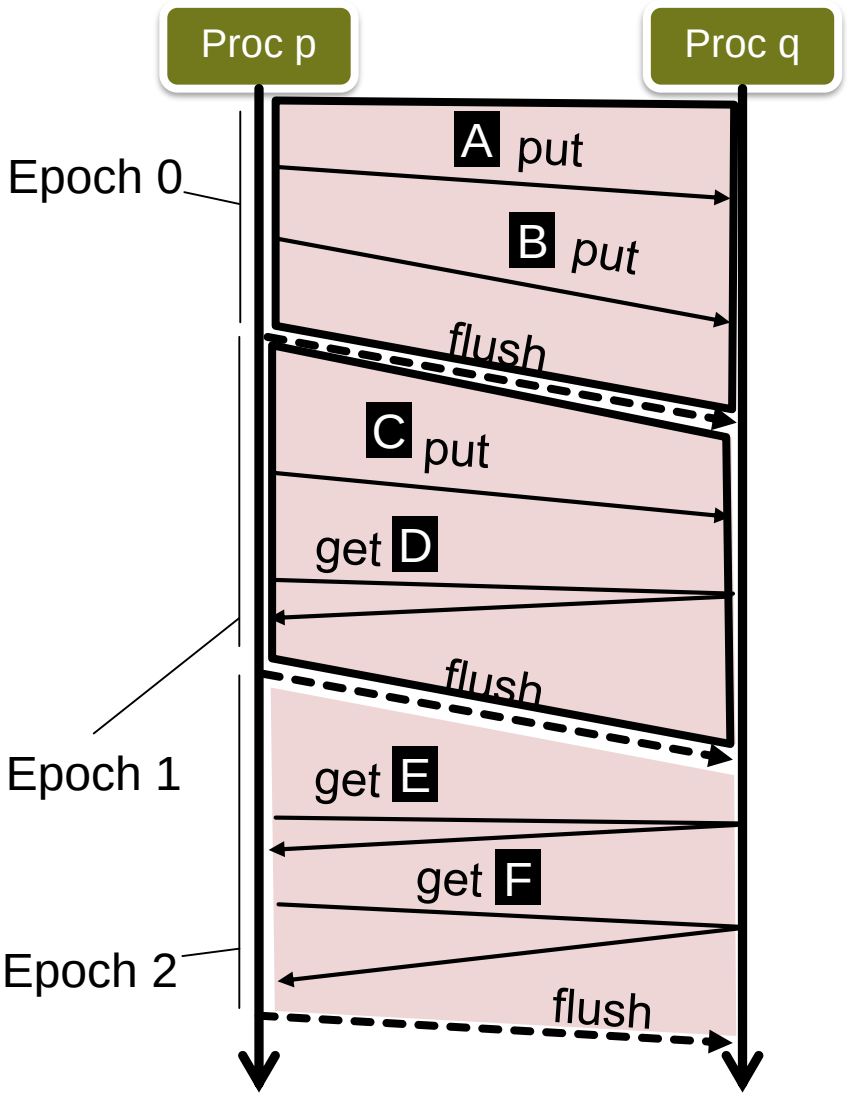
RMA: The Consistency Order $\xrightarrow{co}$



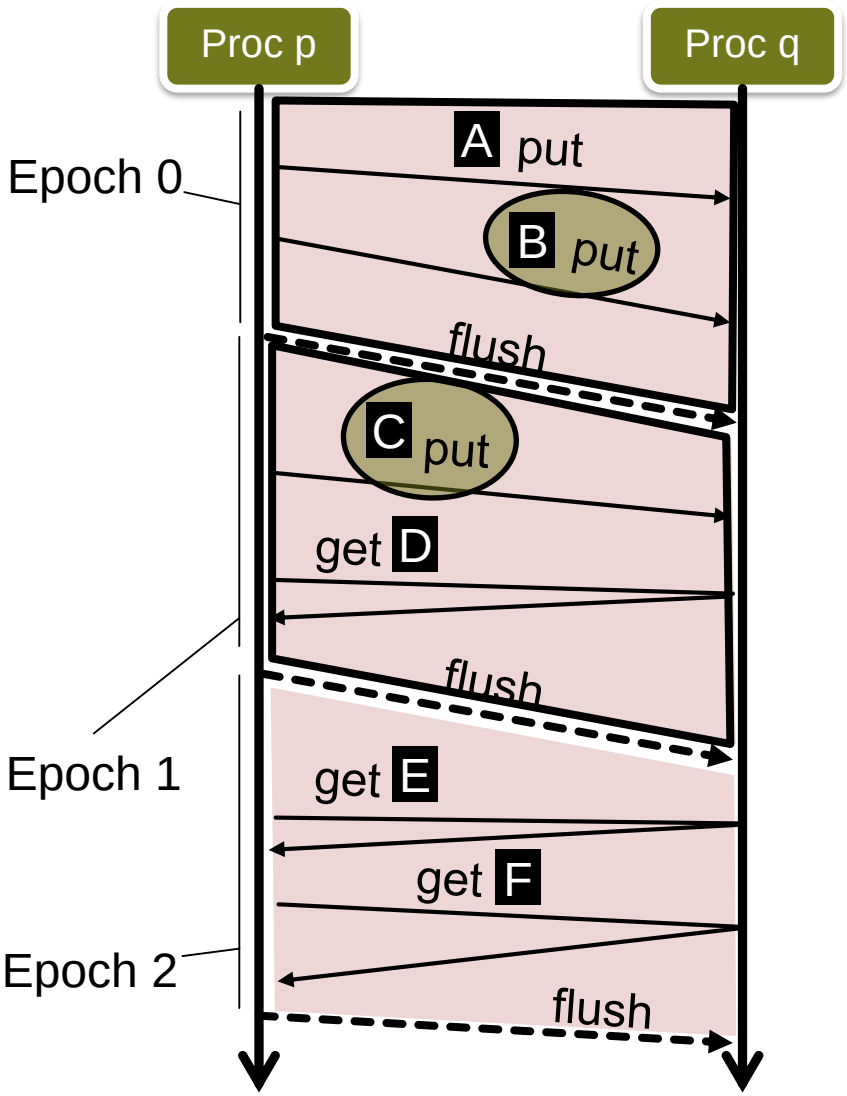
RMA: The Consistency Order $\xrightarrow{co}$



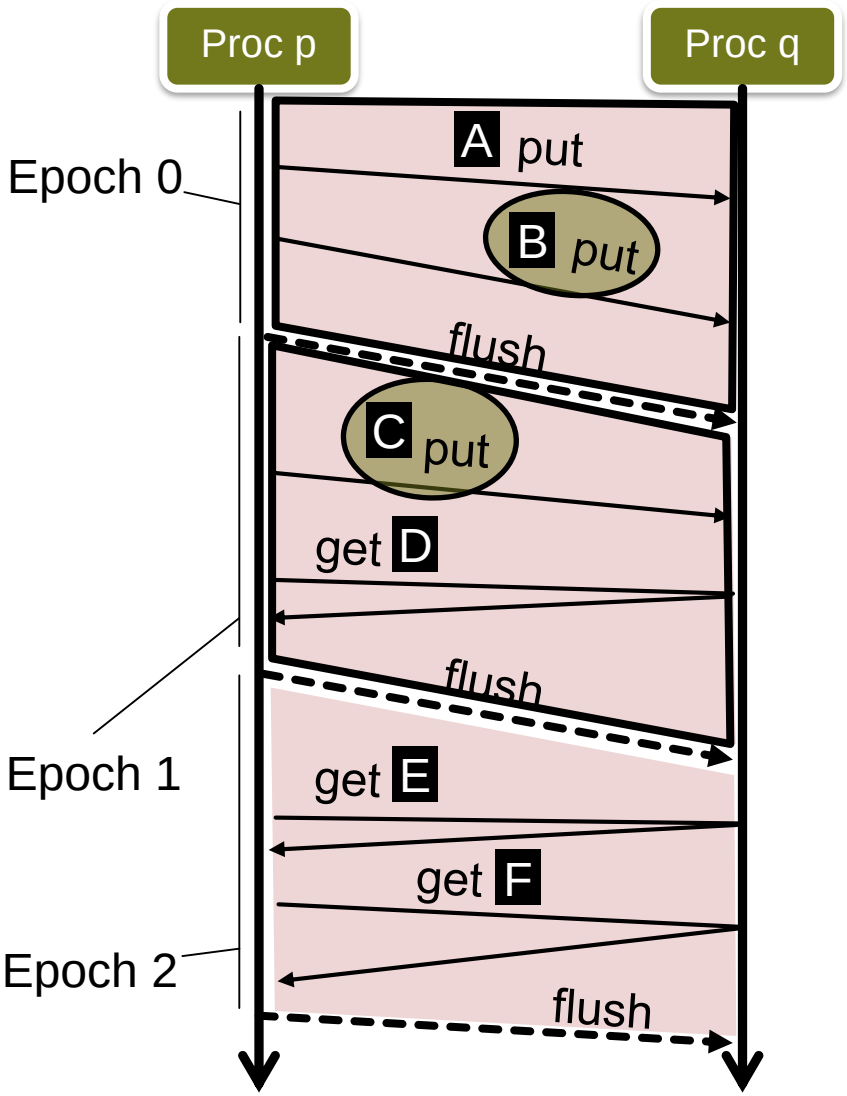
RMA: The Consistency Order $\xrightarrow{co}$



RMA: The Consistency Order $\xrightarrow{co}$

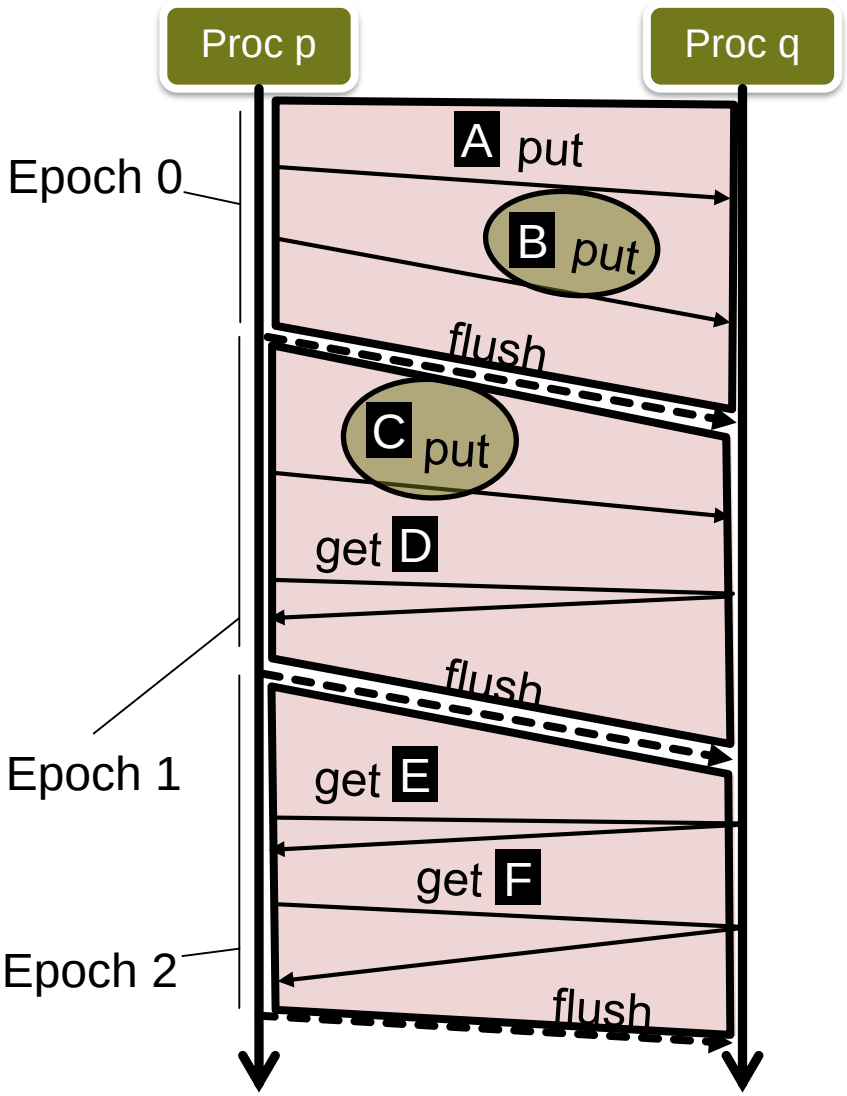


RMA: The Consistency Order $\xrightarrow{co}$



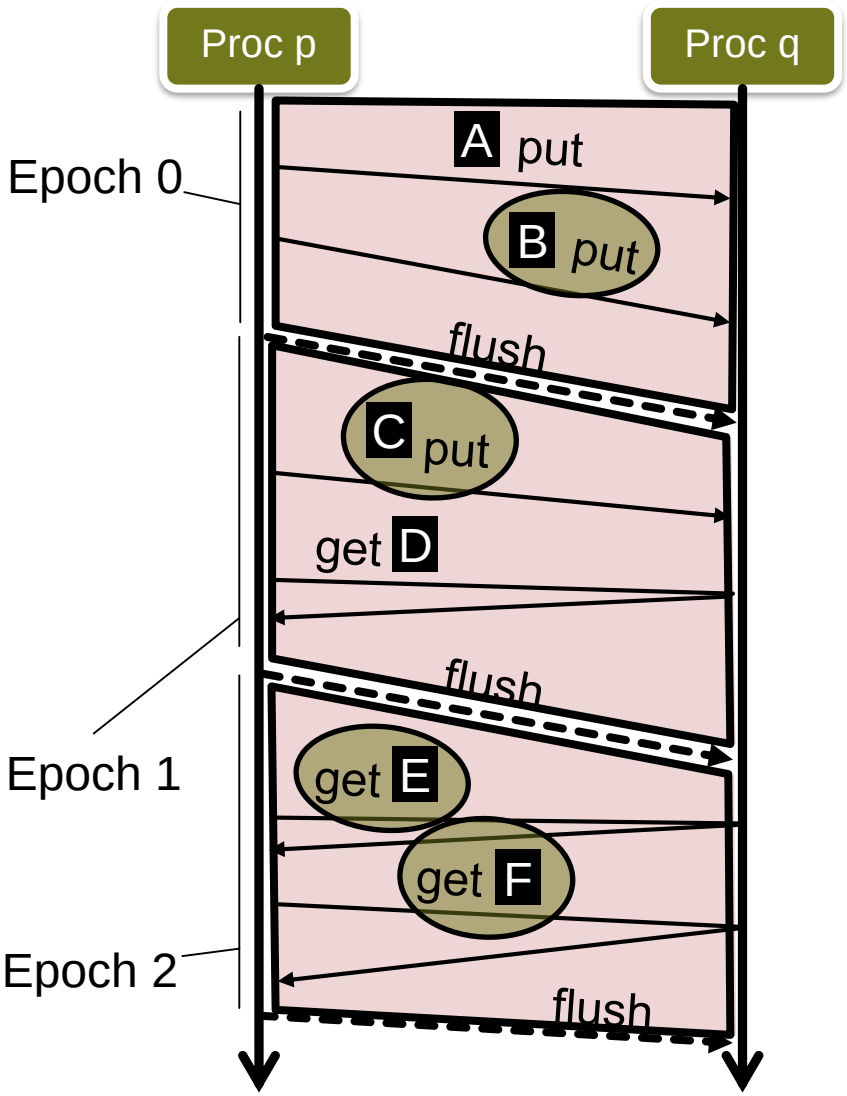
B put $\xrightarrow{co}$ C put

RMA: The Consistency Order $\xrightarrow{co}$



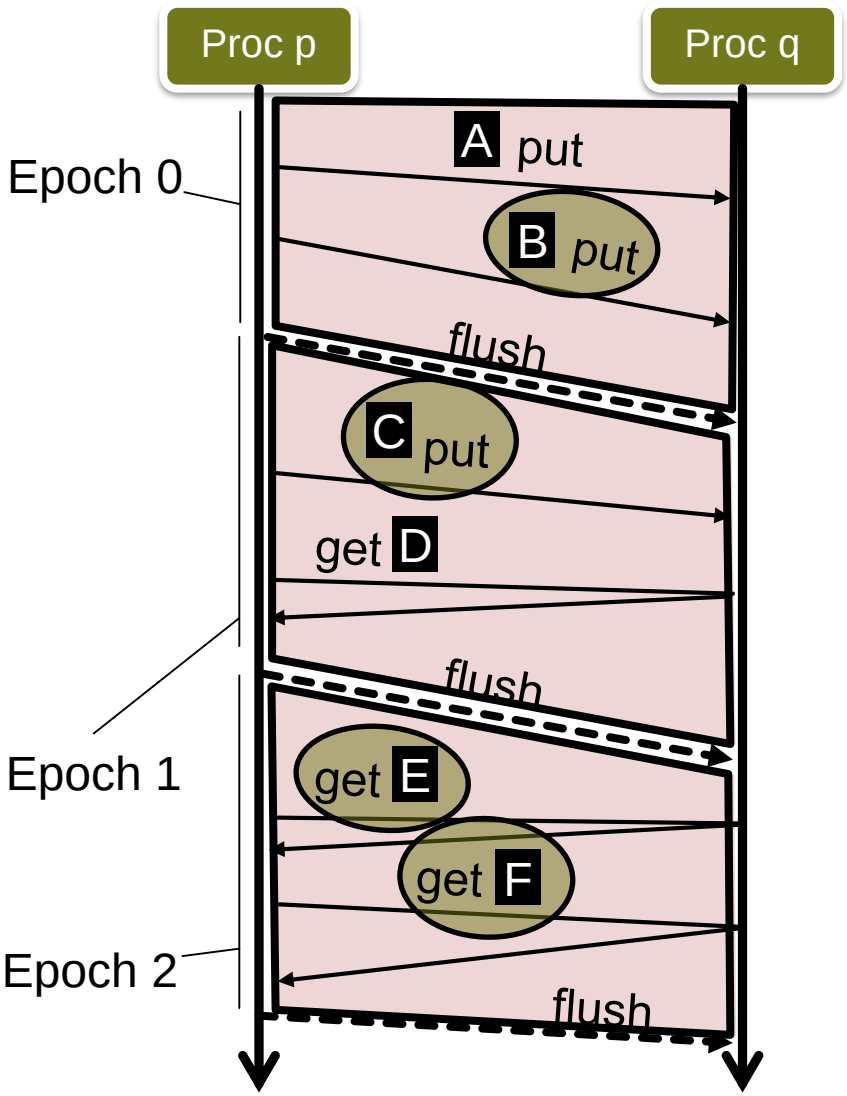
B put $\xrightarrow{co}$ C put

RMA: The Consistency Order $\xrightarrow{co}$



B put $\xrightarrow{co}$ C put

RMA: The Consistency Order $\xrightarrow{co}$

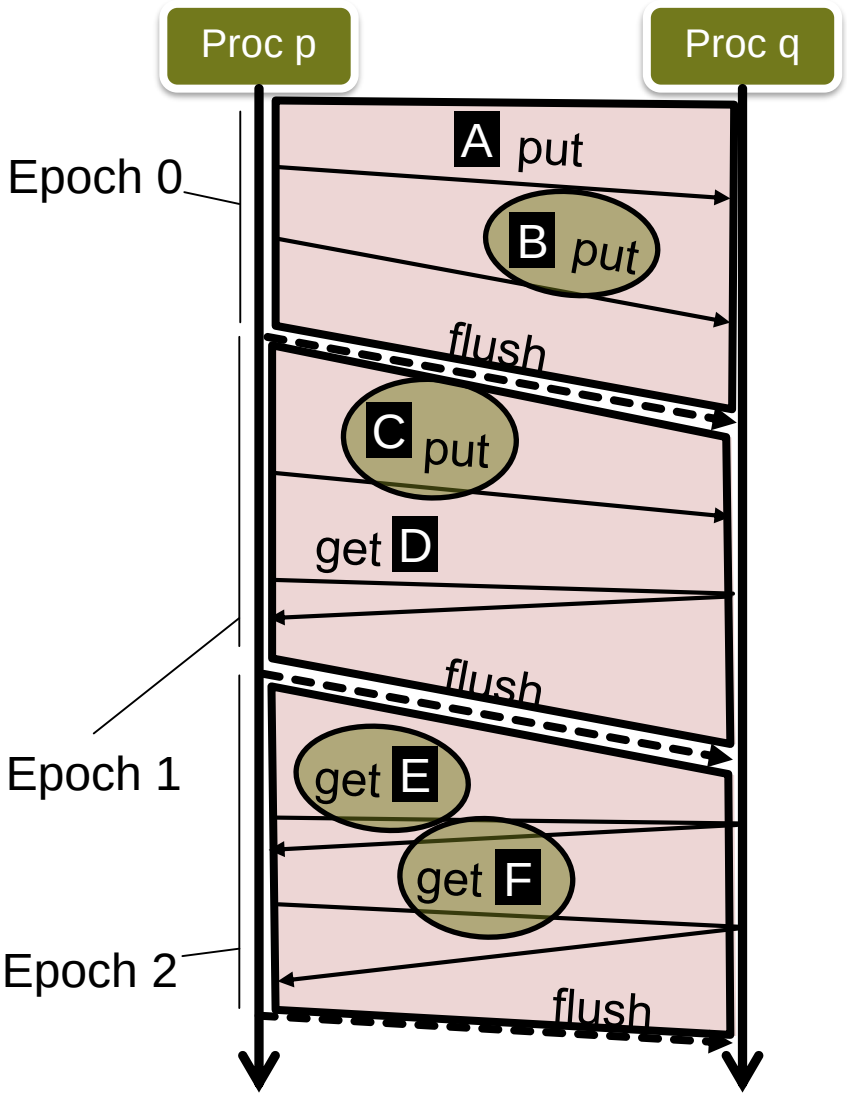


B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!



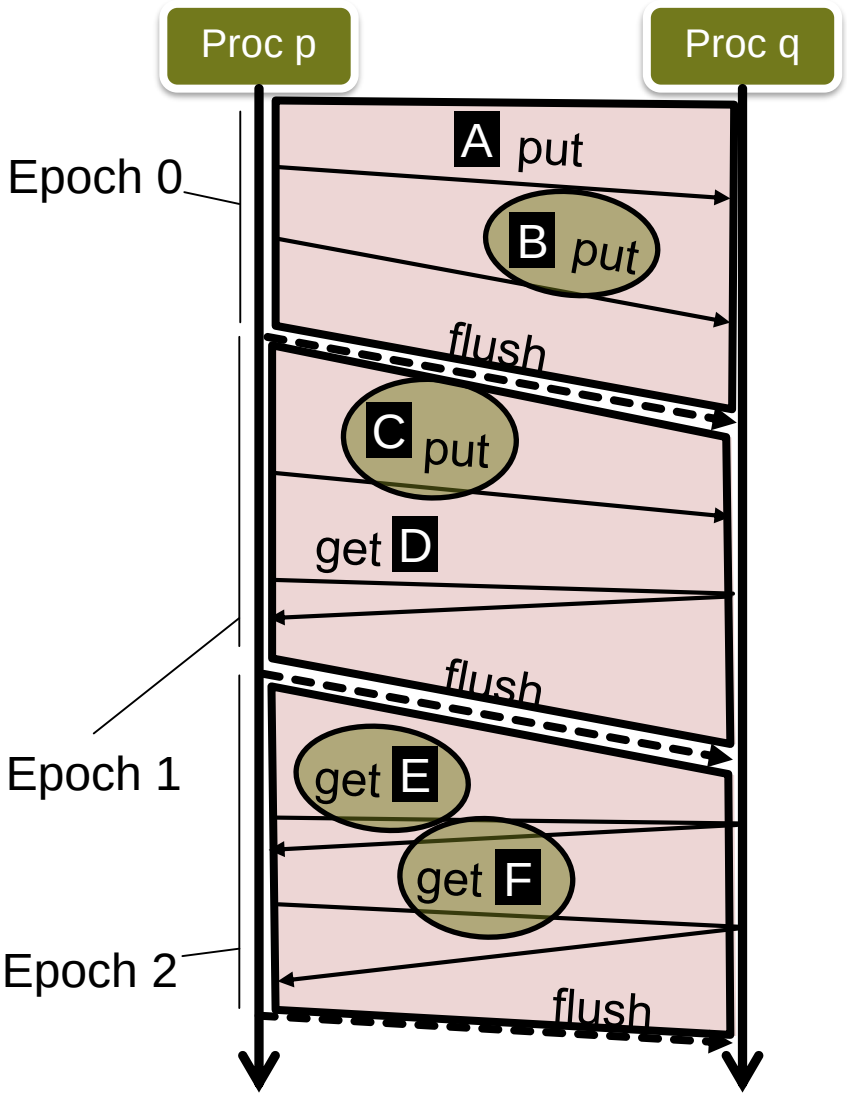
B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!

For this, we use **epoch counters (EC)**



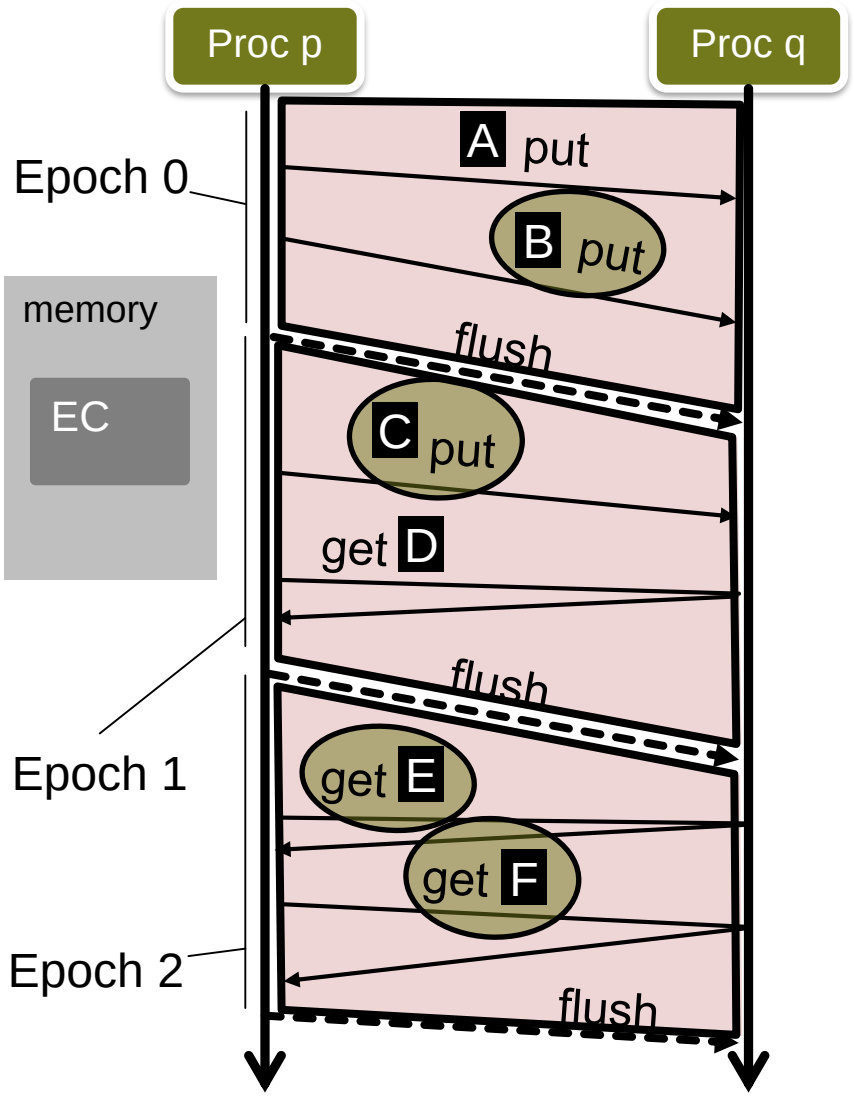
B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!

For this, we use **epoch counters (EC)**



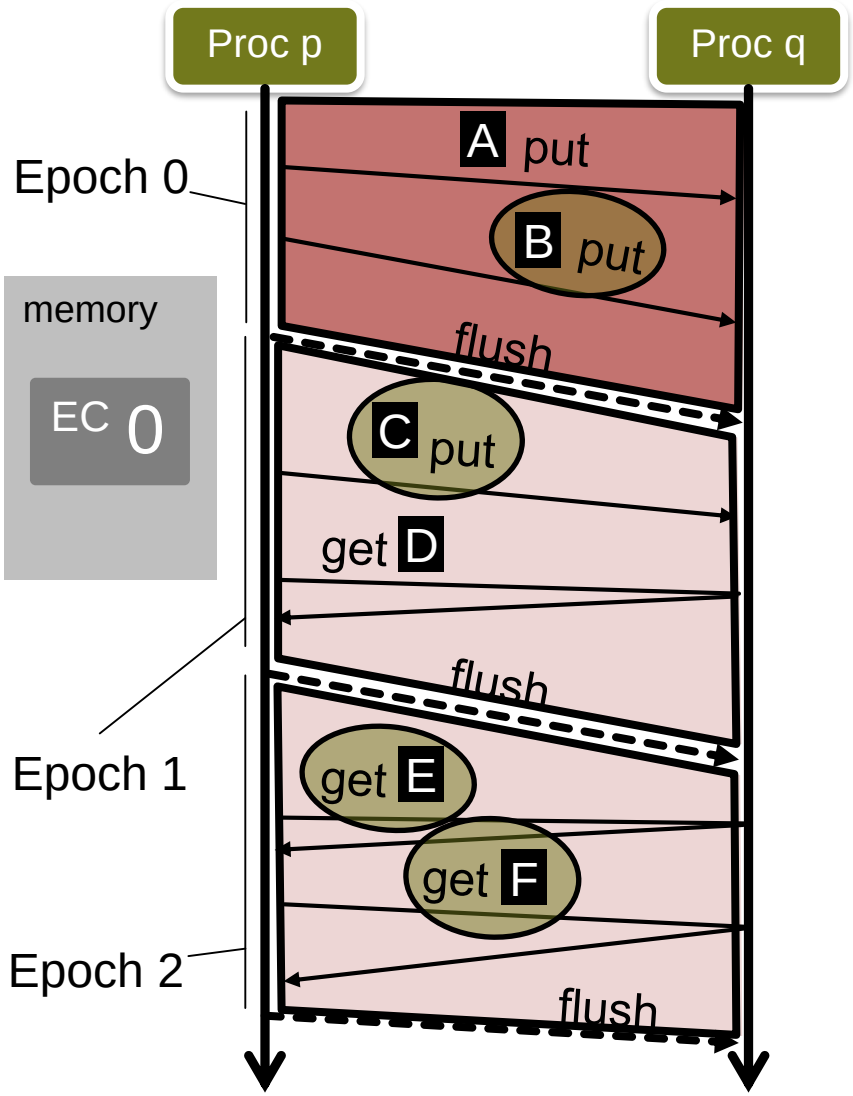
B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!

For this, we use **epoch counters (EC)**



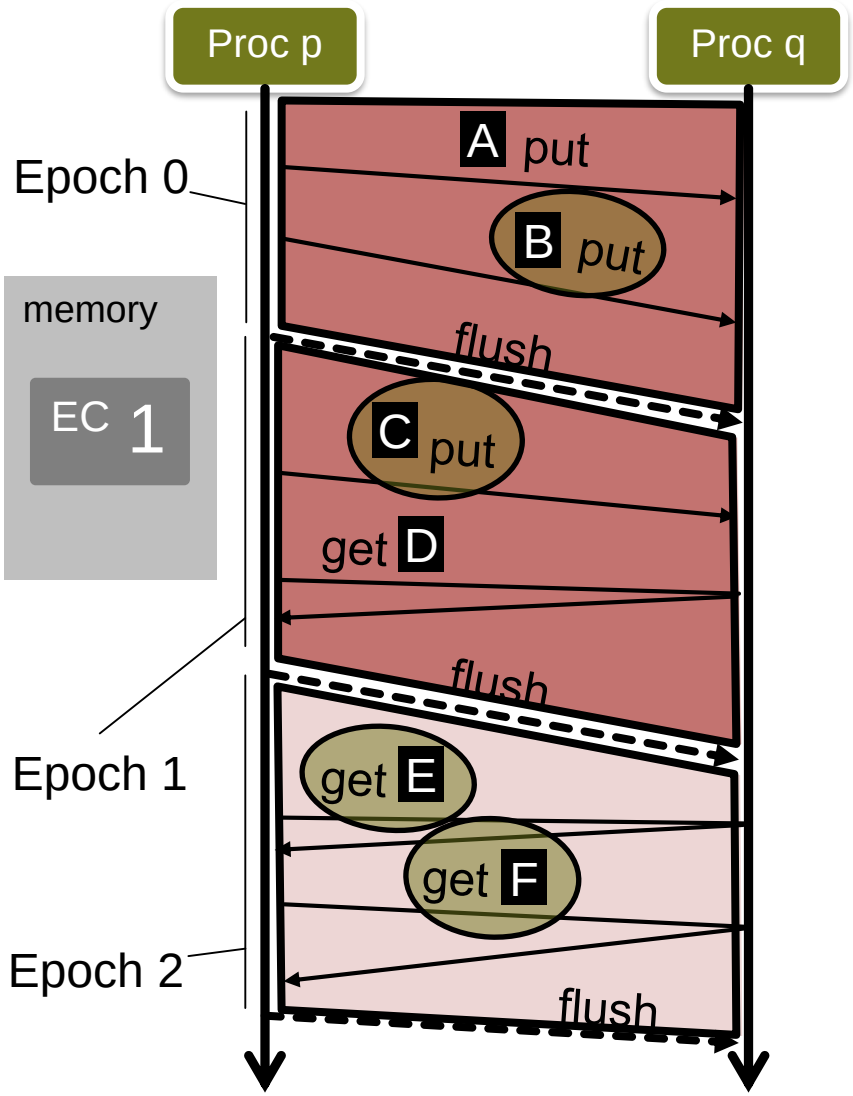
B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!

For this, we use **epoch counters (EC)**



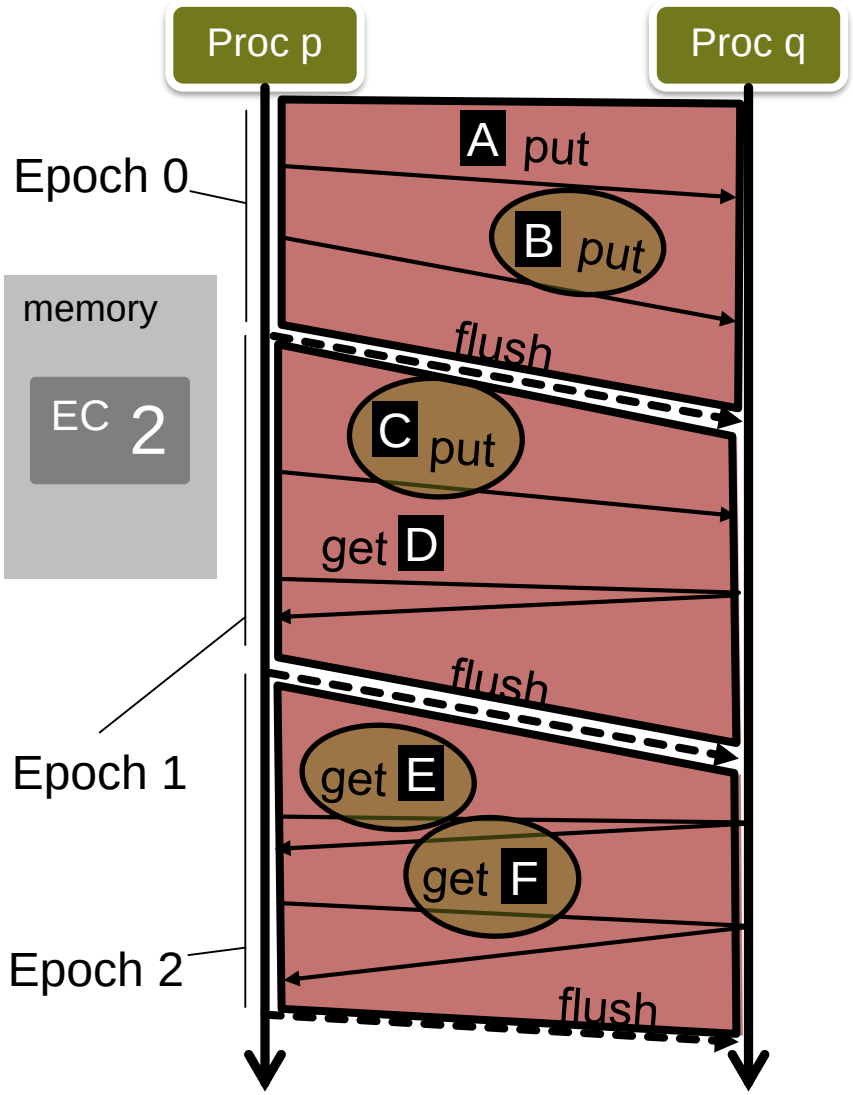
B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!

For this, we use **epoch counters (EC)**



B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\text{PUT}(p \rightrightarrows q)$
$\text{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

MP vs. RMA

Schemes

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Holistic fault-tolerance library

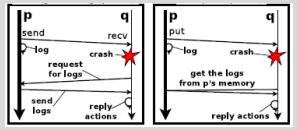
Design

Checkpoints on demand

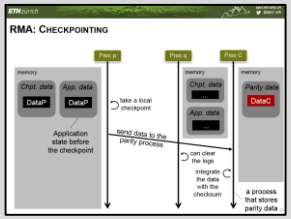
Optimizations

Performance

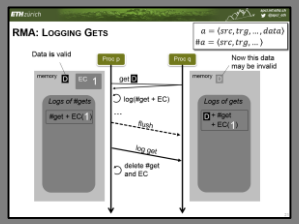
UC in RMA



MP vs. RMA



Checkpointing schemes



Logging accesses

Topology-awareness

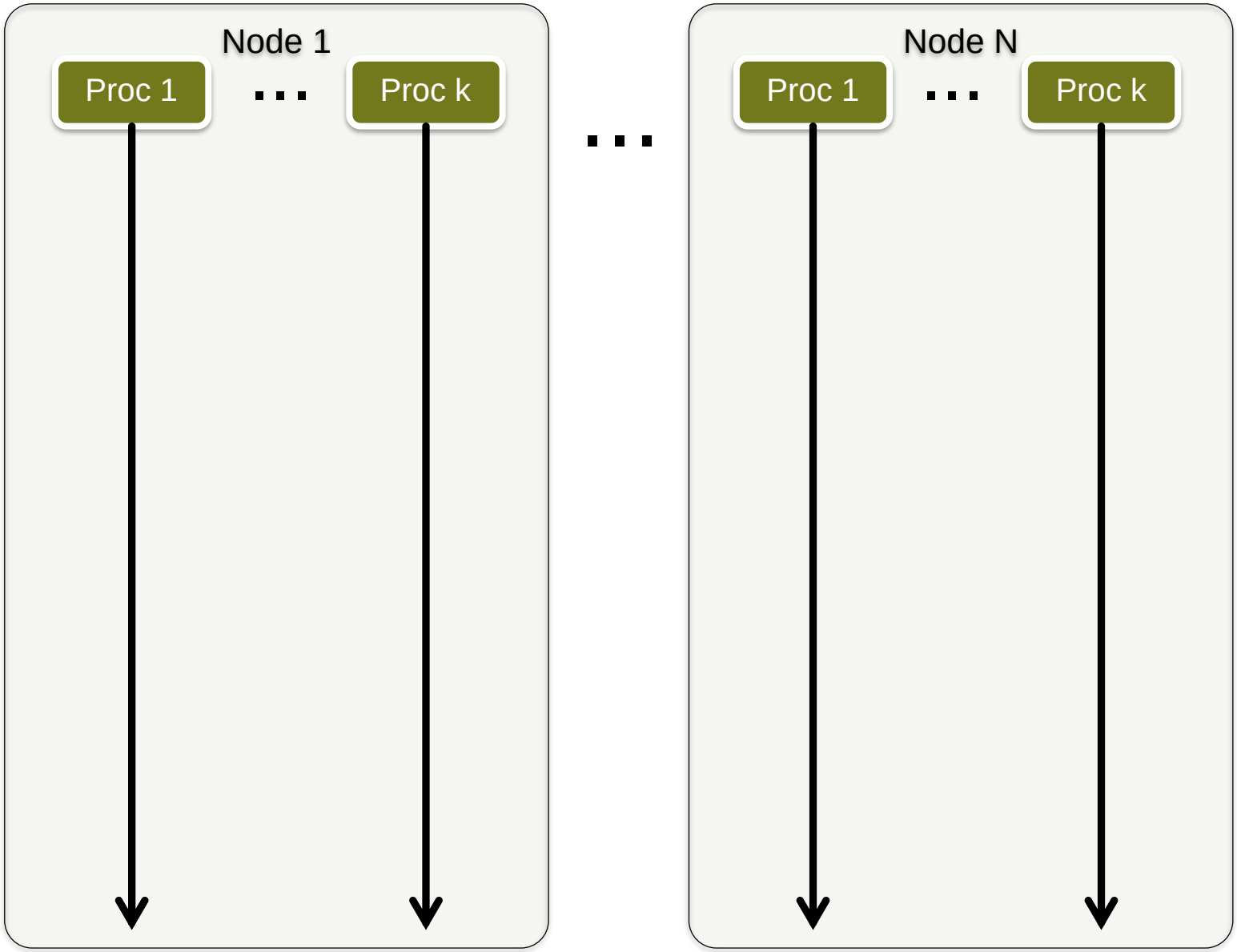
$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

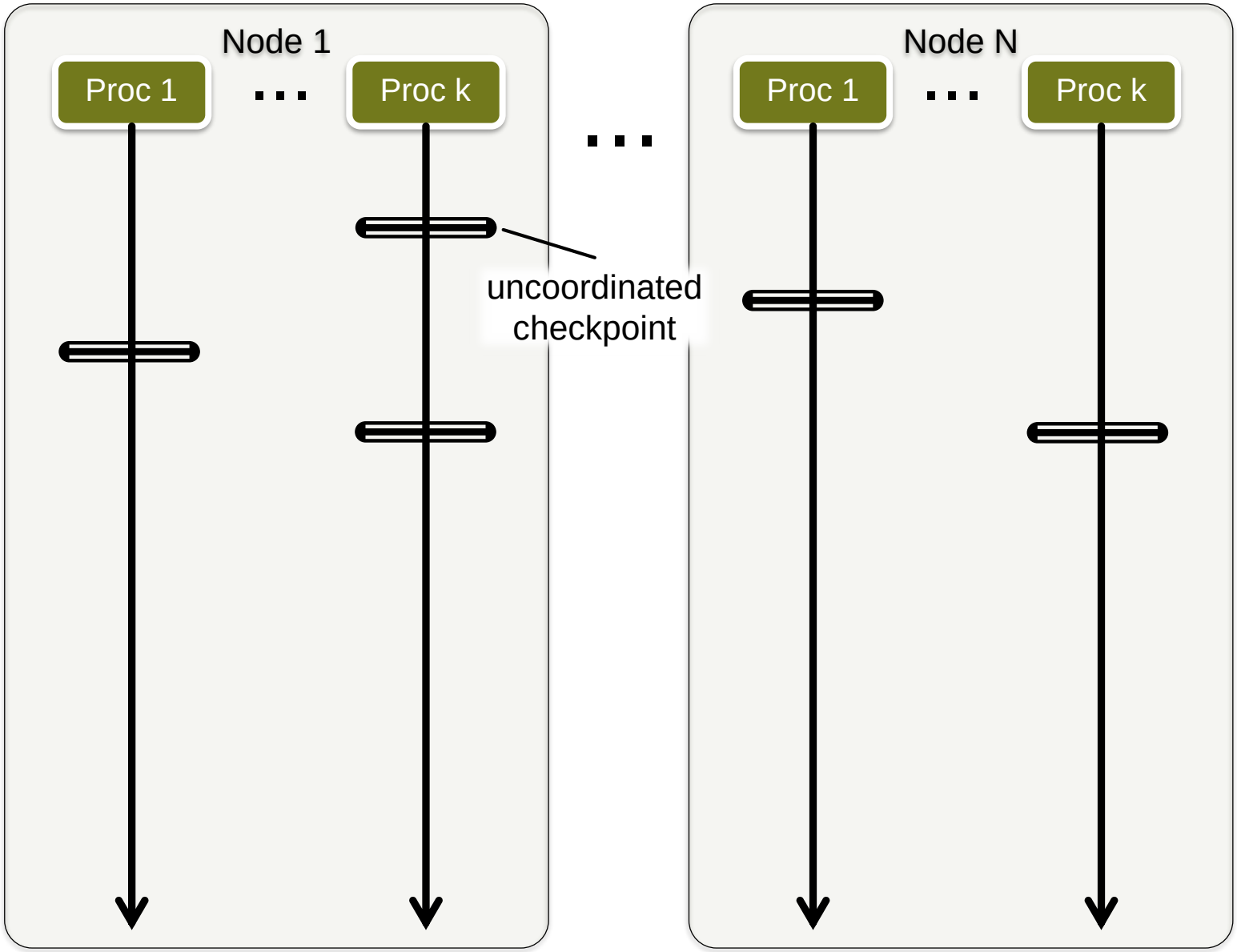
Distribution of processes

Decreasing failure prob.

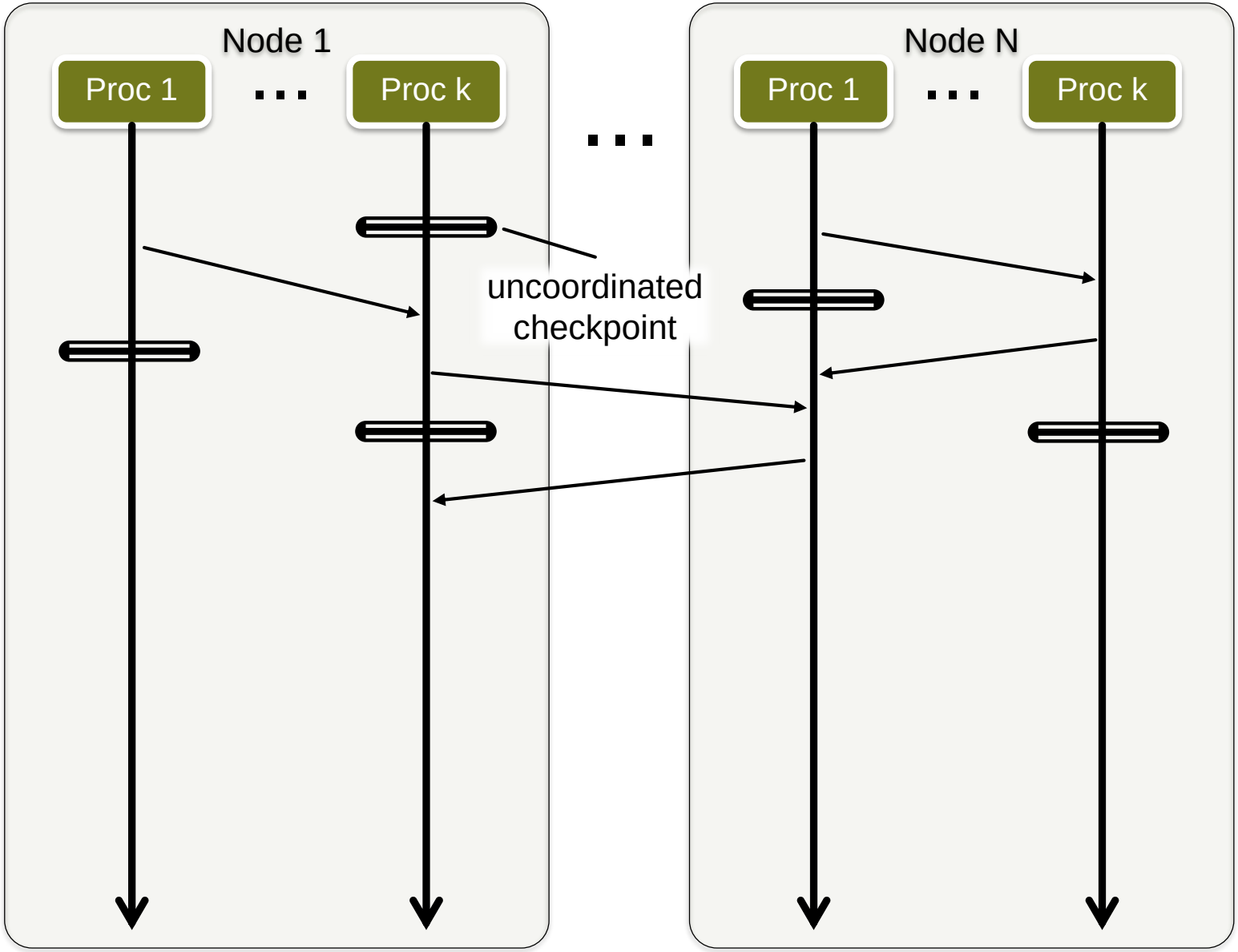
UNCOORDINATED CHECKPOINTING (MP)



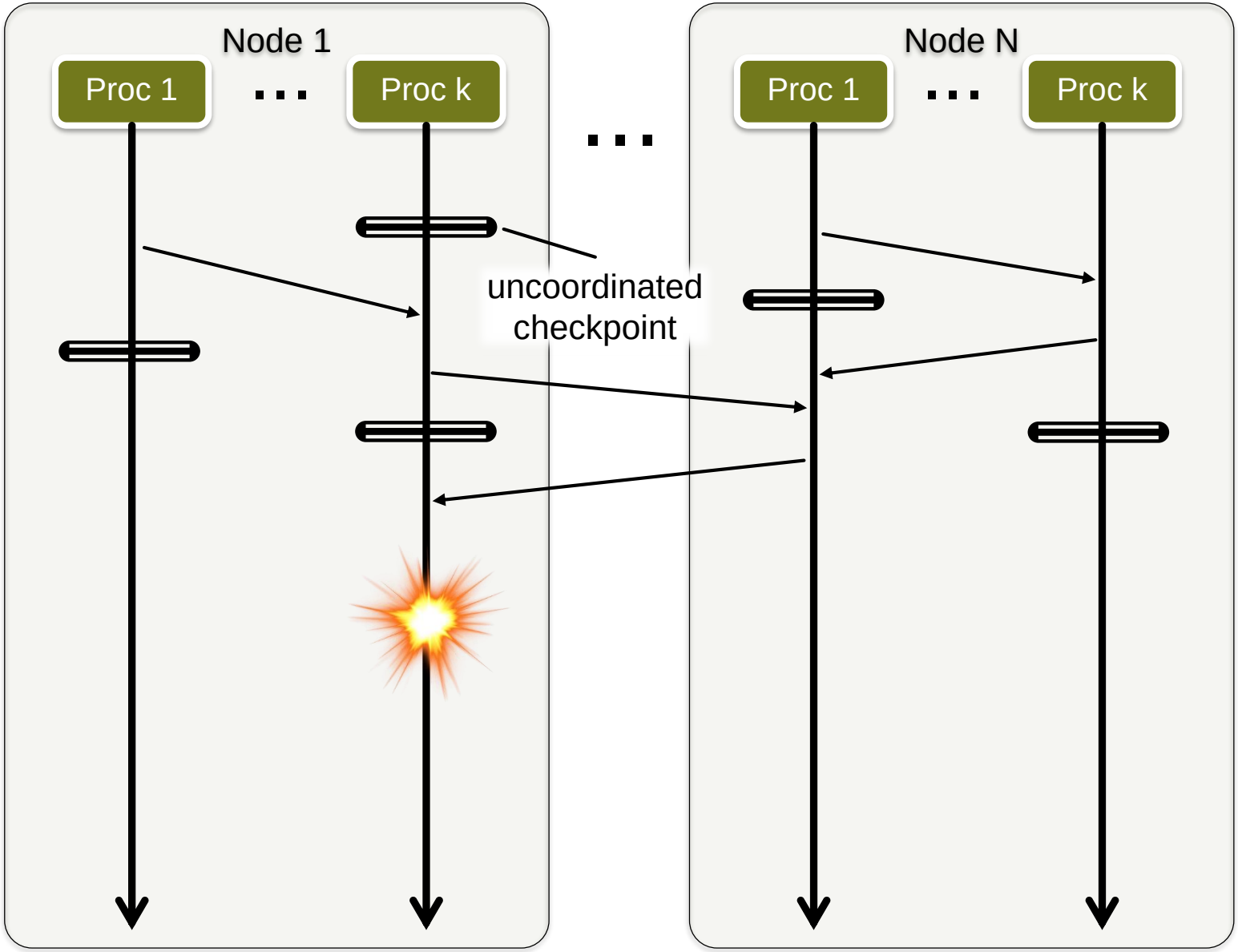
UNCOORDINATED CHECKPOINTING (MP)



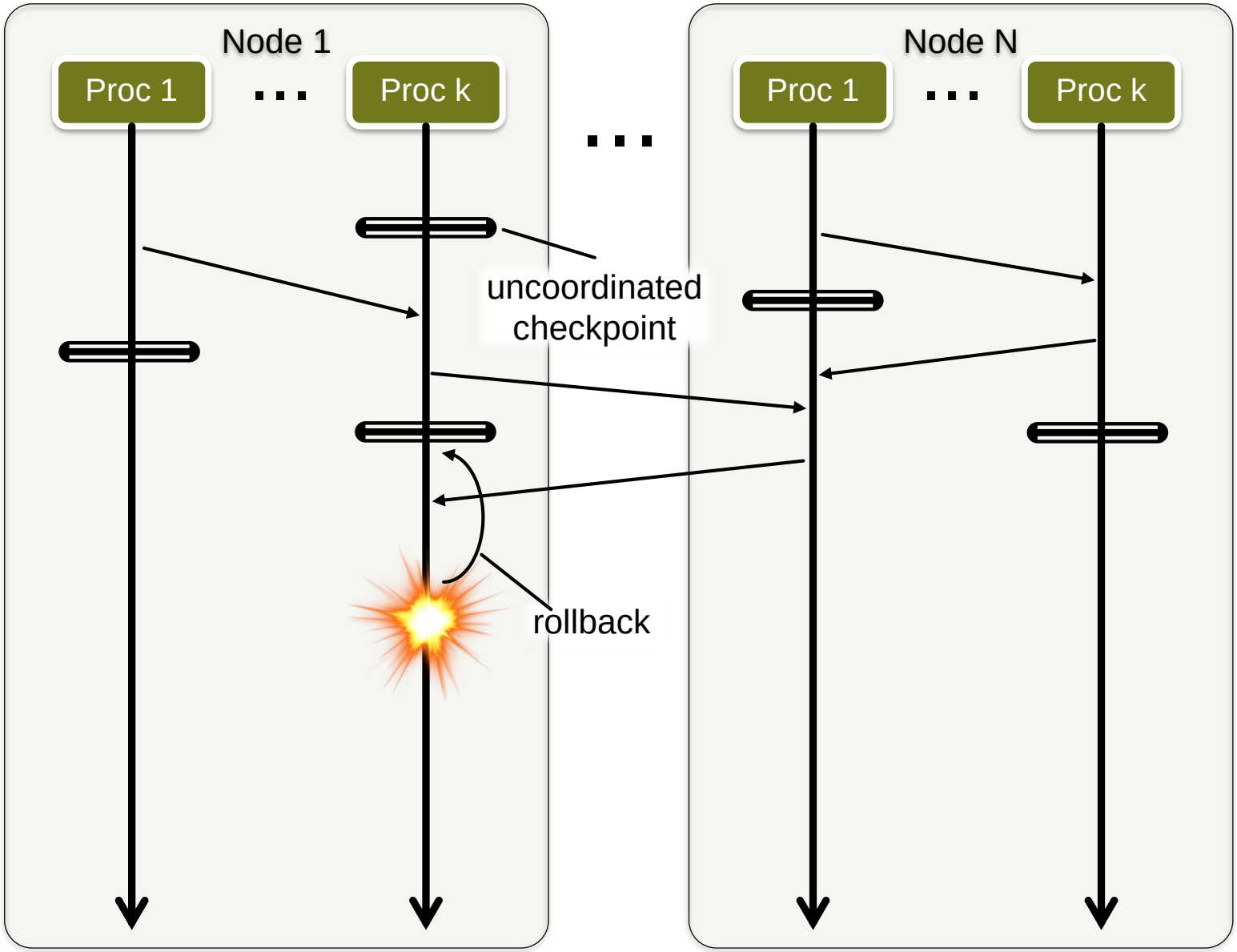
UNCOORDINATED CHECKPOINTING (MP)



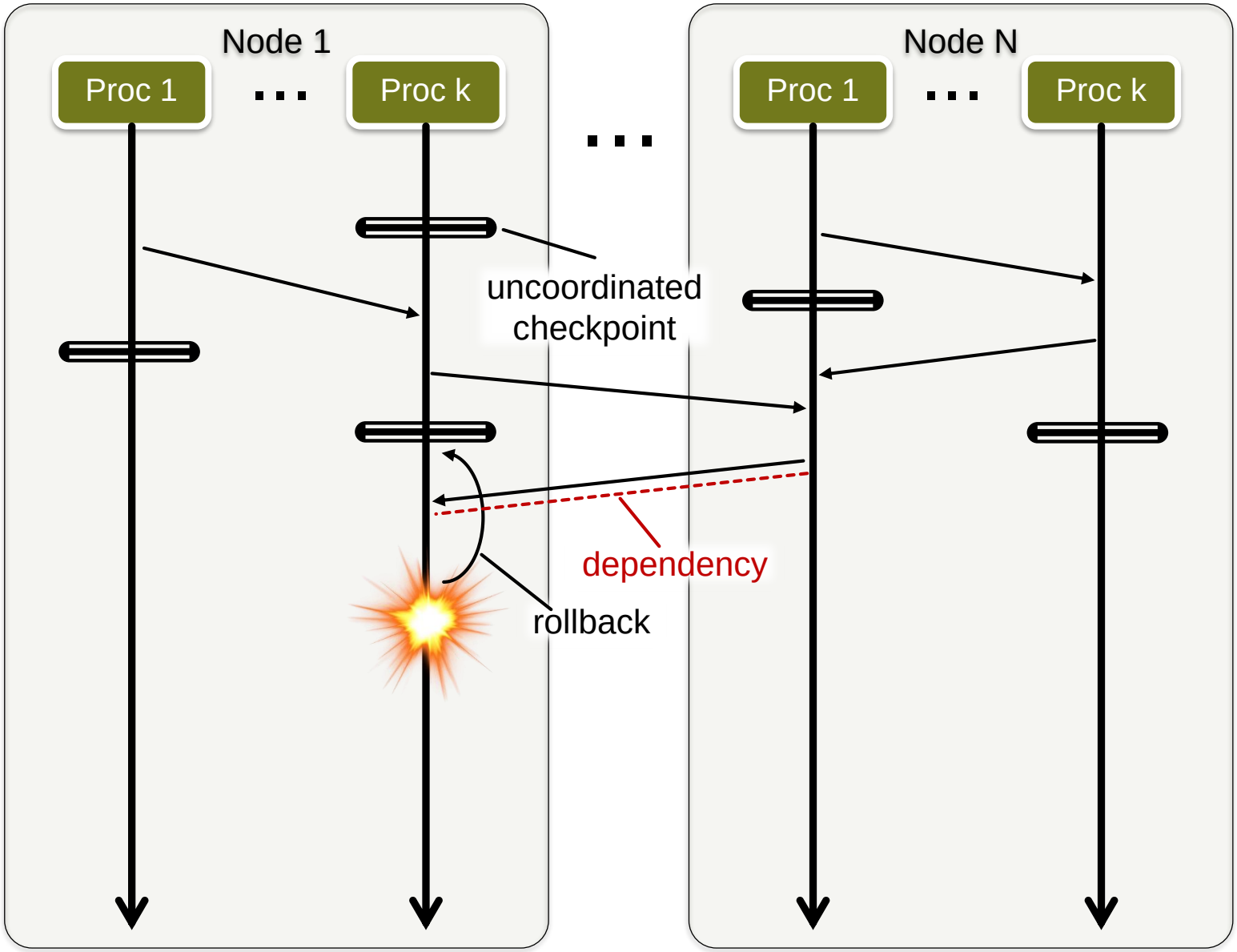
UNCOORDINATED CHECKPOINTING (MP)



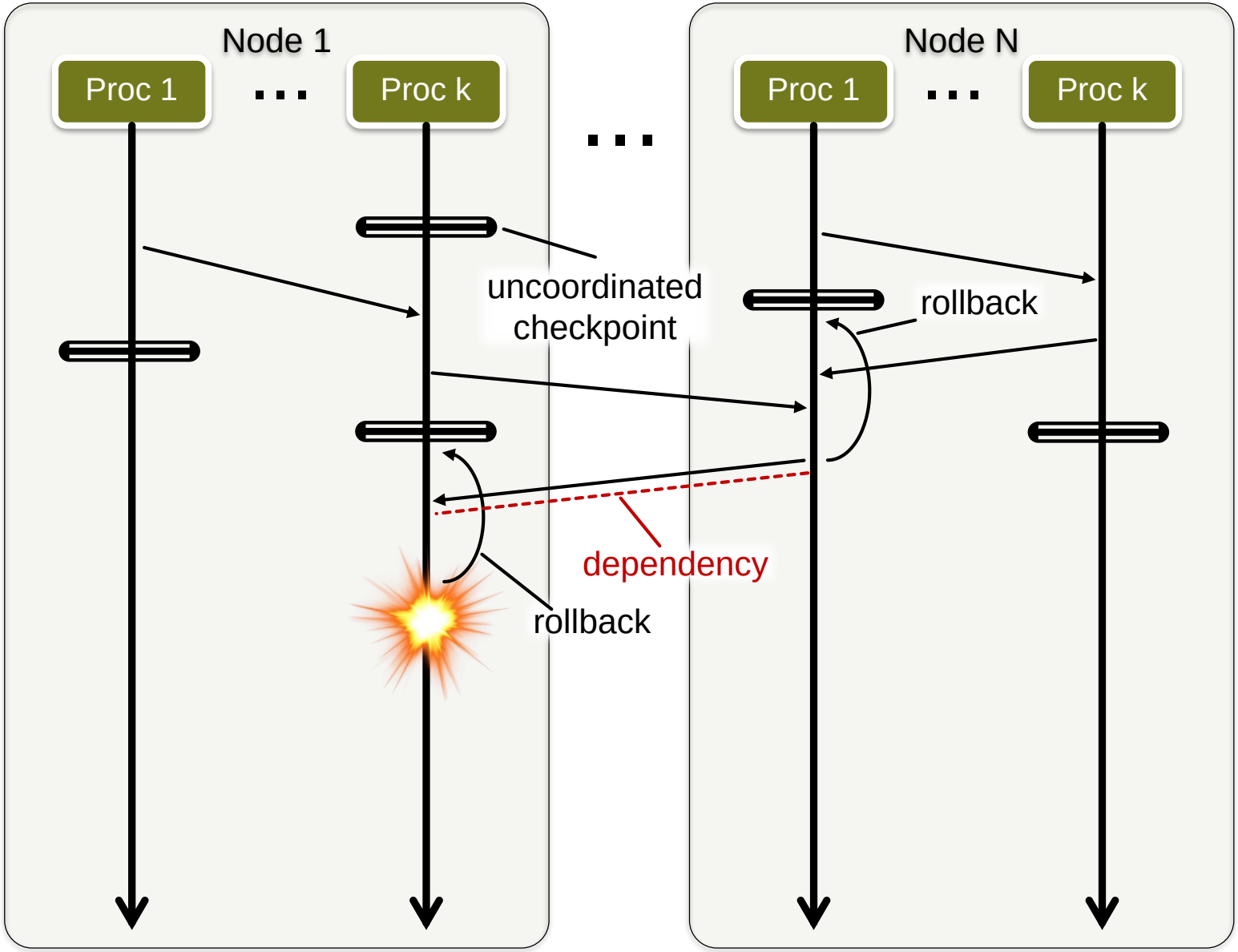
UNCOORDINATED CHECKPOINTING (MP)



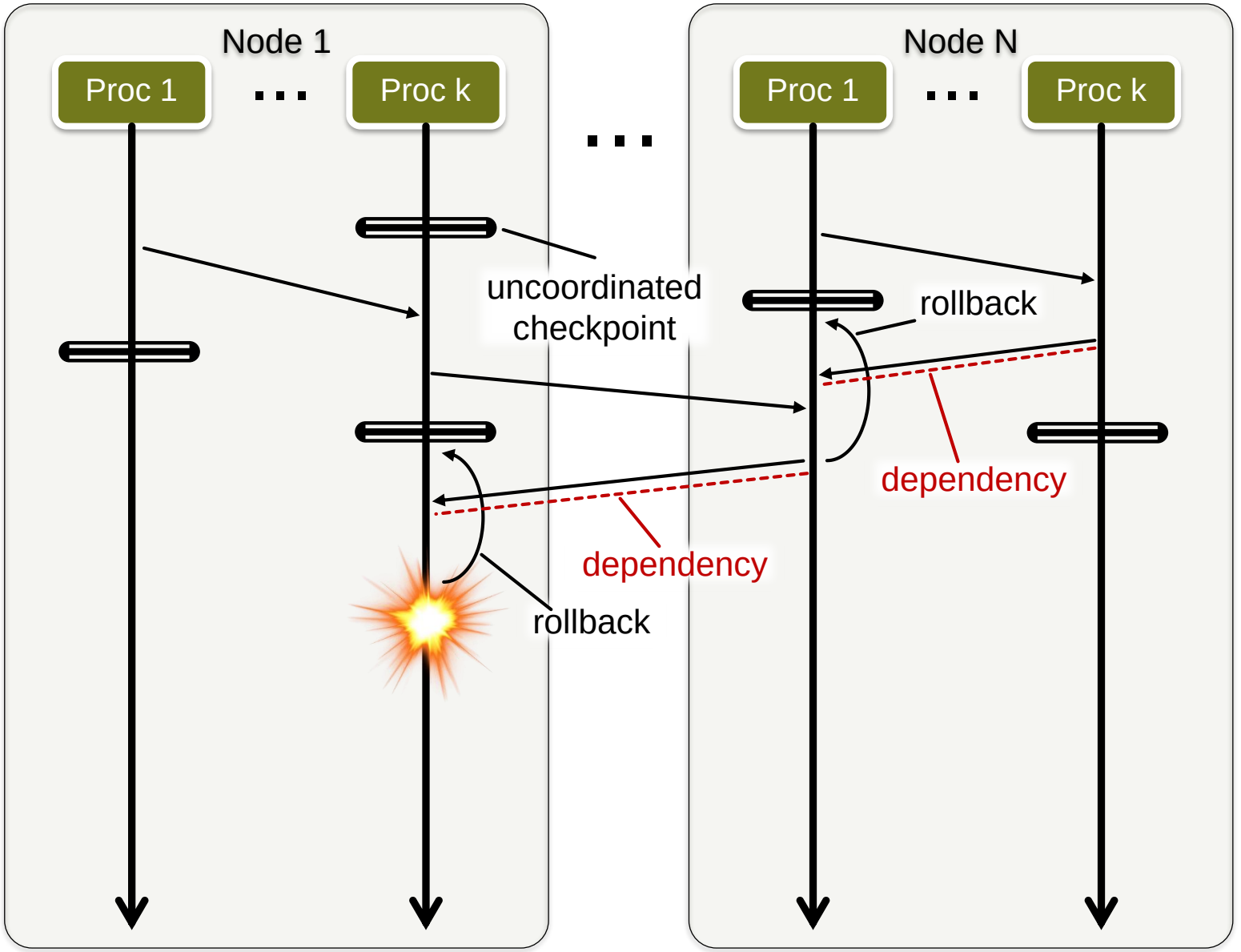
UNCOORDINATED CHECKPOINTING (MP)



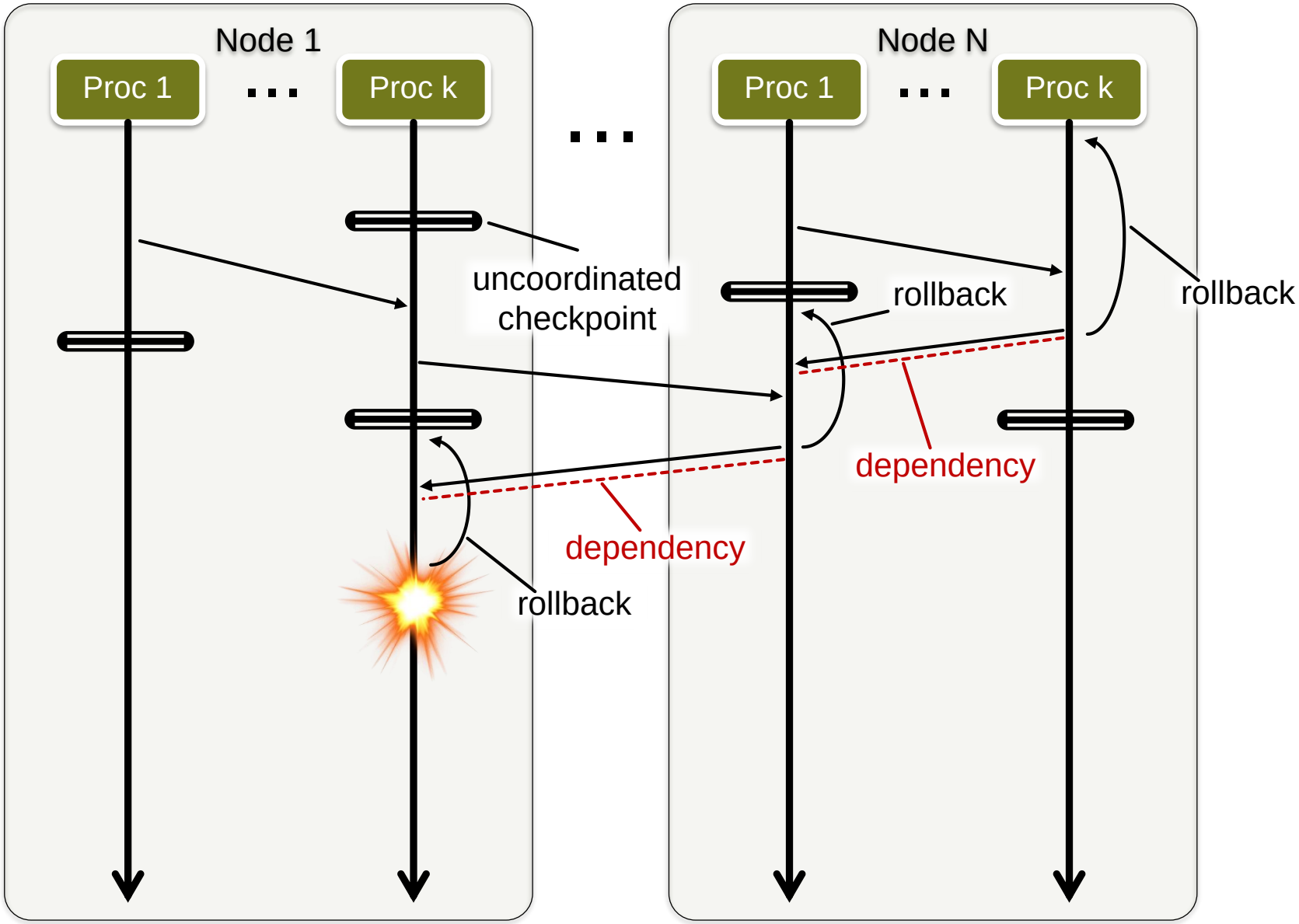
UNCOORDINATED CHECKPOINTING (MP)



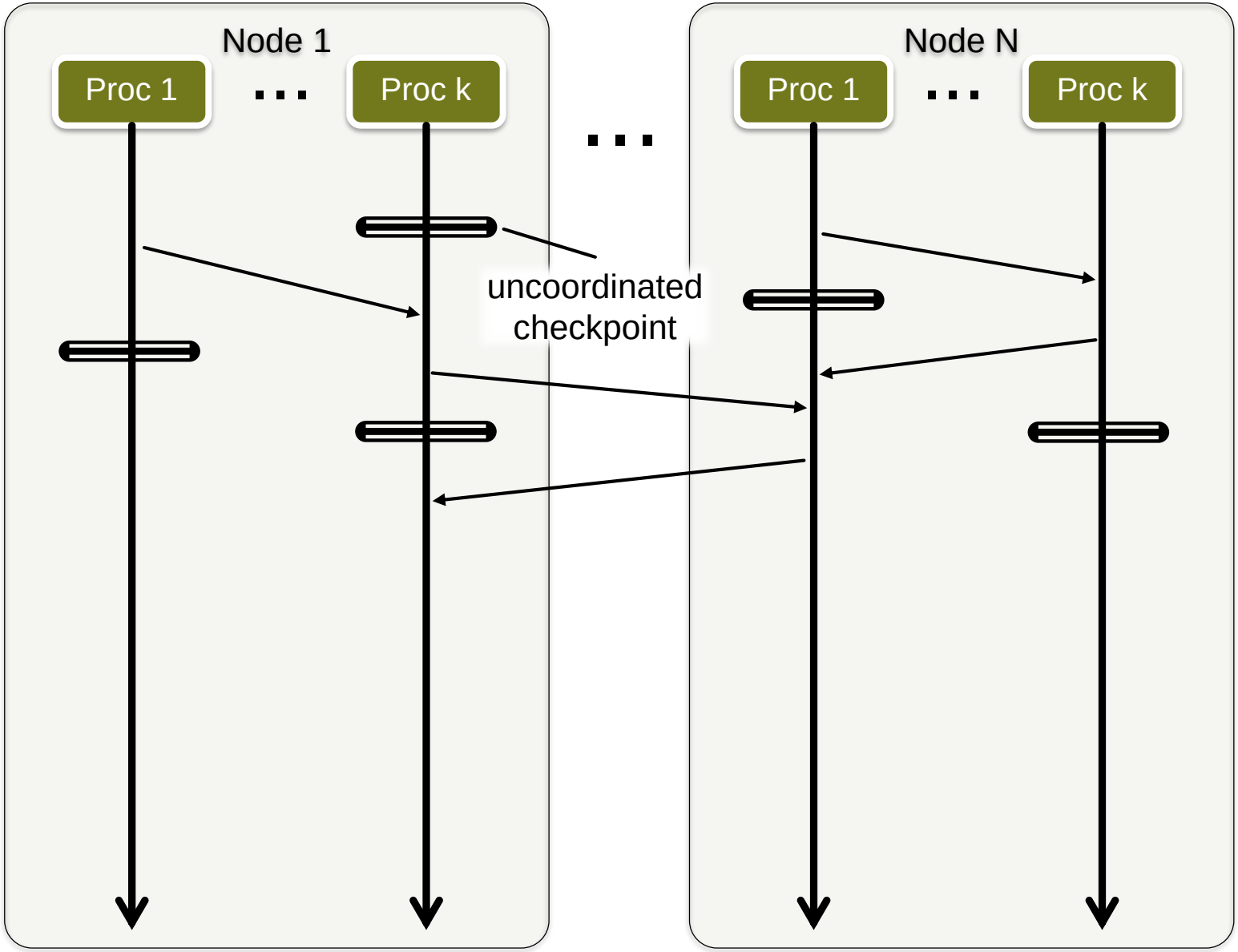
UNCOORDINATED CHECKPOINTING (MP)



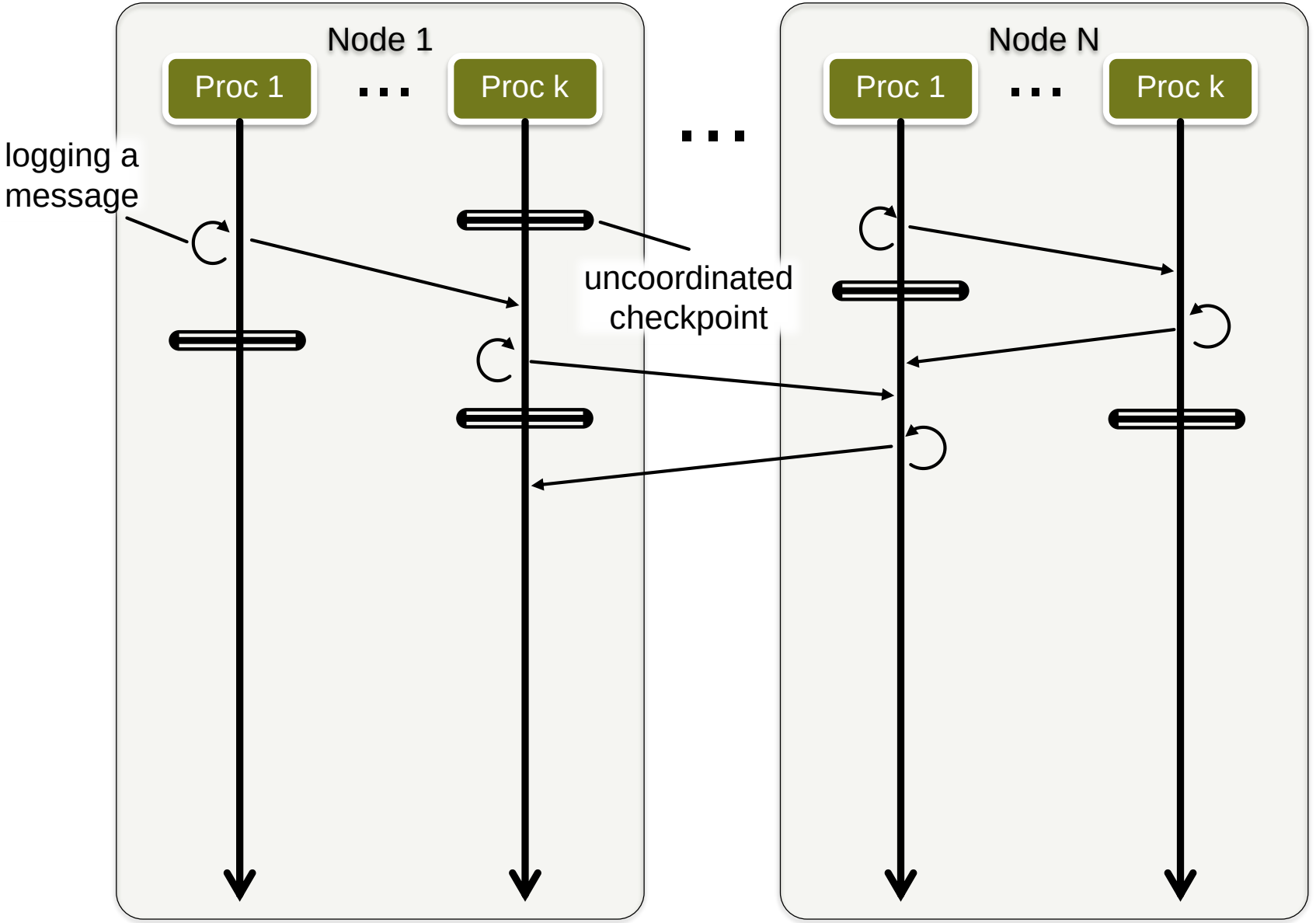
UNCOORDINATED CHECKPOINTING (MP)



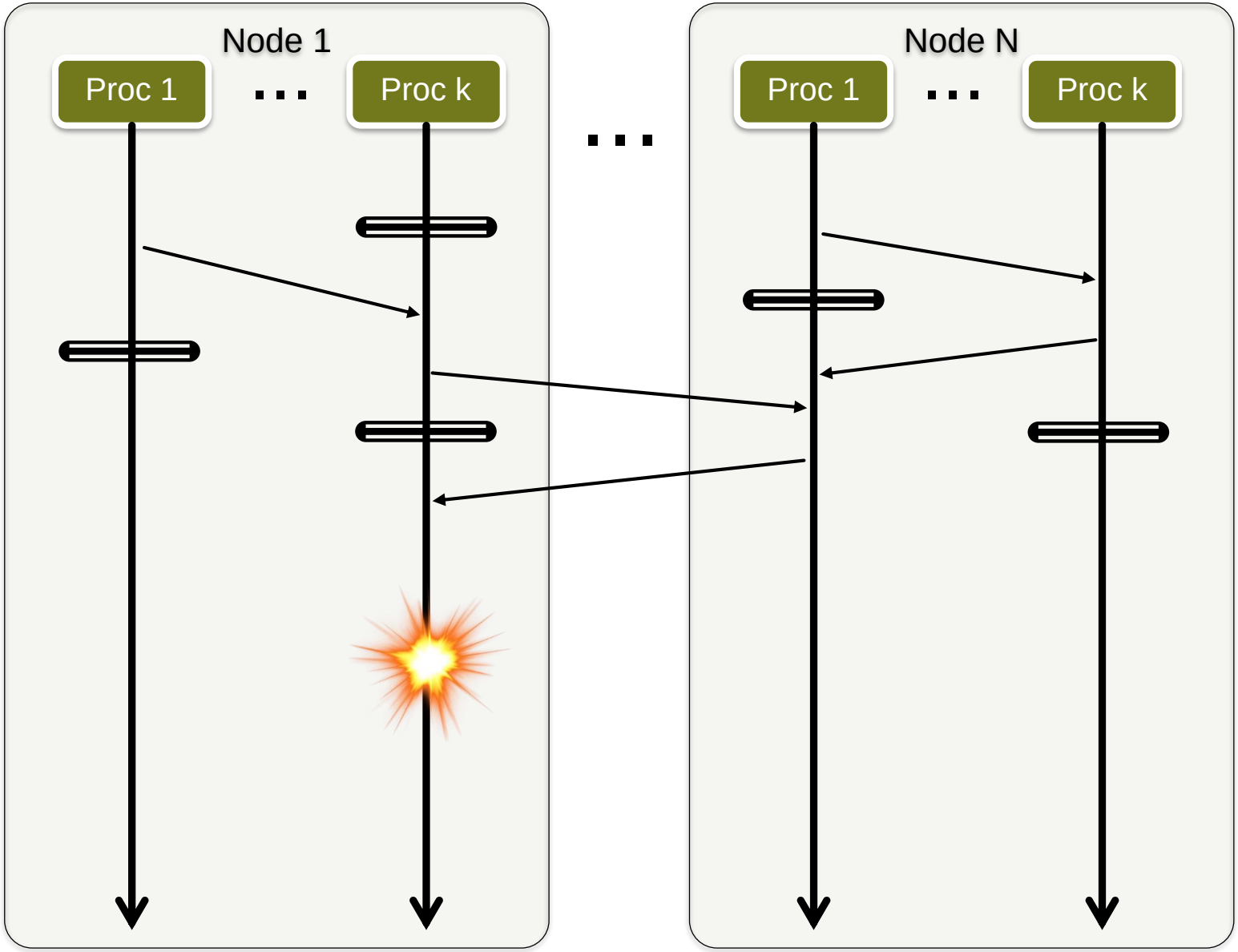
UNCOORDINATED CHECKPOINTING (MP)



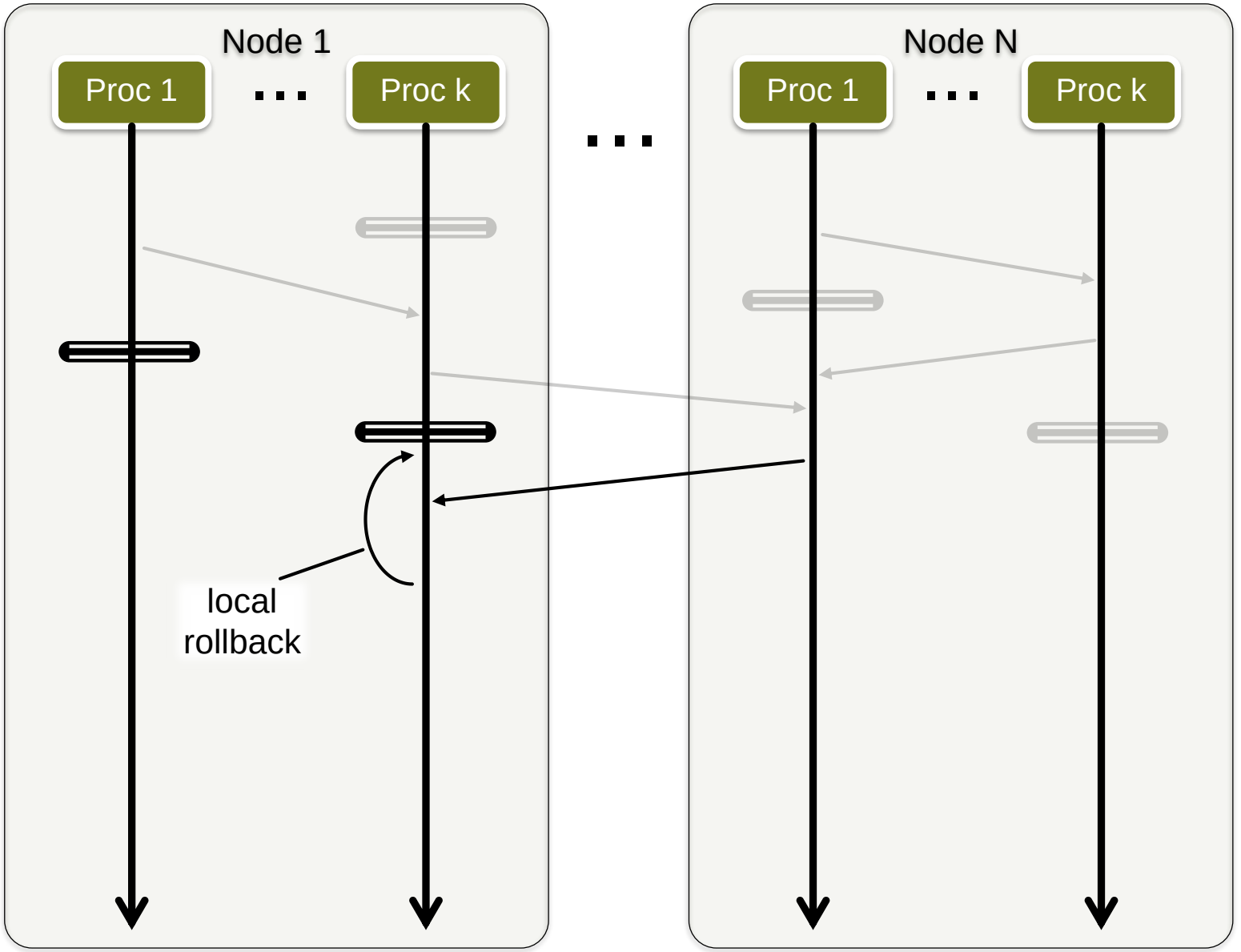
UNCOORDINATED CHECKPOINTING (MP)



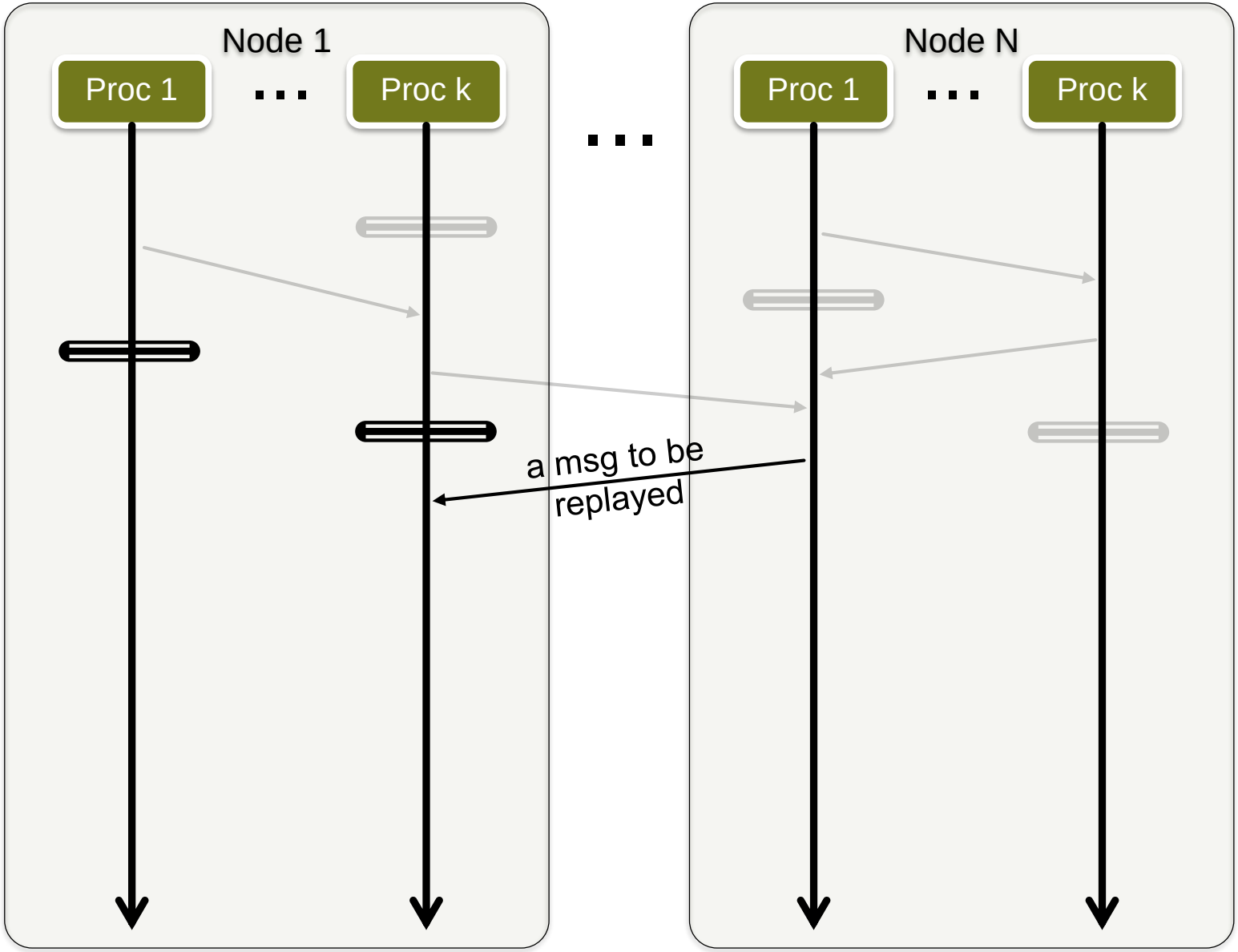
UNCOORDINATED CHECKPOINTING (MP)



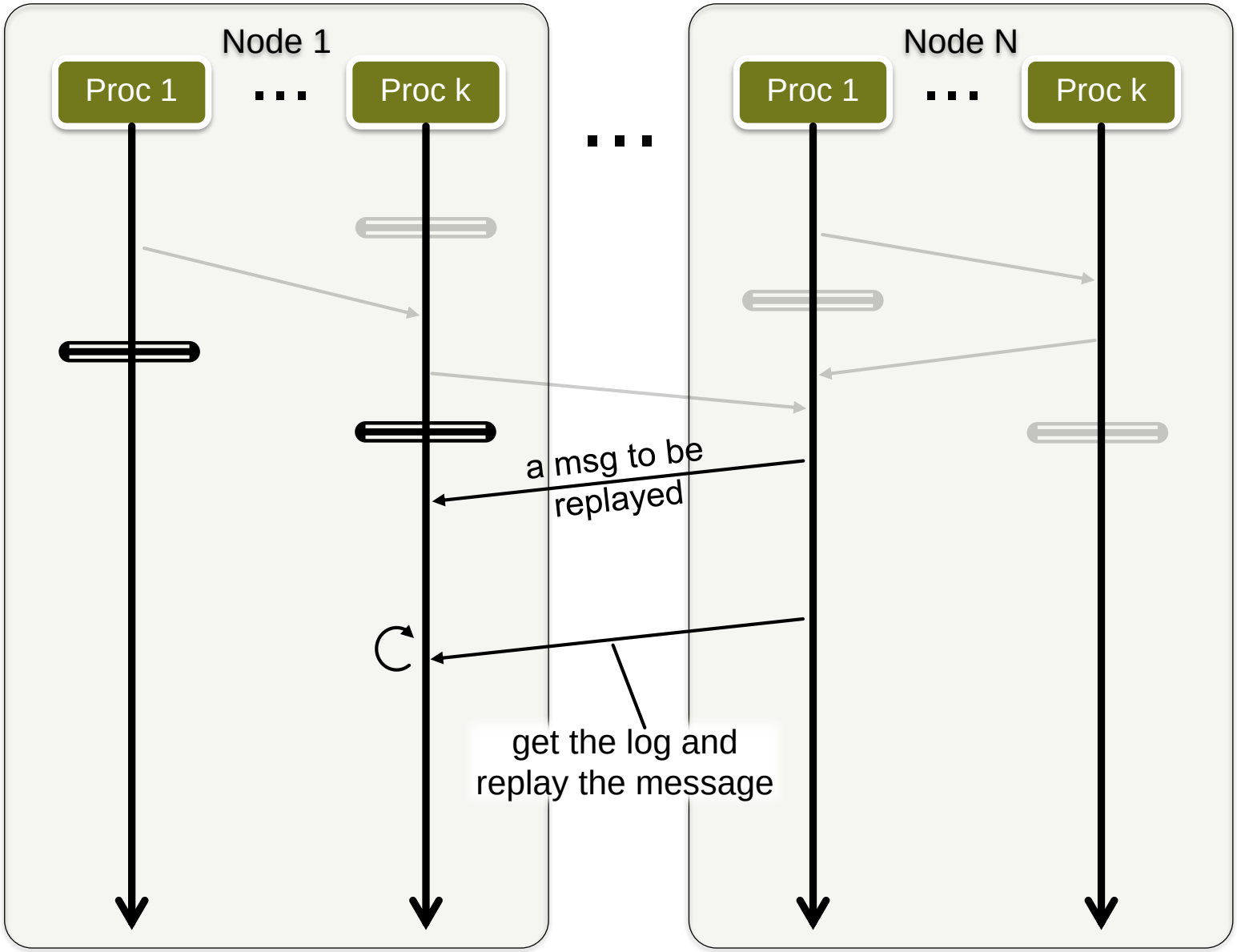
UNCOORDINATED CHECKPOINTING (MP)



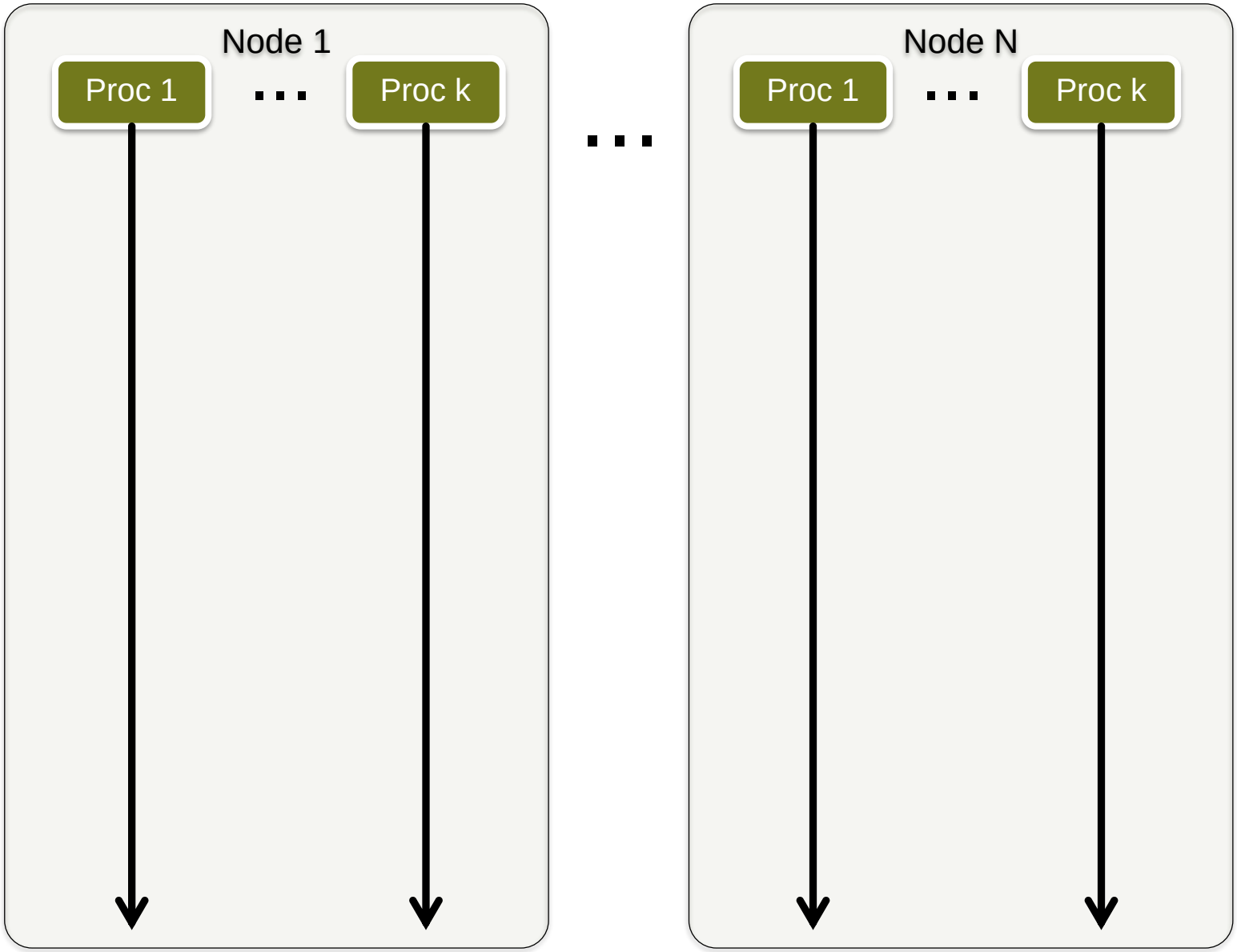
UNCOORDINATED CHECKPOINTING (MP)



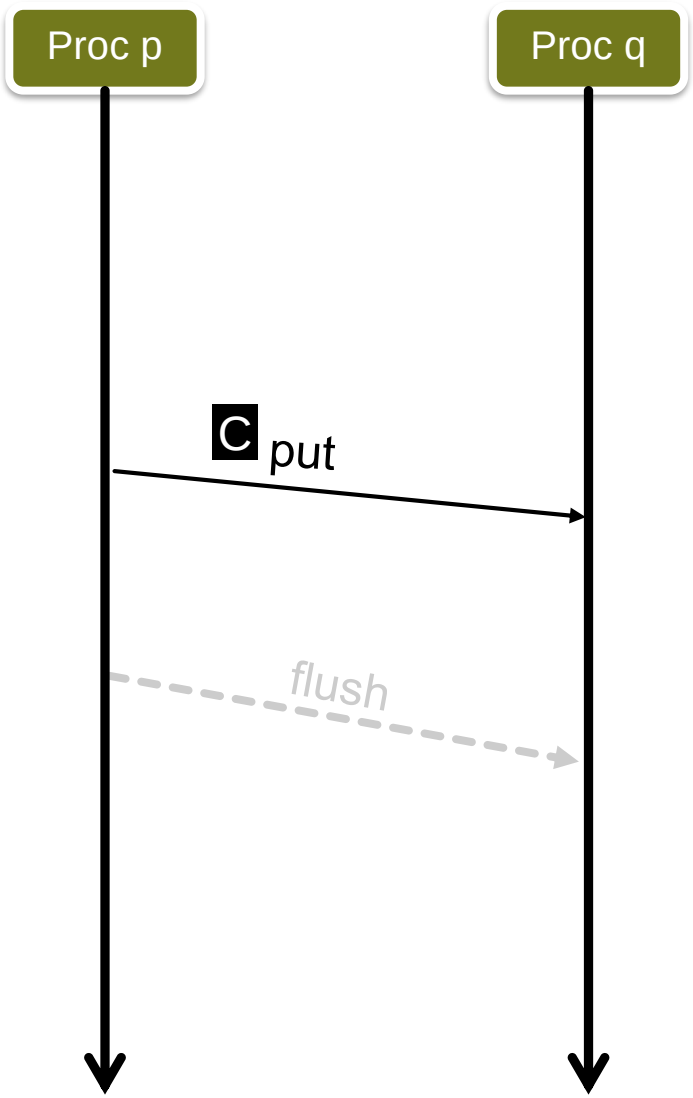
UNCOORDINATED CHECKPOINTING (MP)



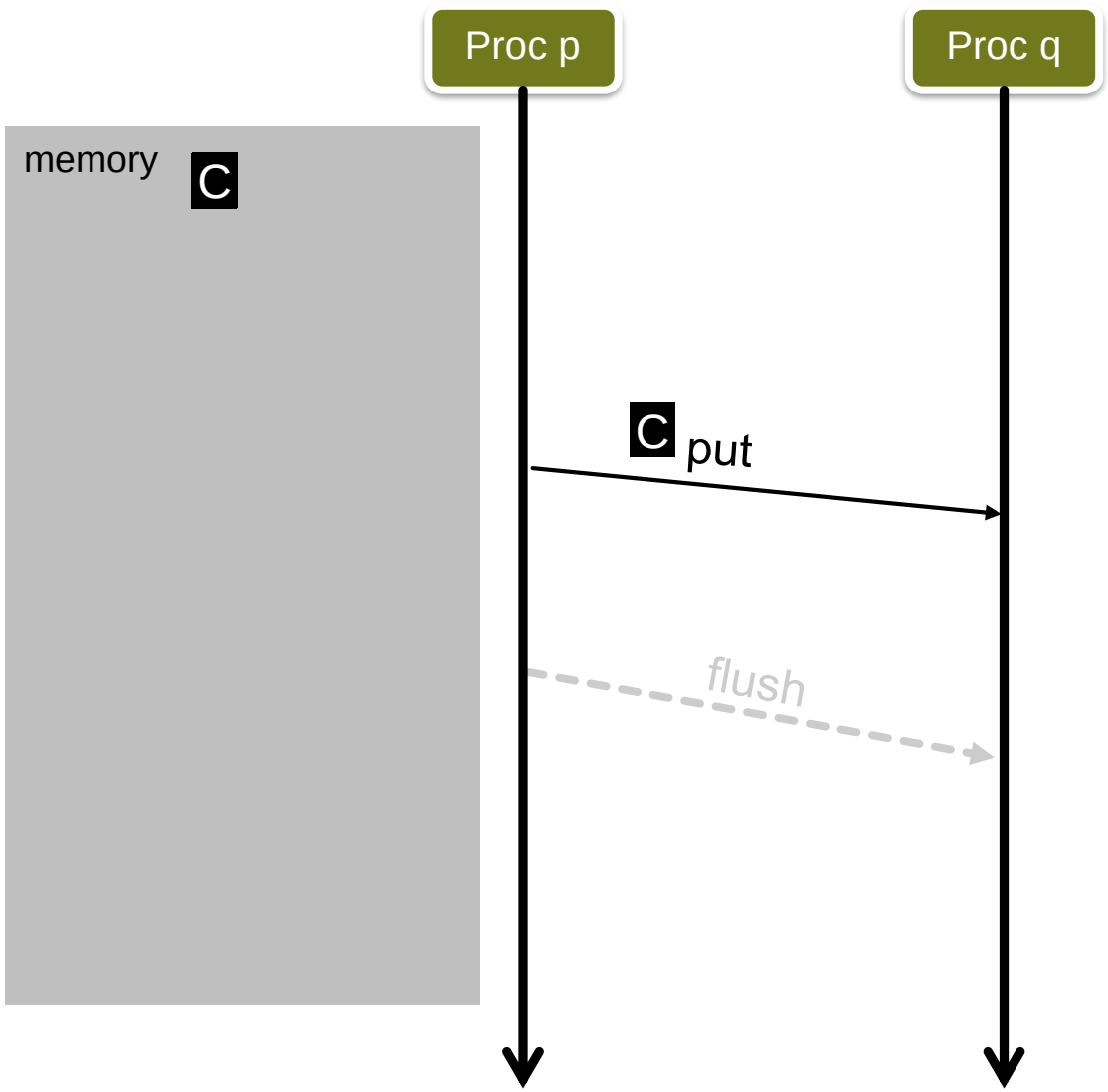
UNCOORDINATED CHECKPOINTING (MP)



RMA: Logging Puts

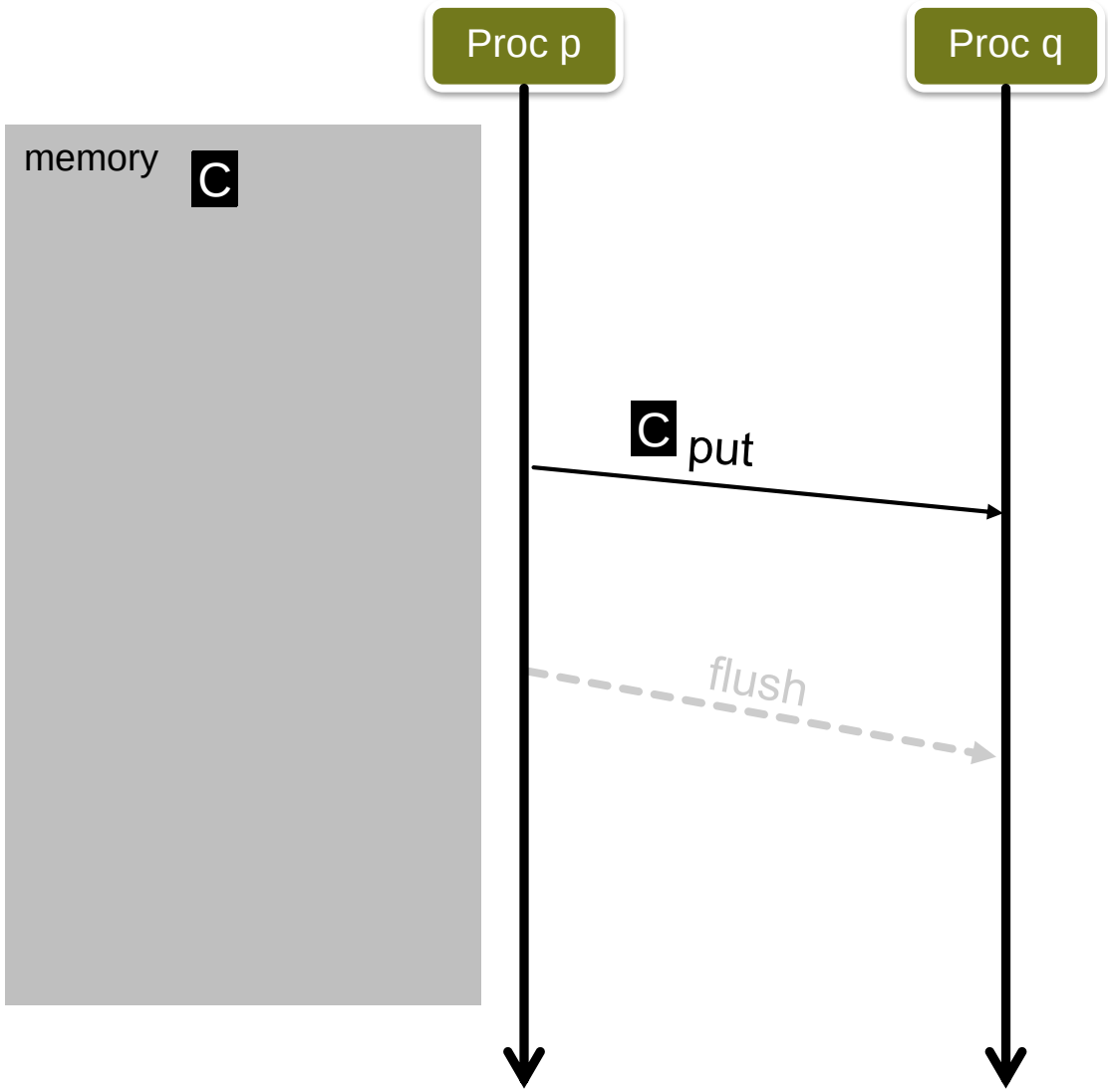


RMA: Logging Puts



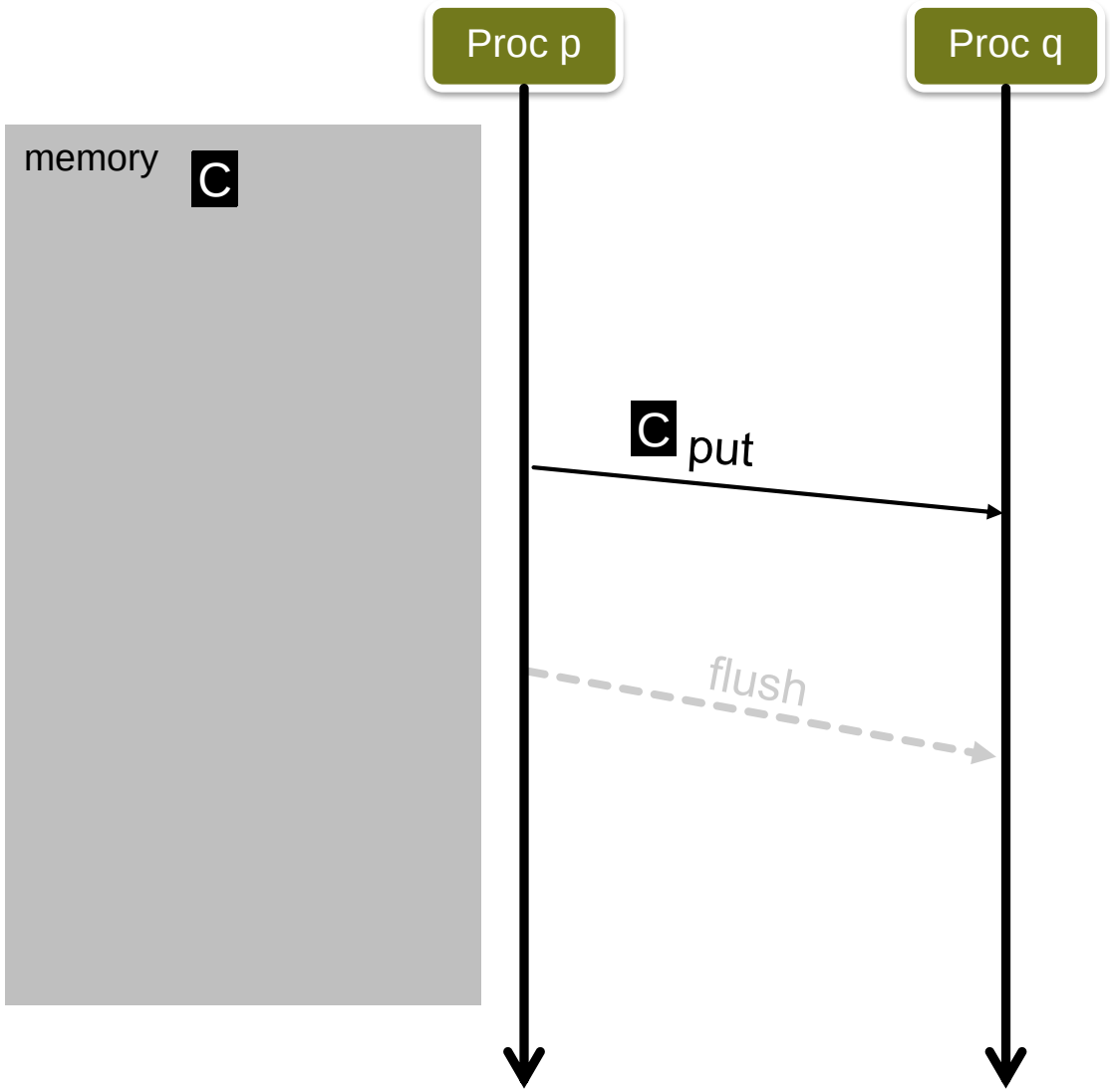
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$



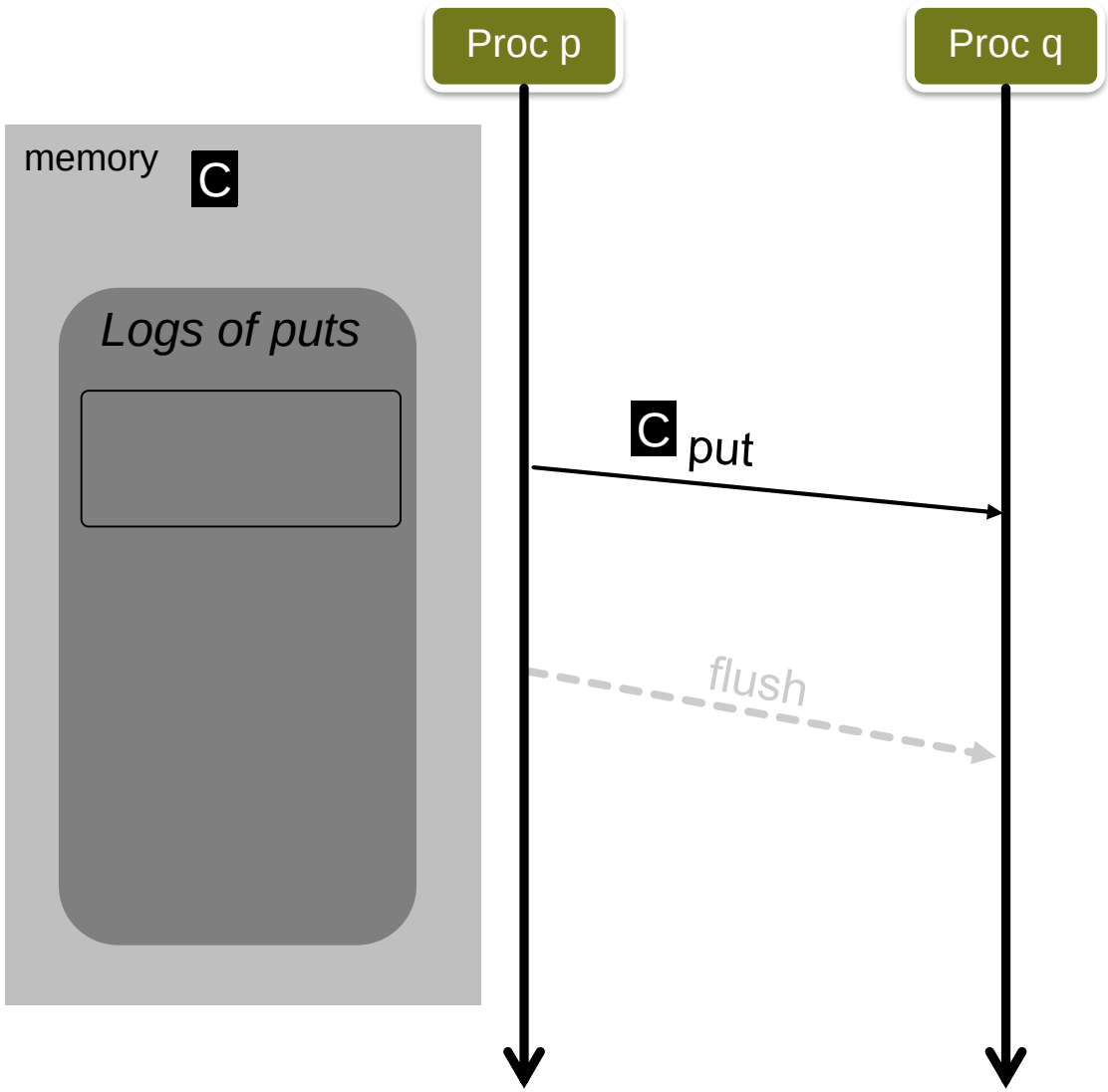
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



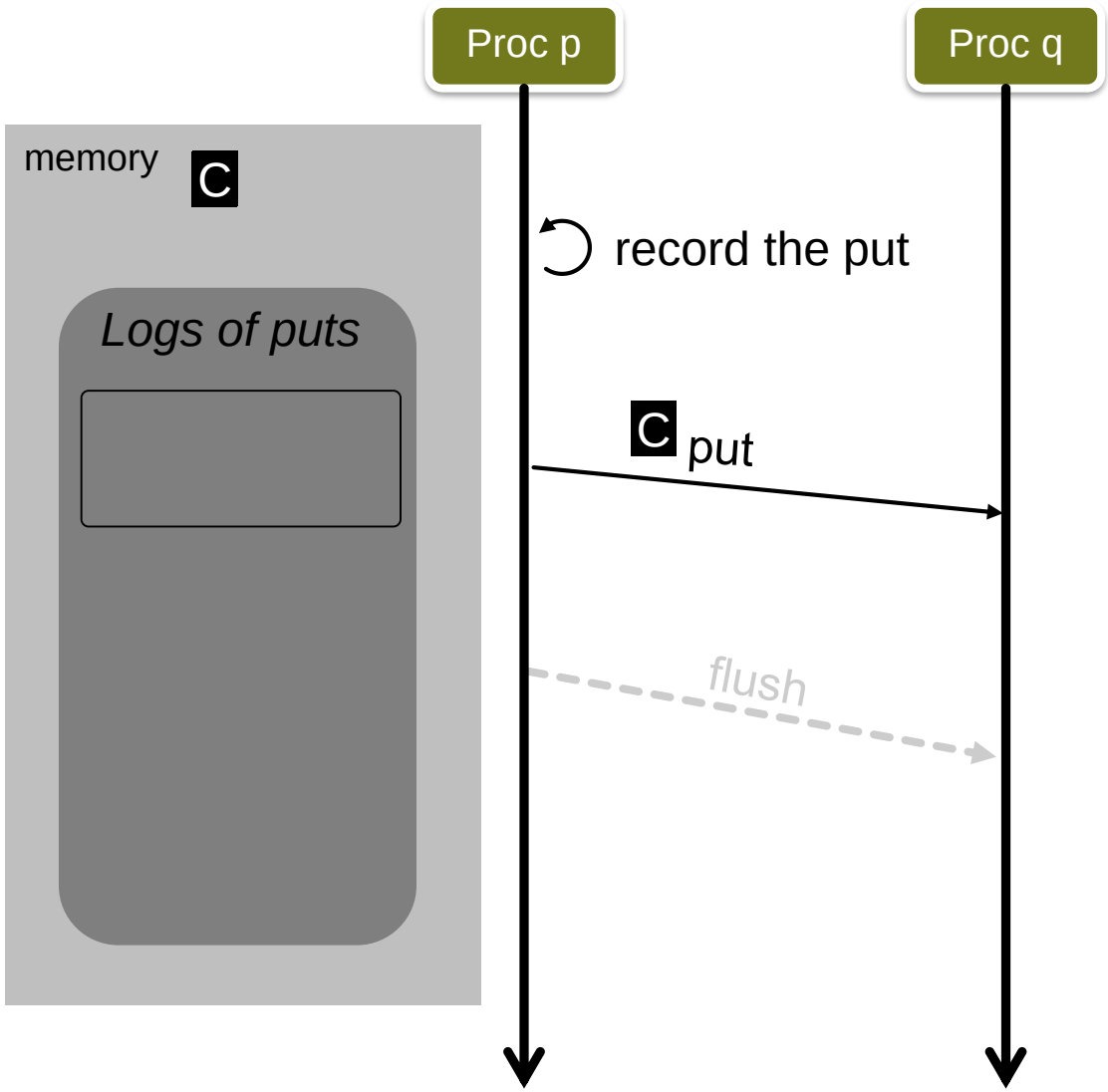
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



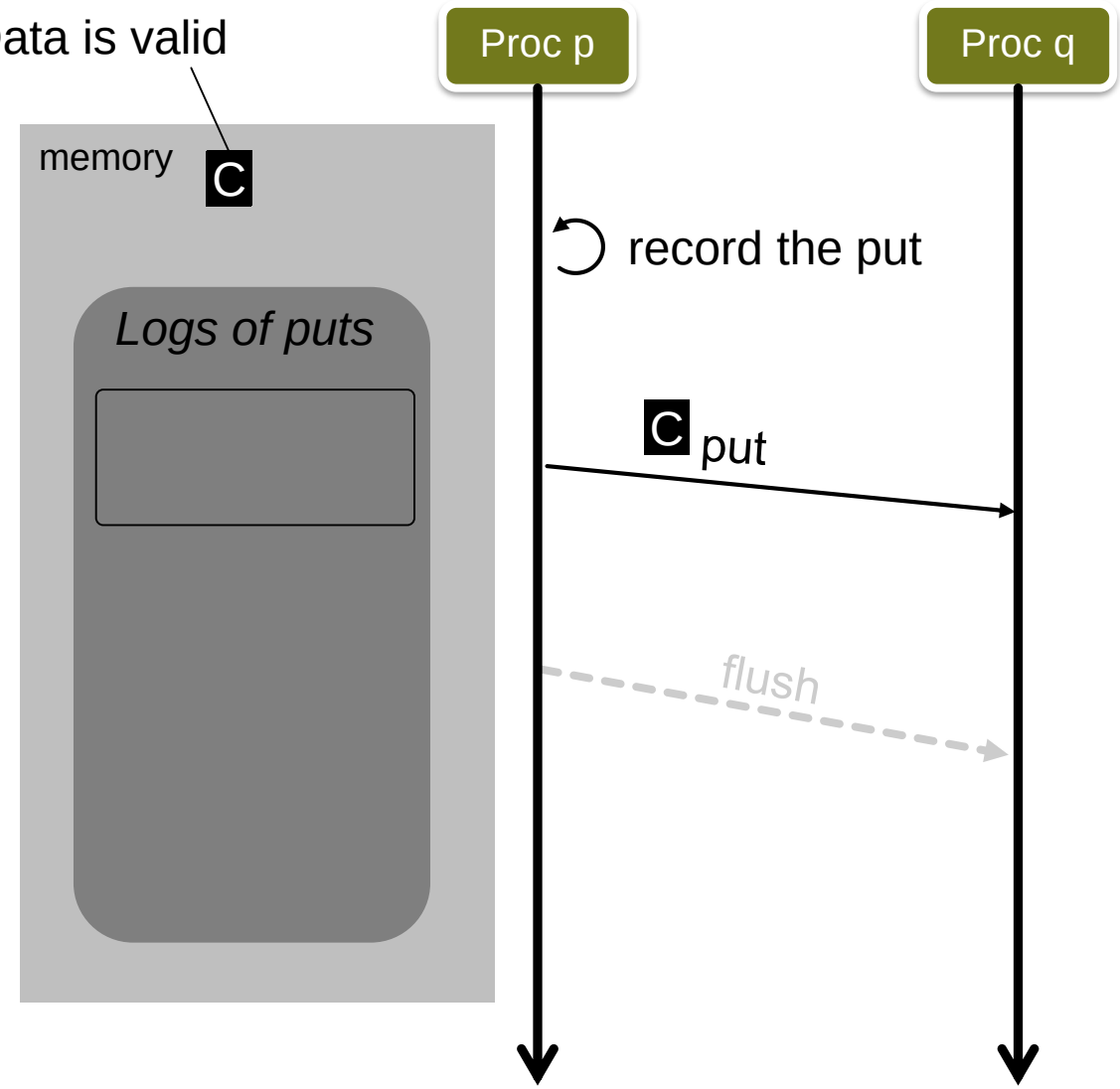
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



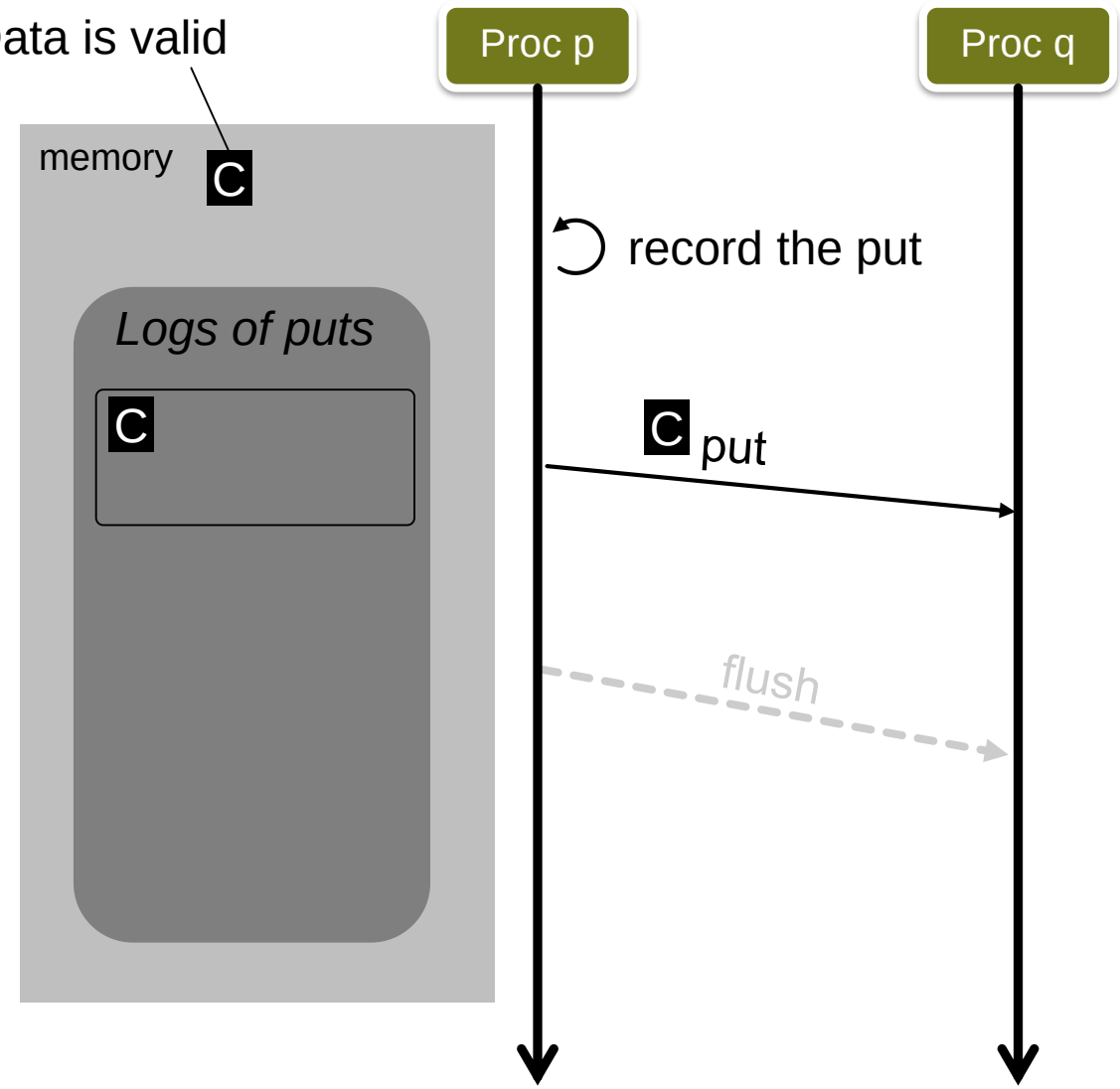
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



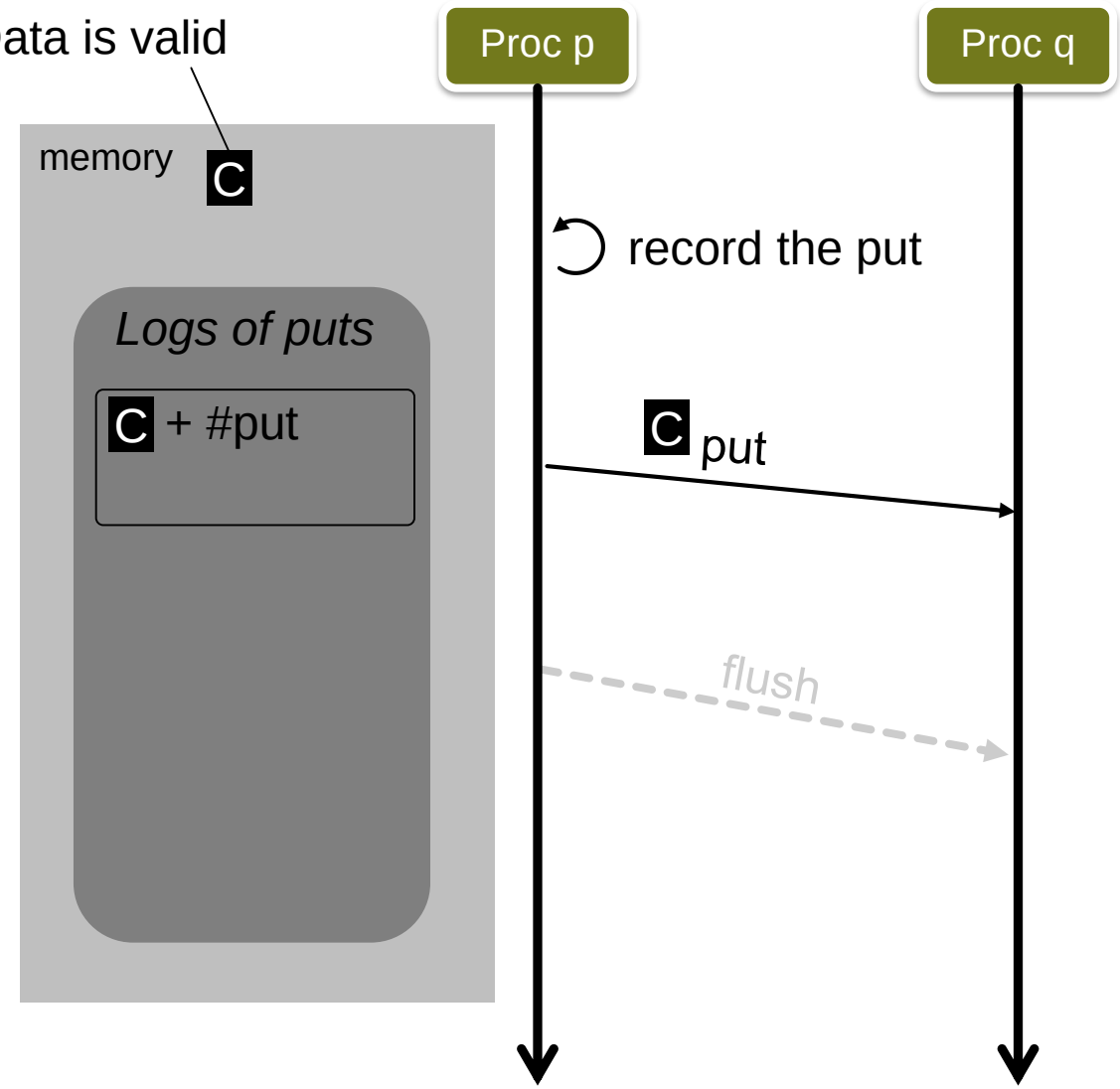
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



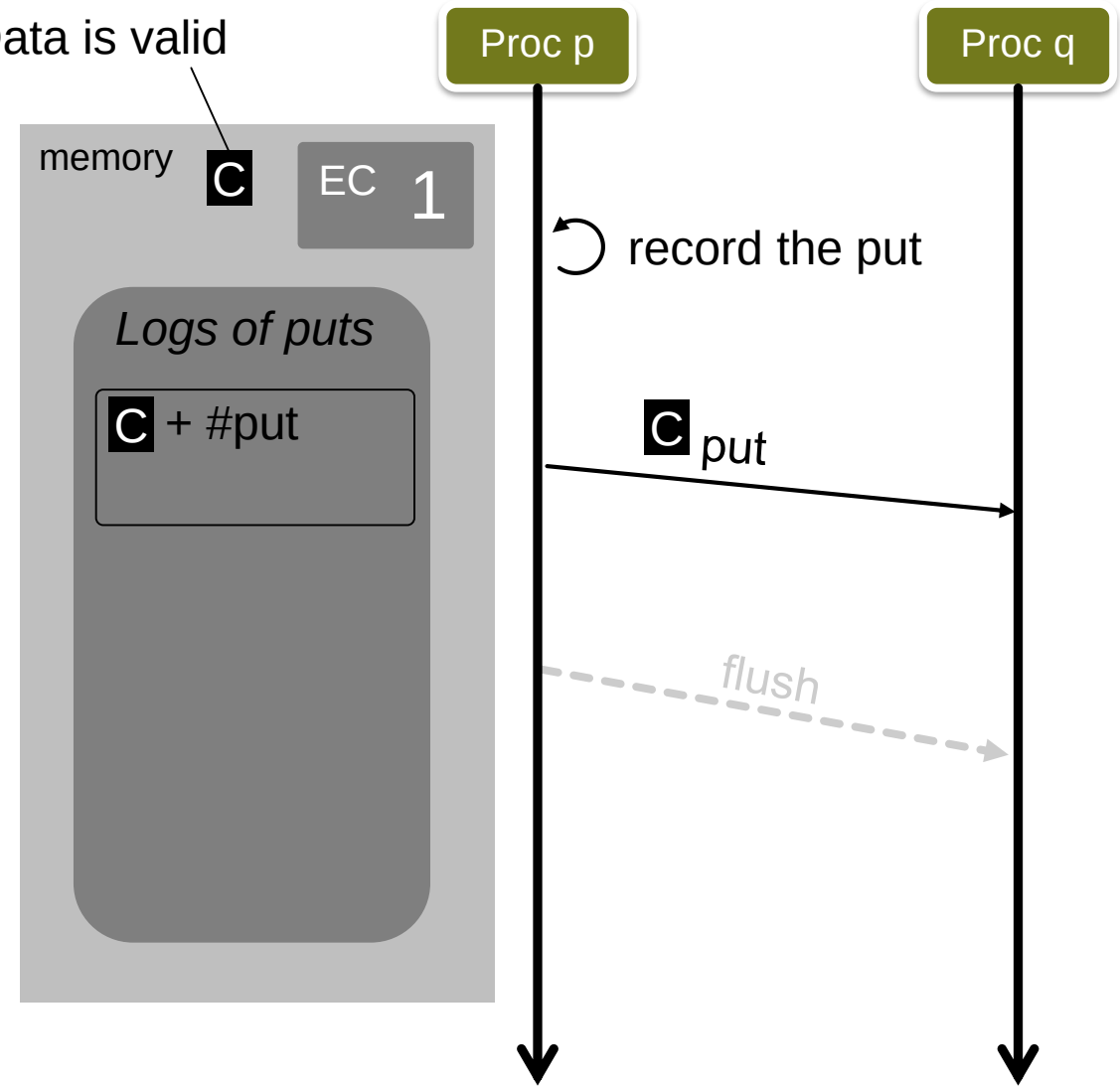
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



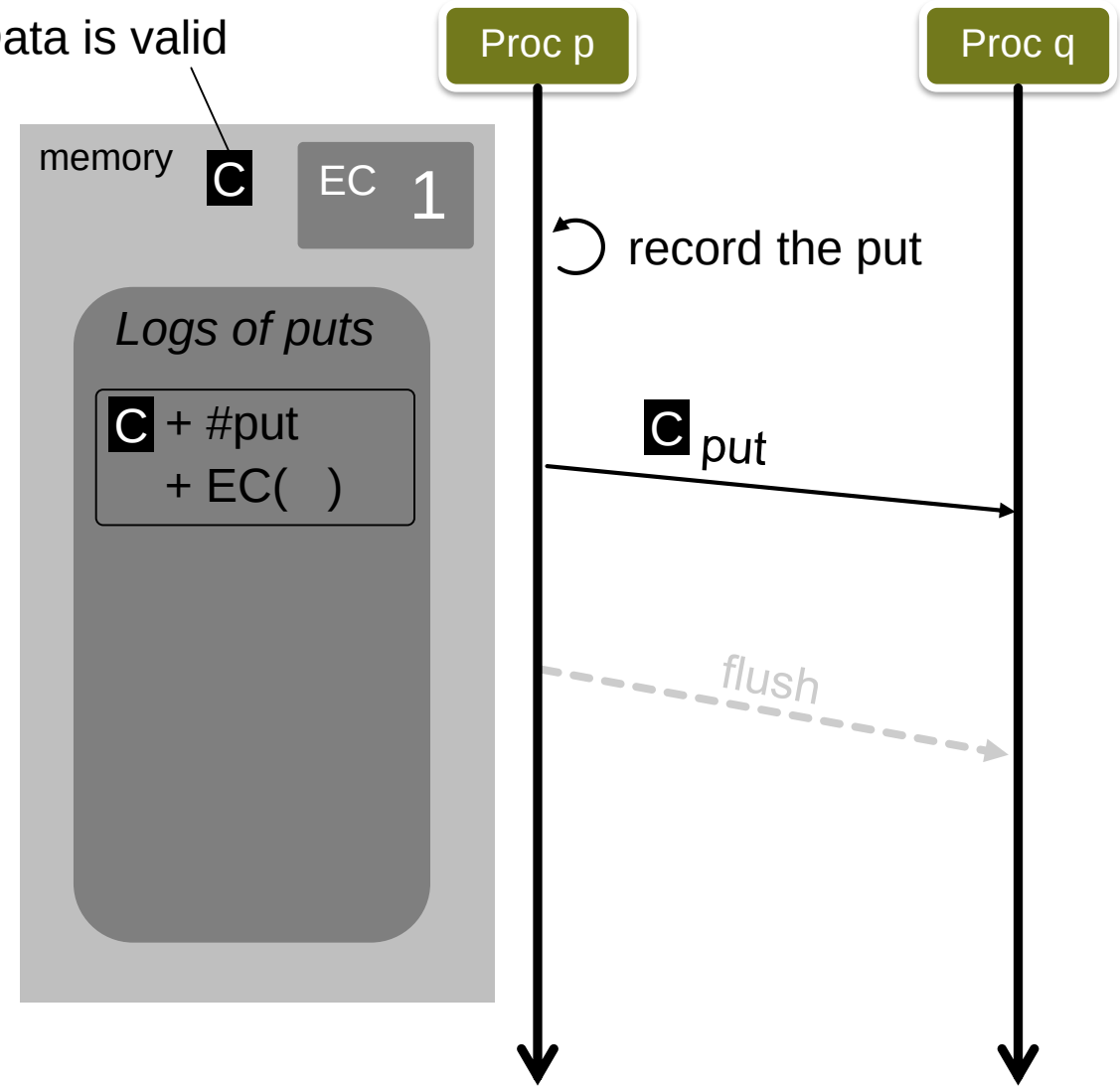
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



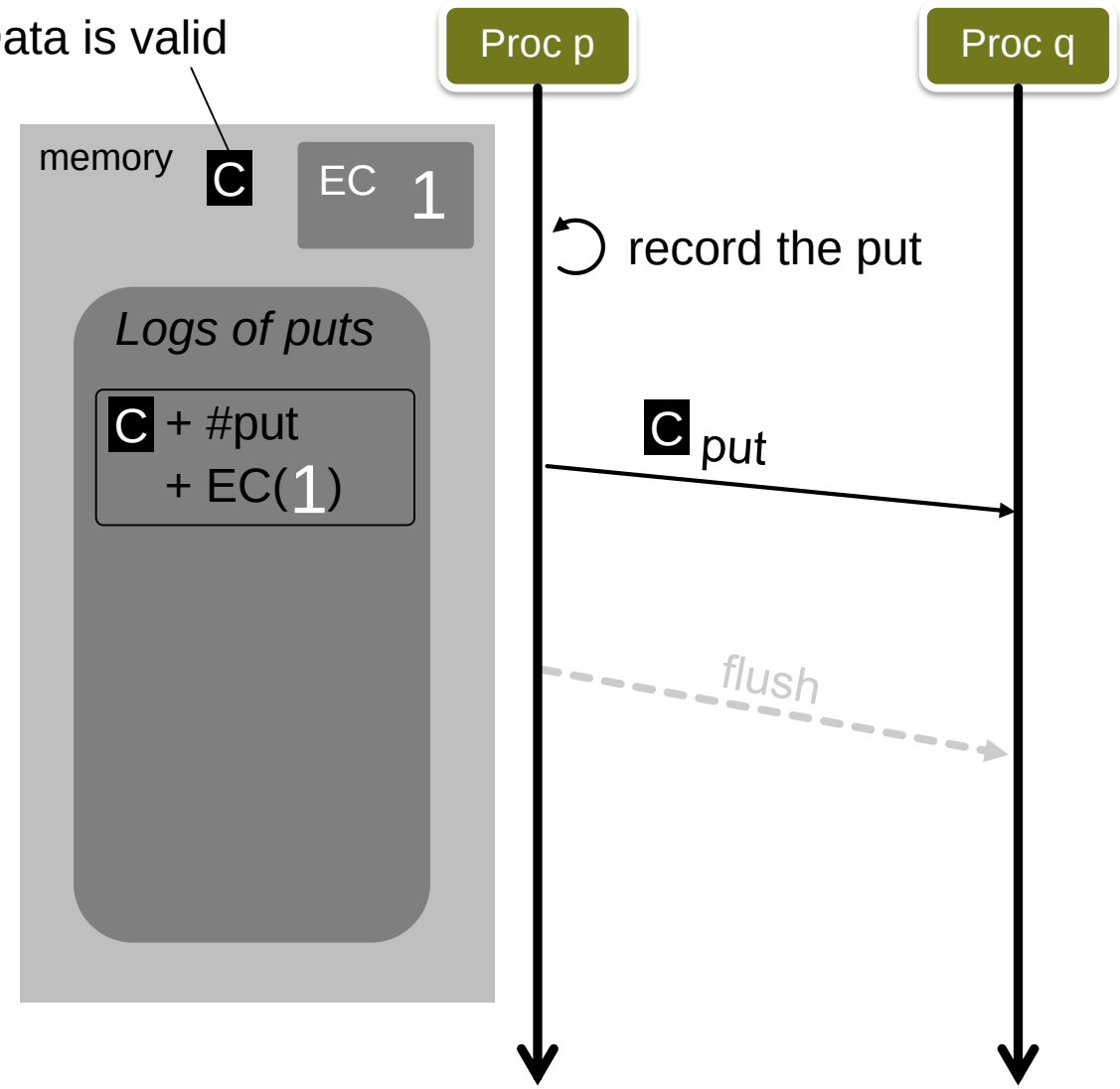
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



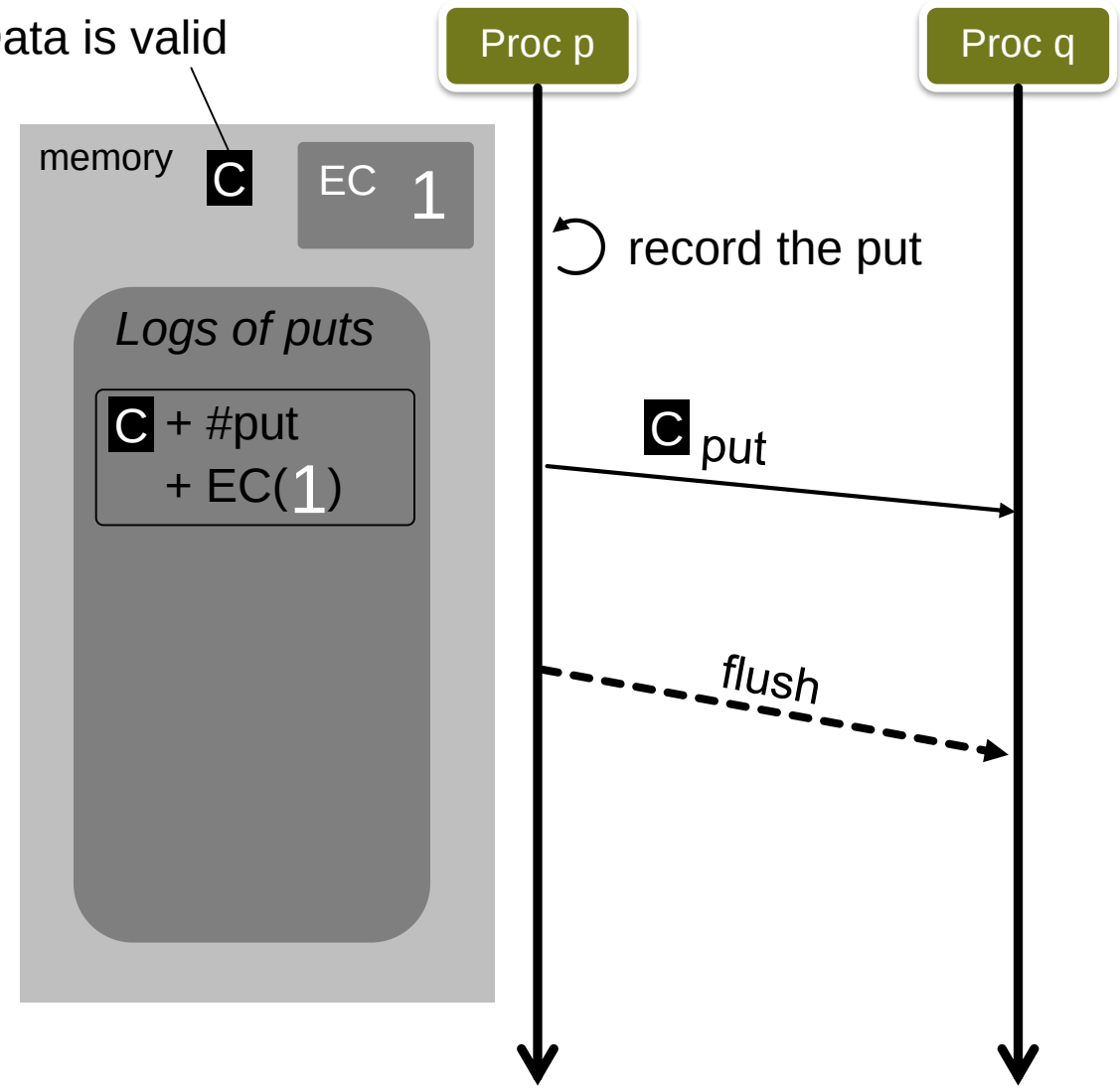
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



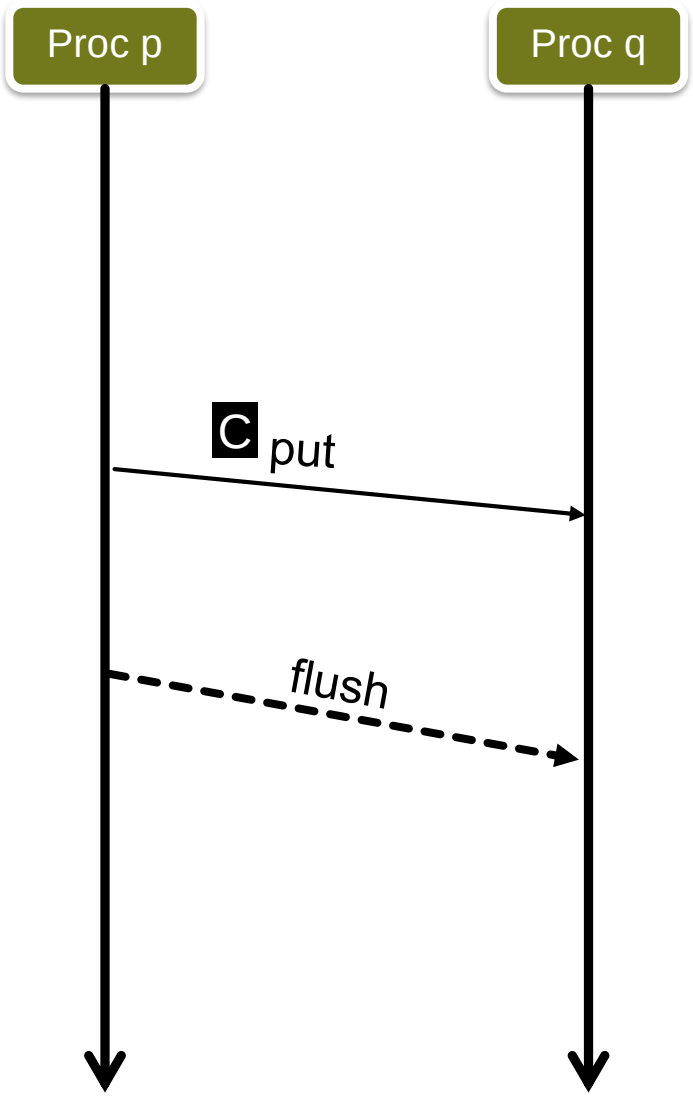
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



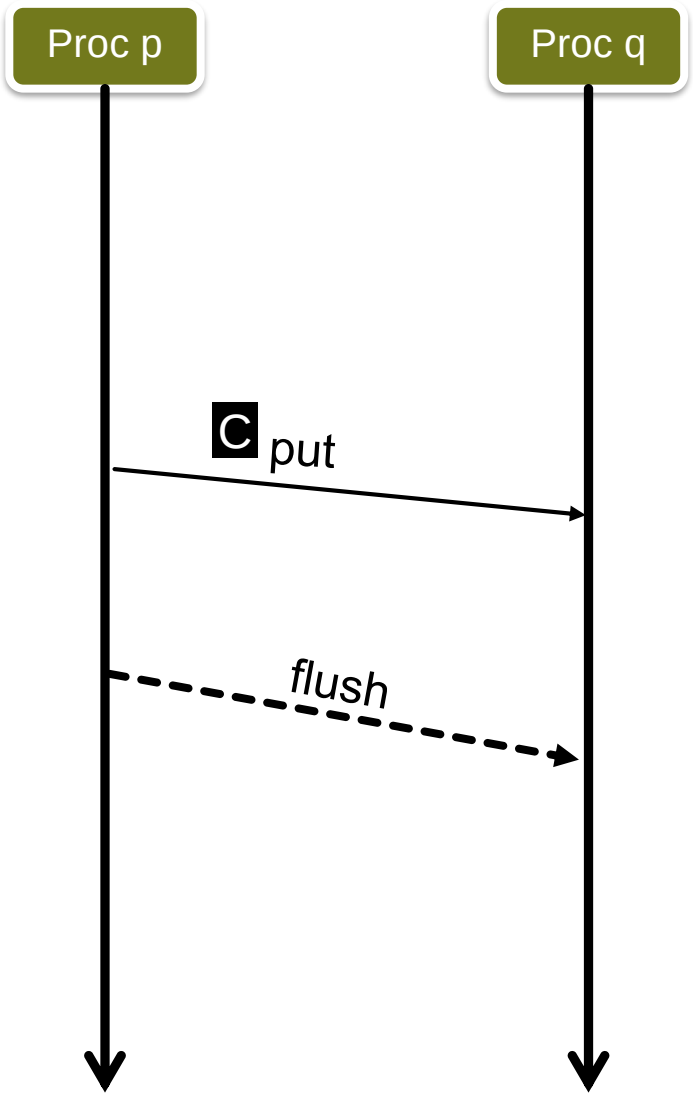
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



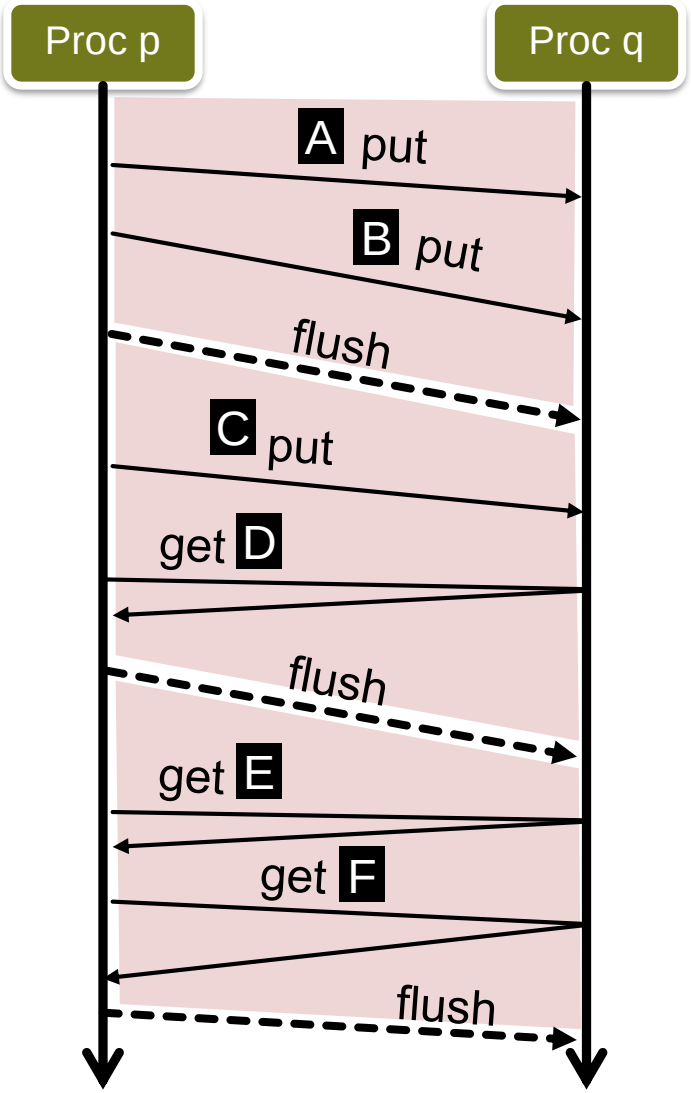
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



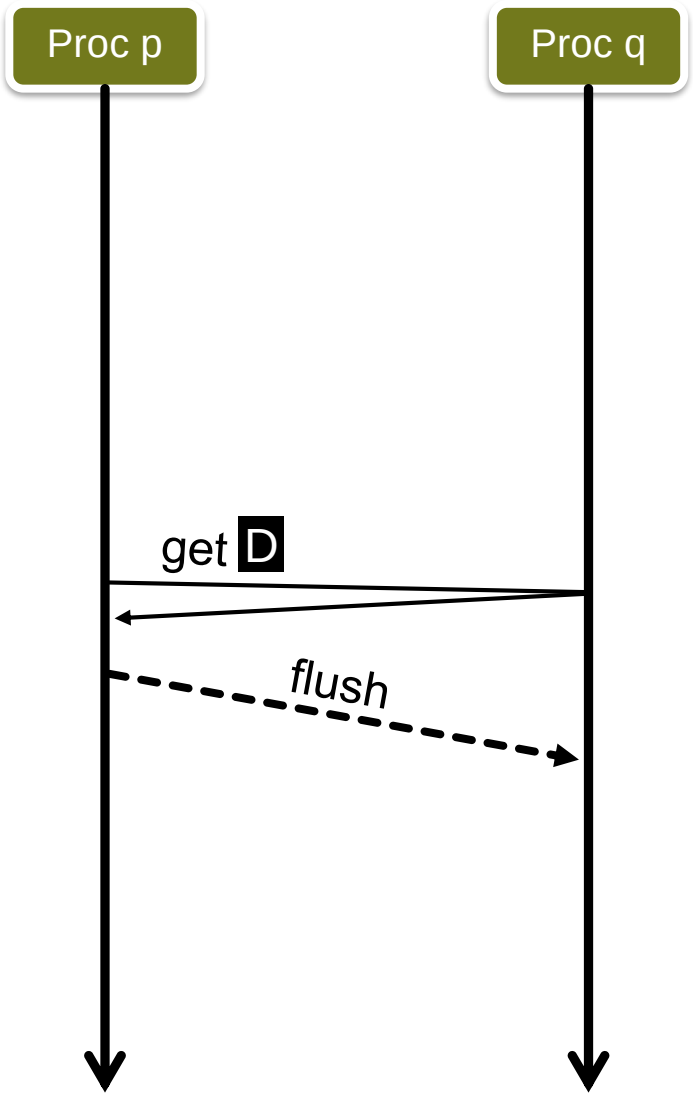
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



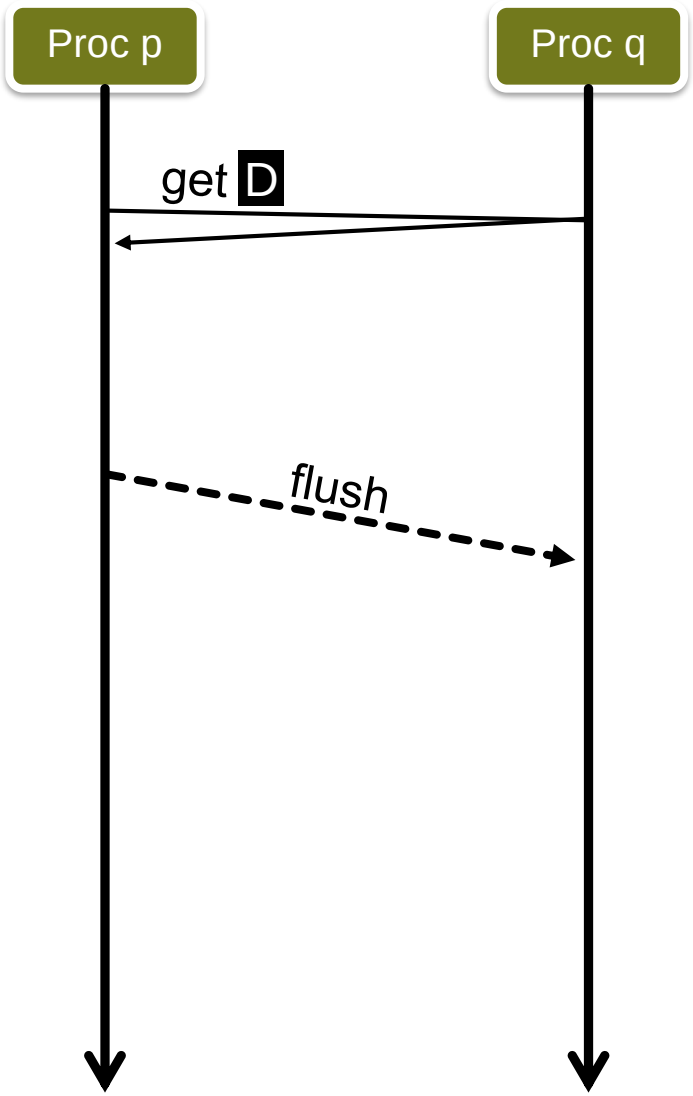
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



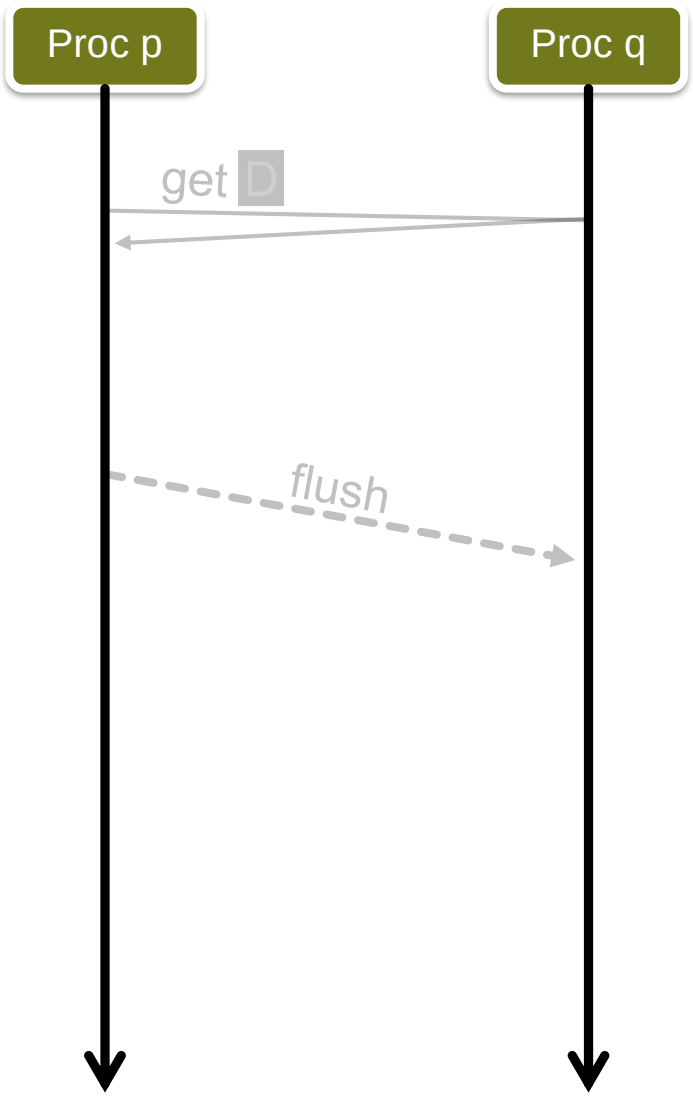
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



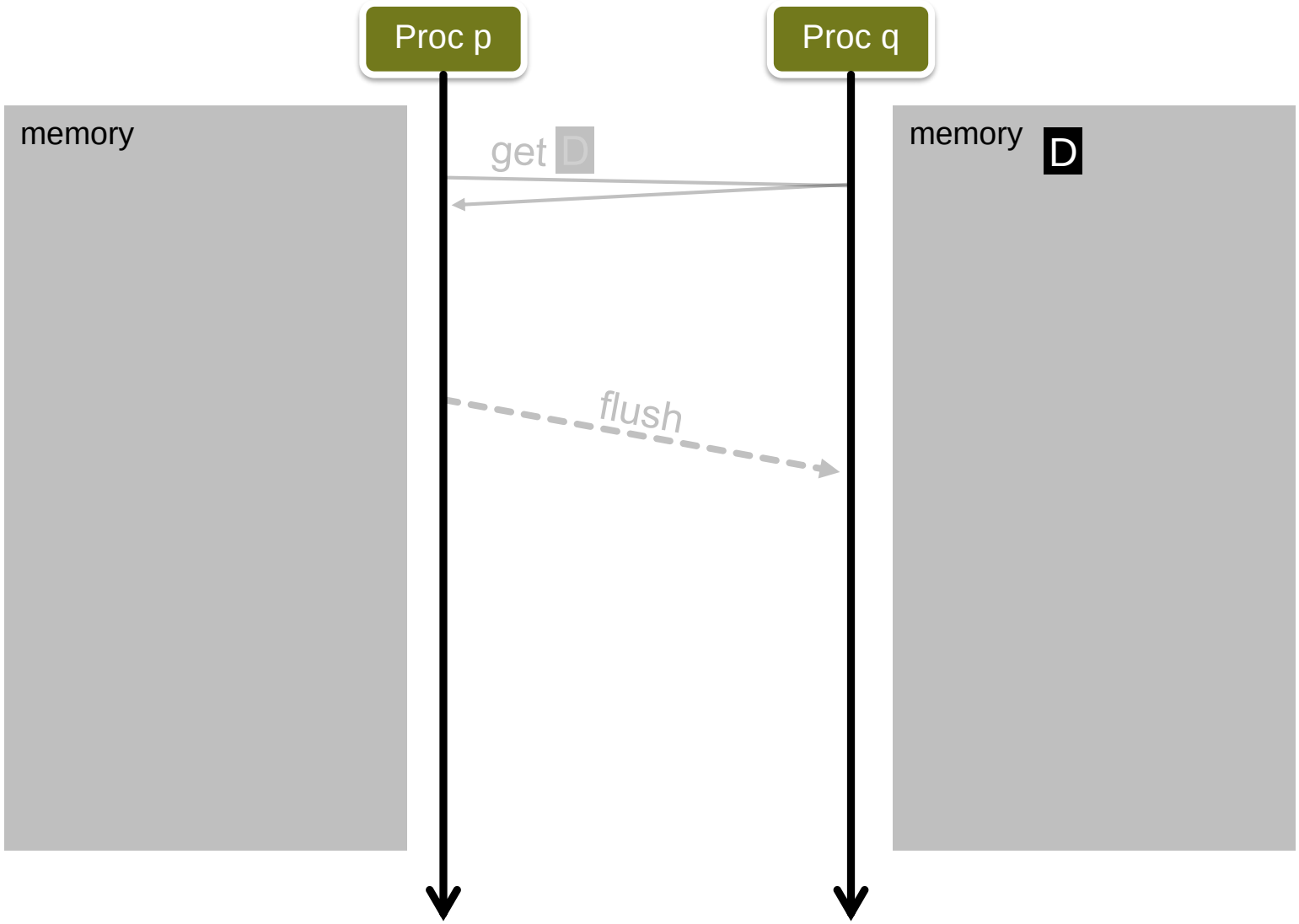
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



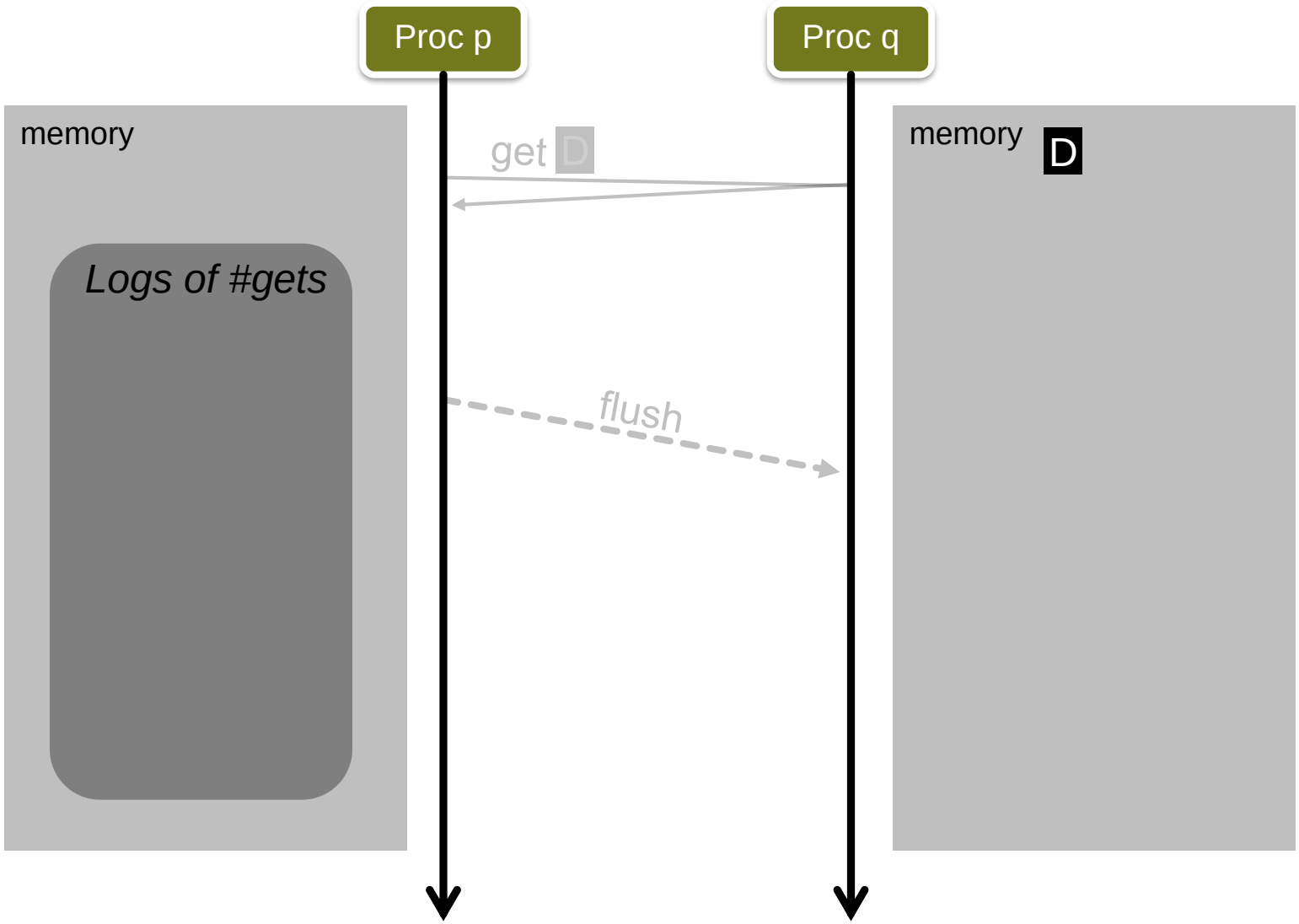
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



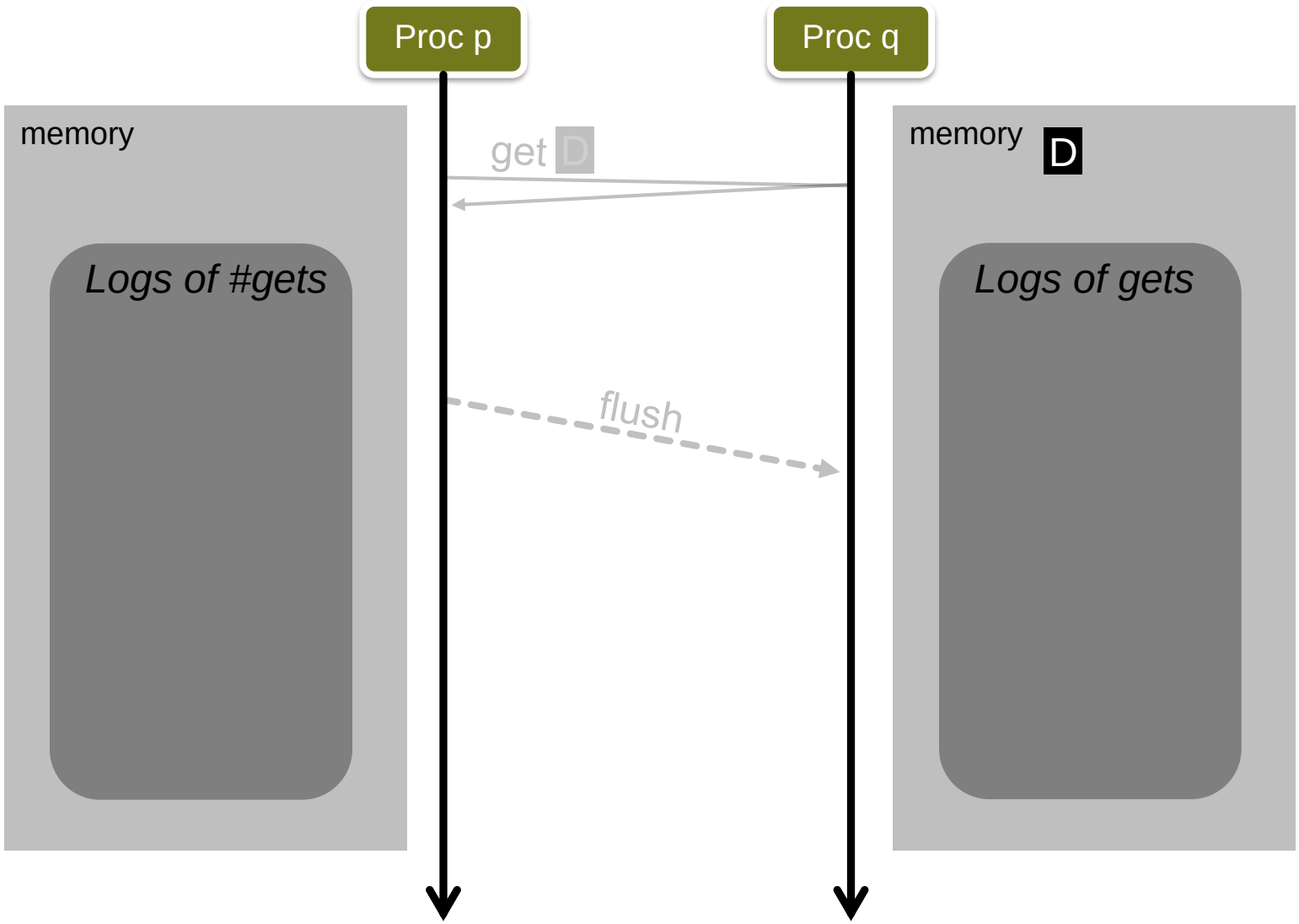
RMA: Logging Gets

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



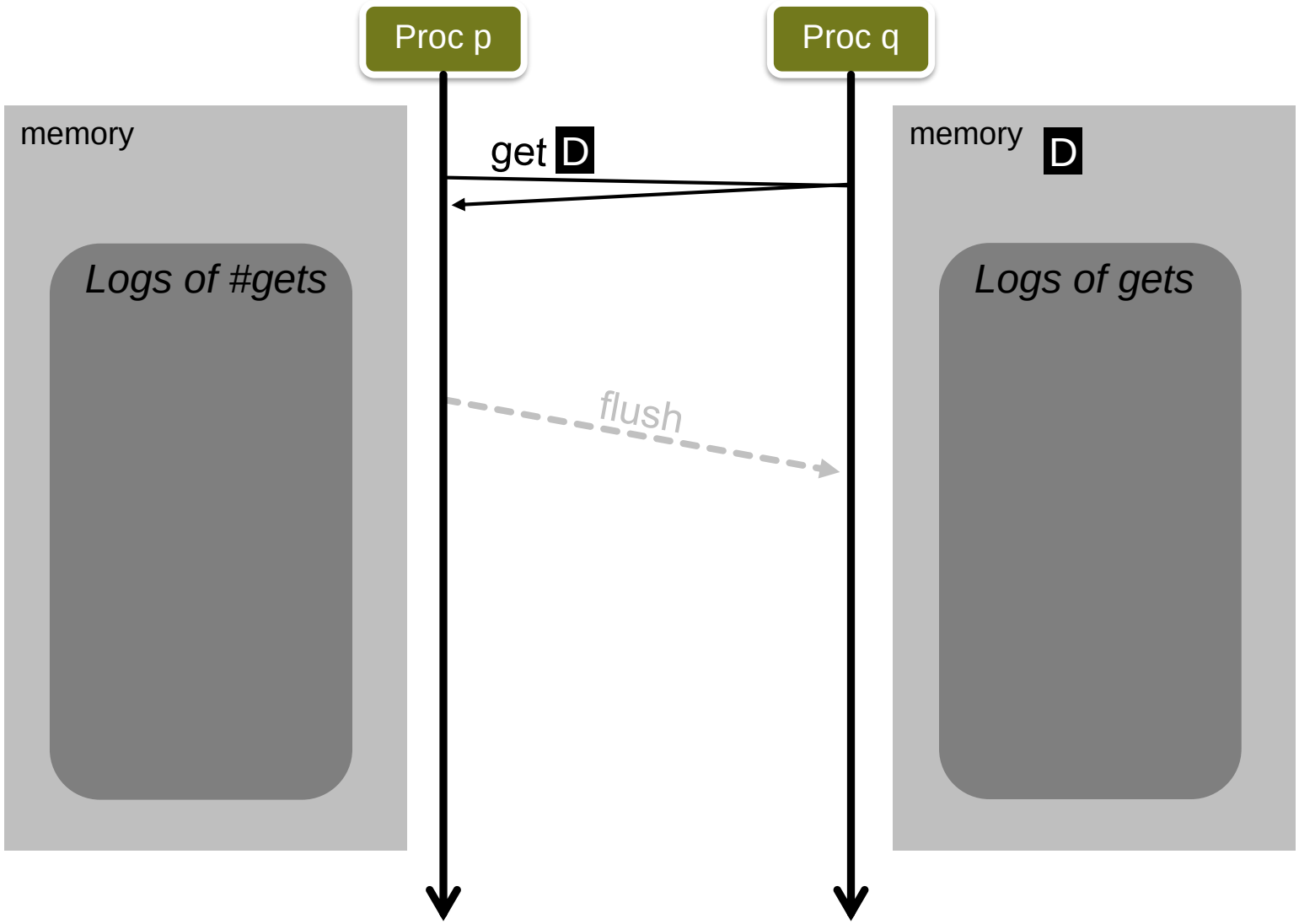
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



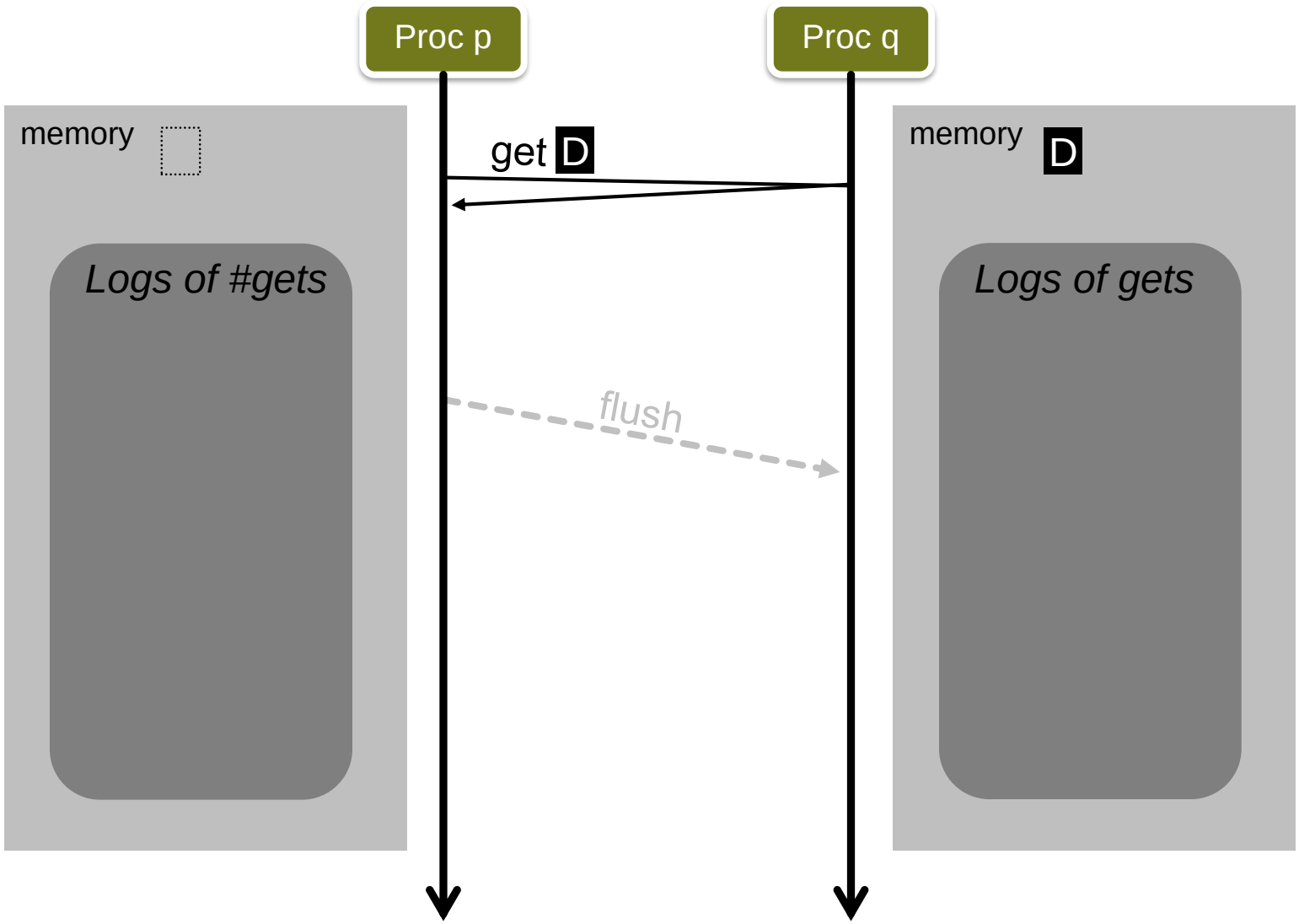
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



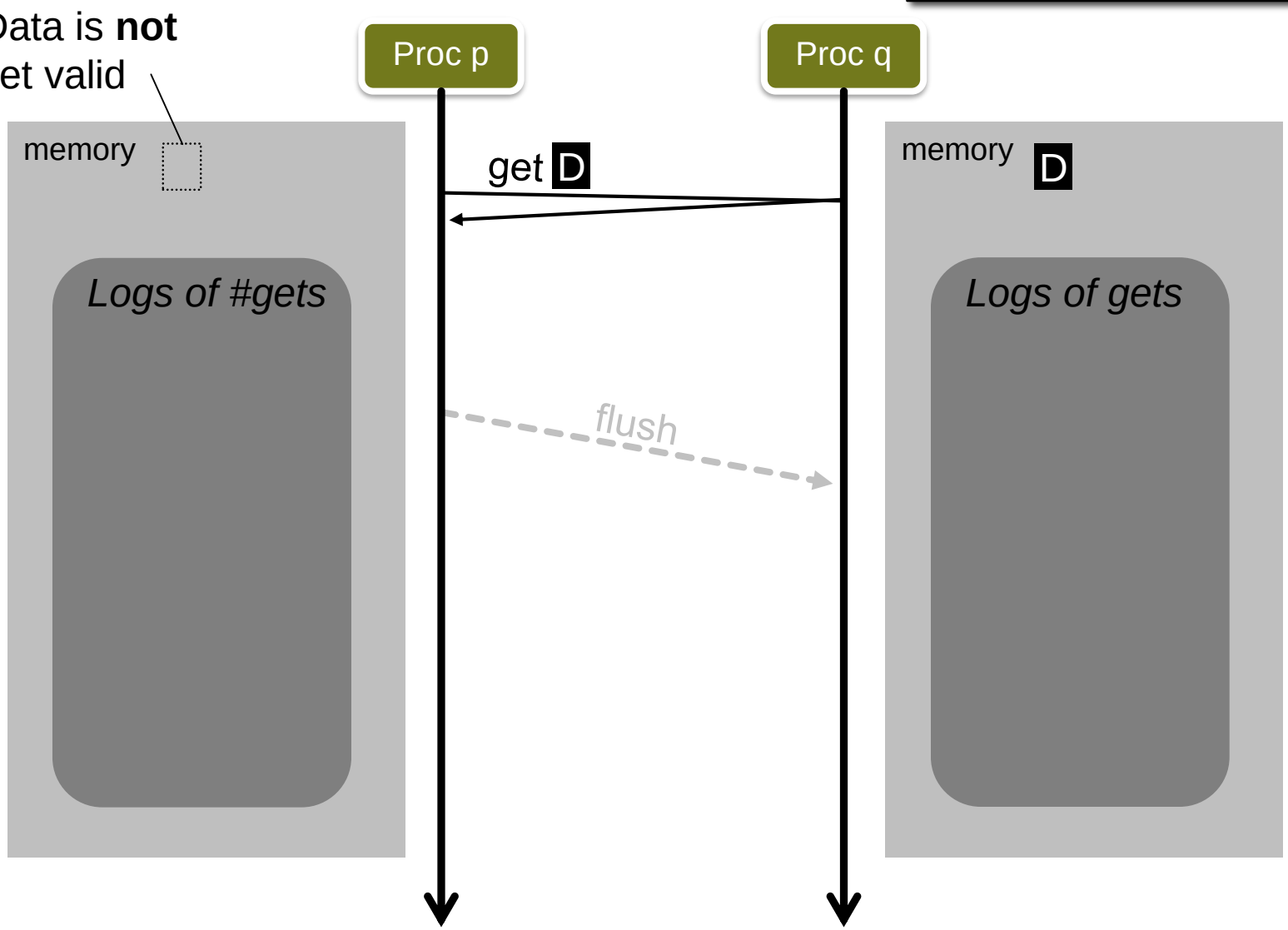
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



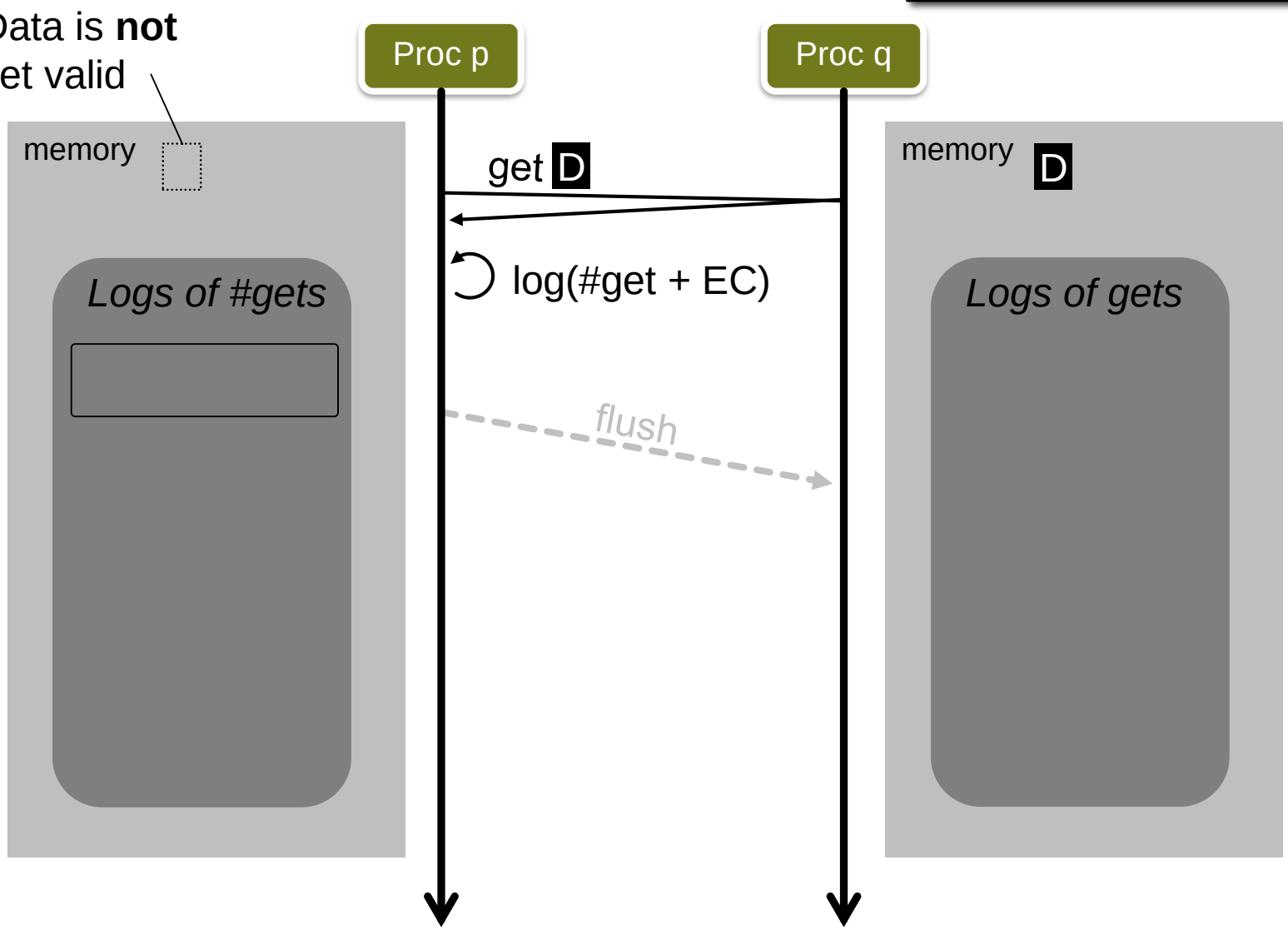
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



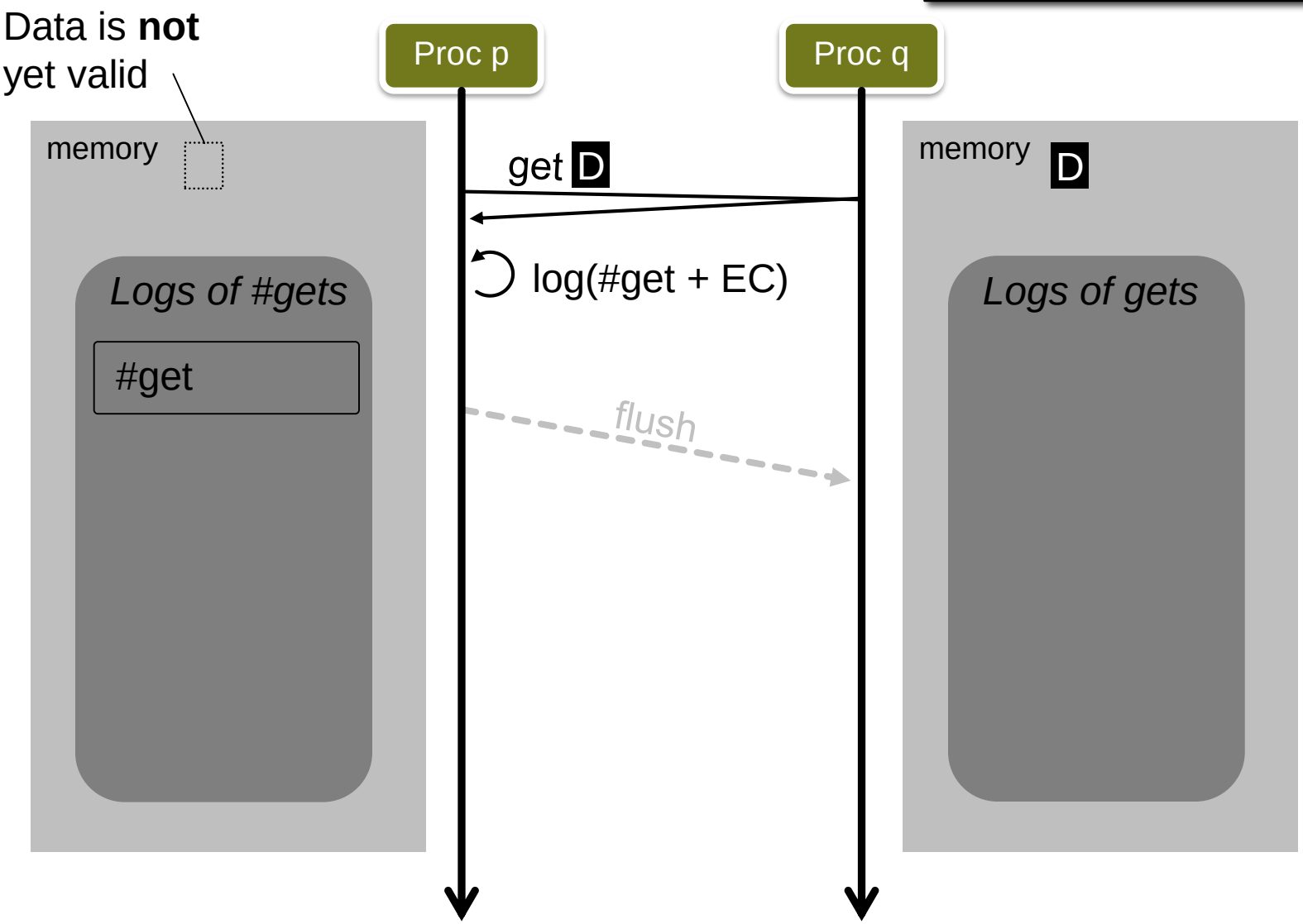
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



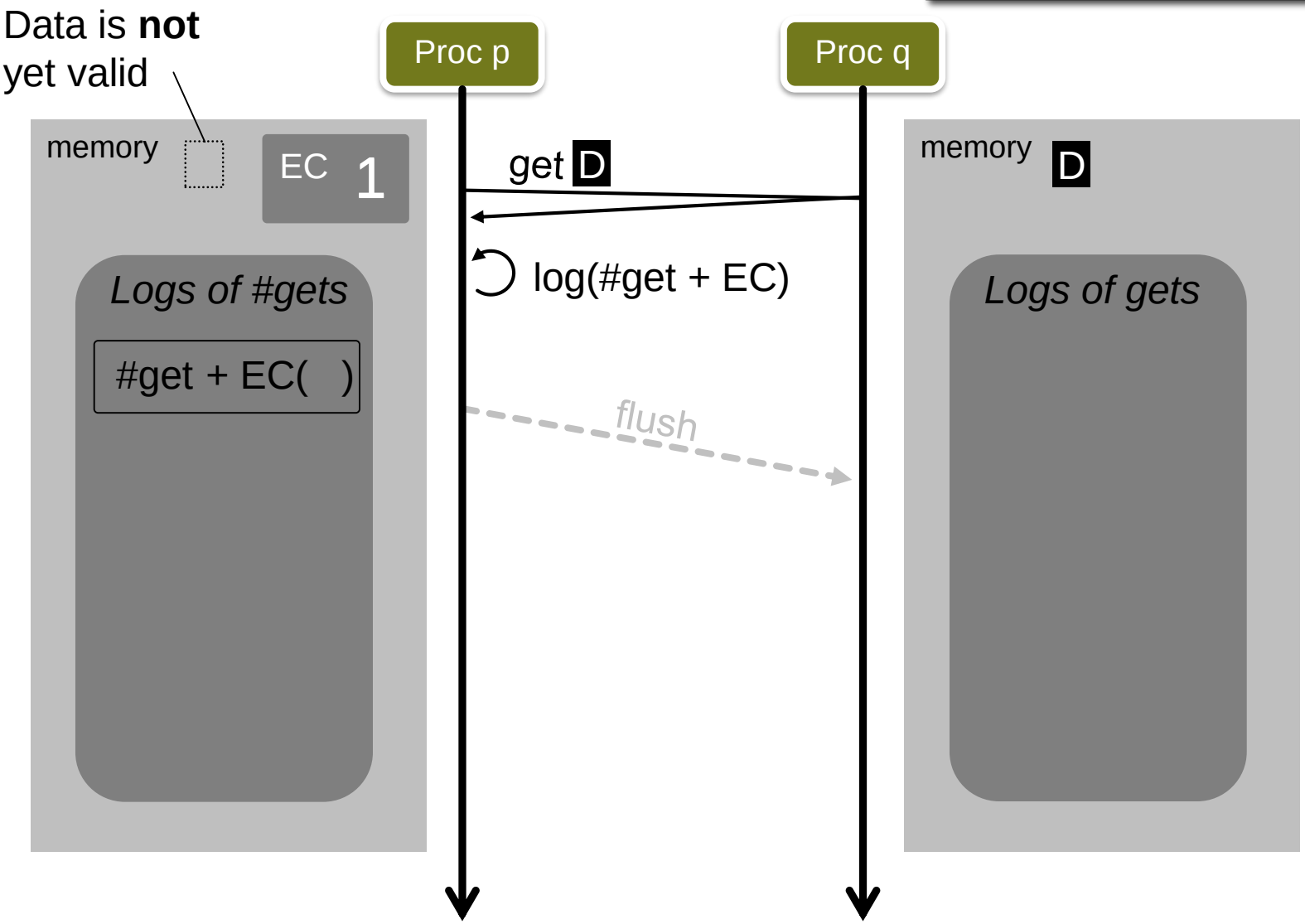
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



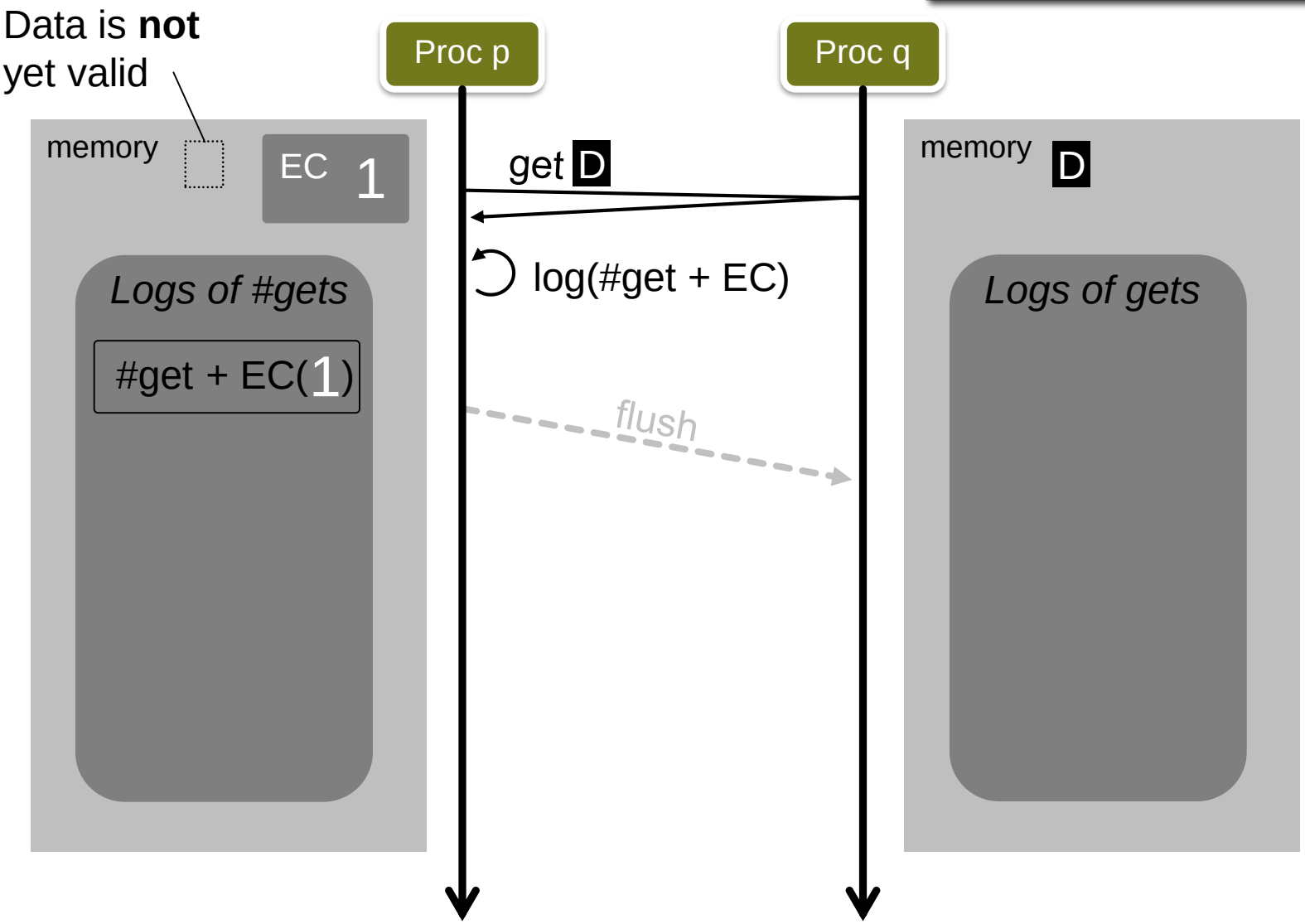
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



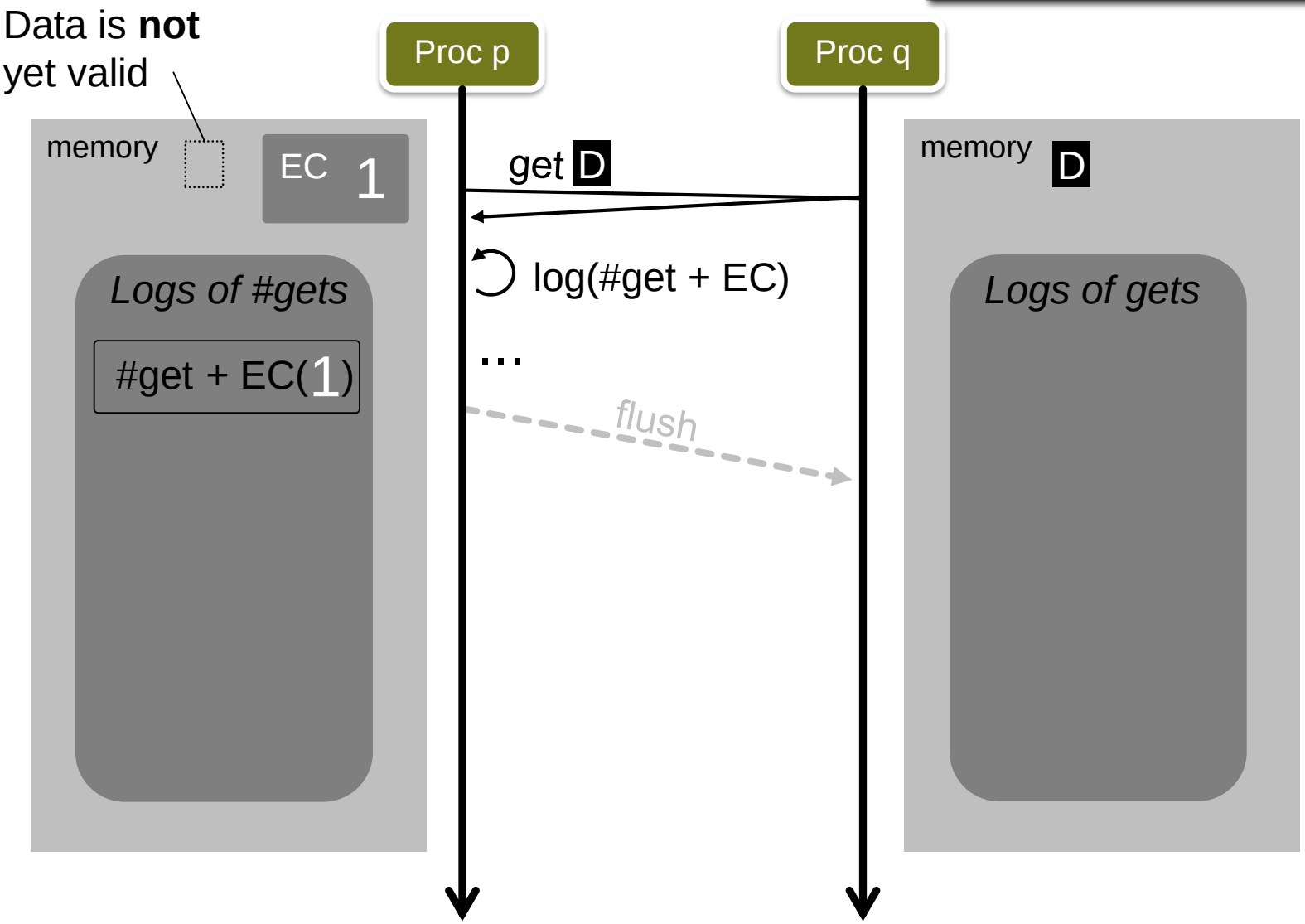
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



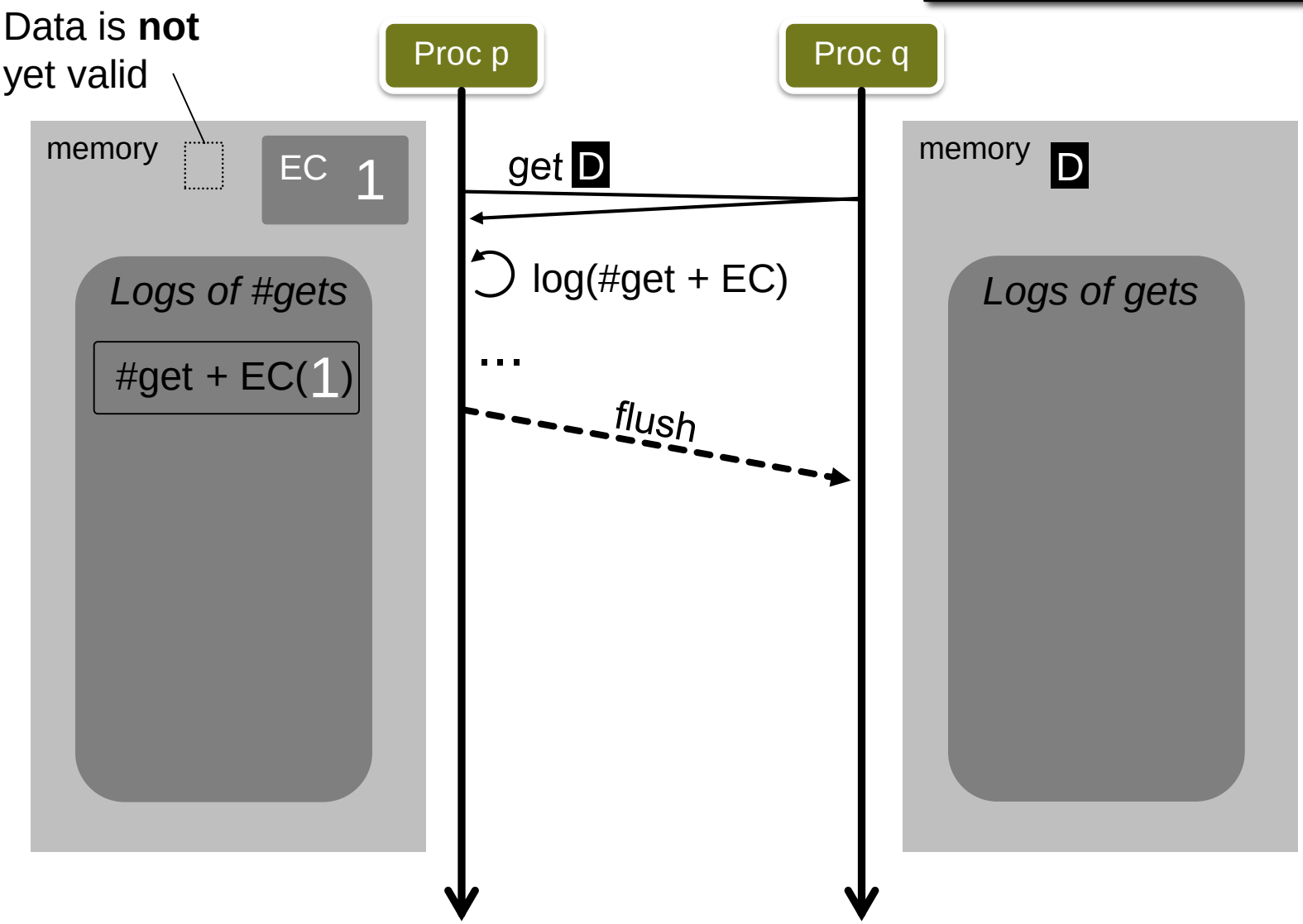
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



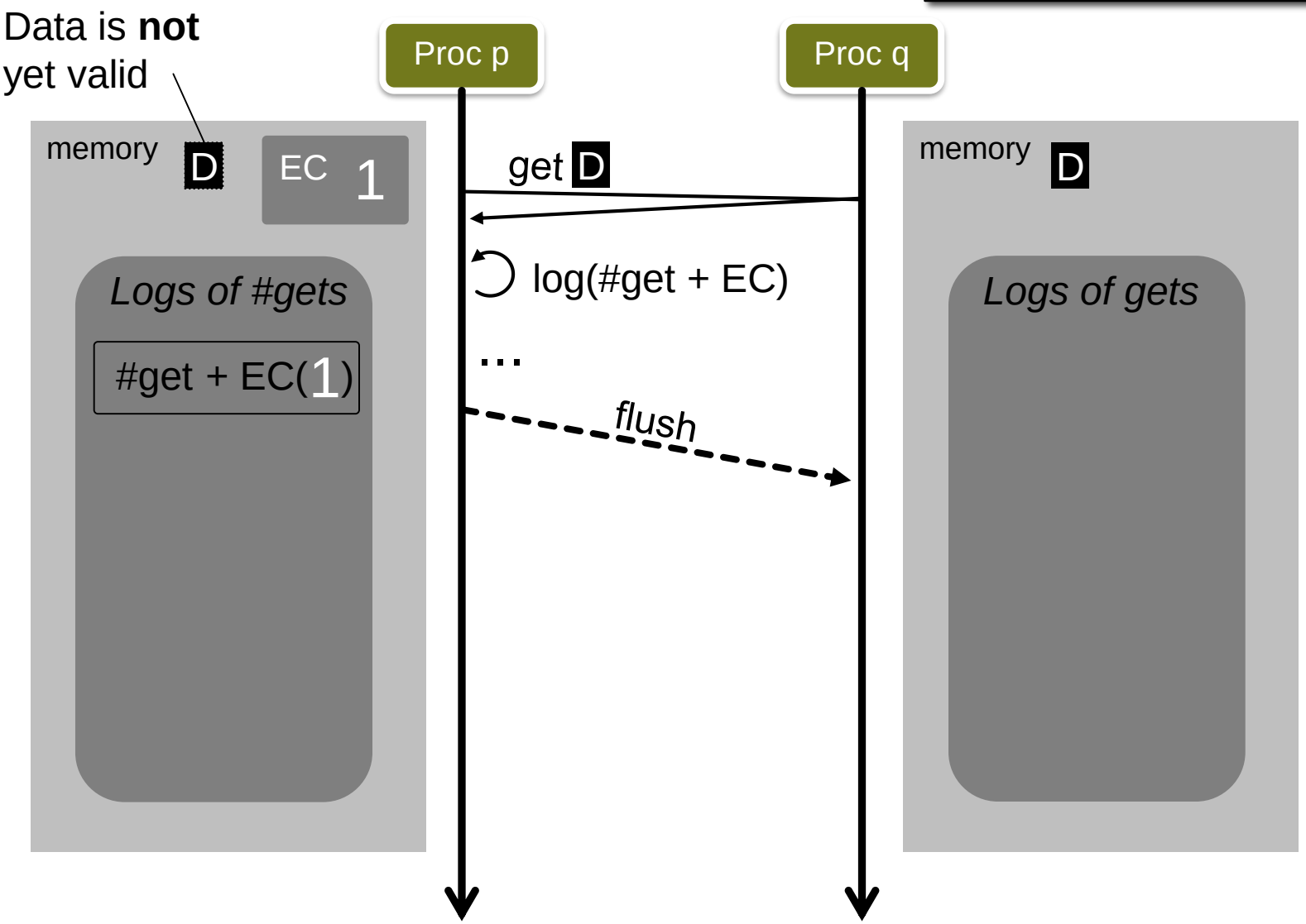
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



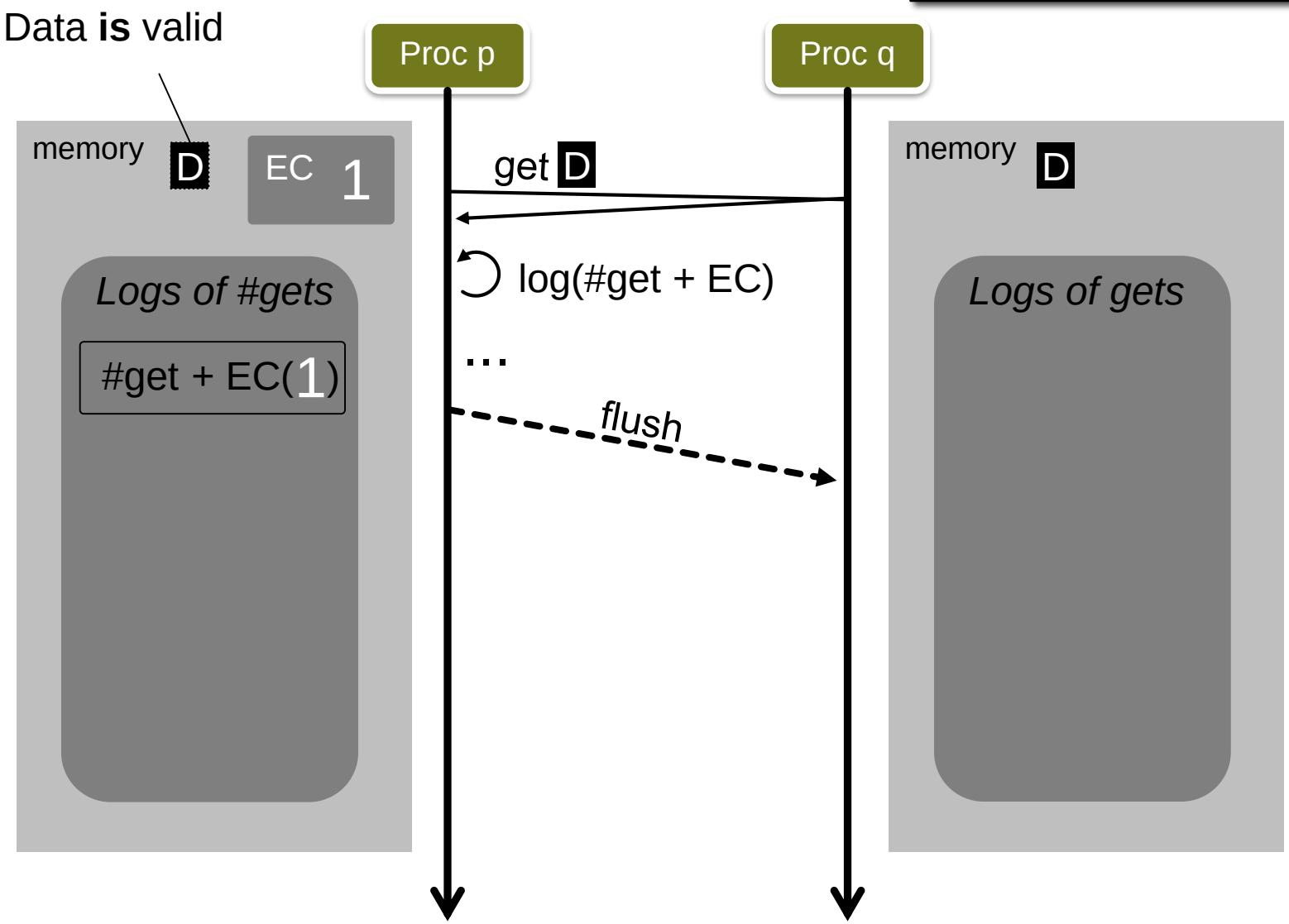
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



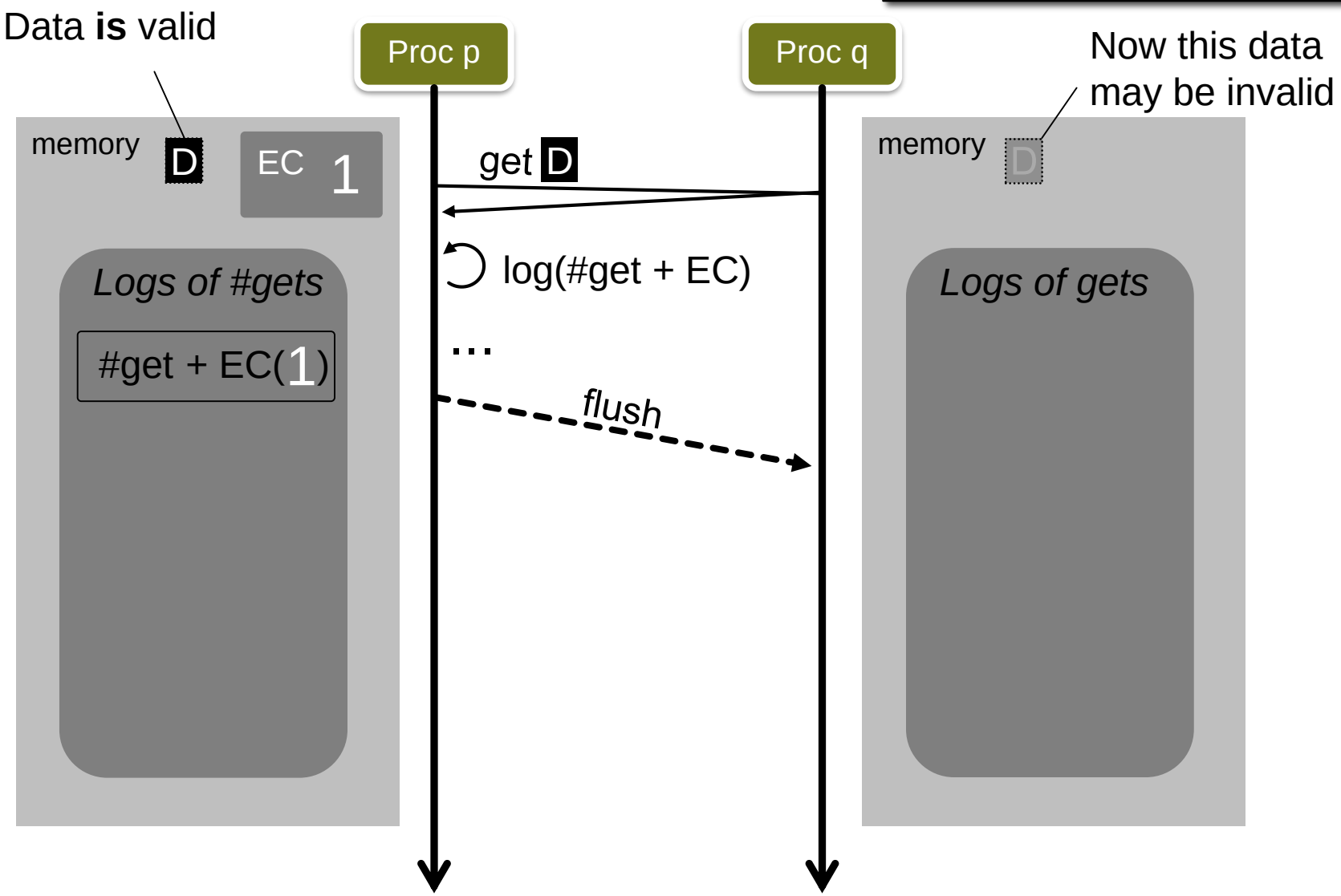
RMA: Logging Gets

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



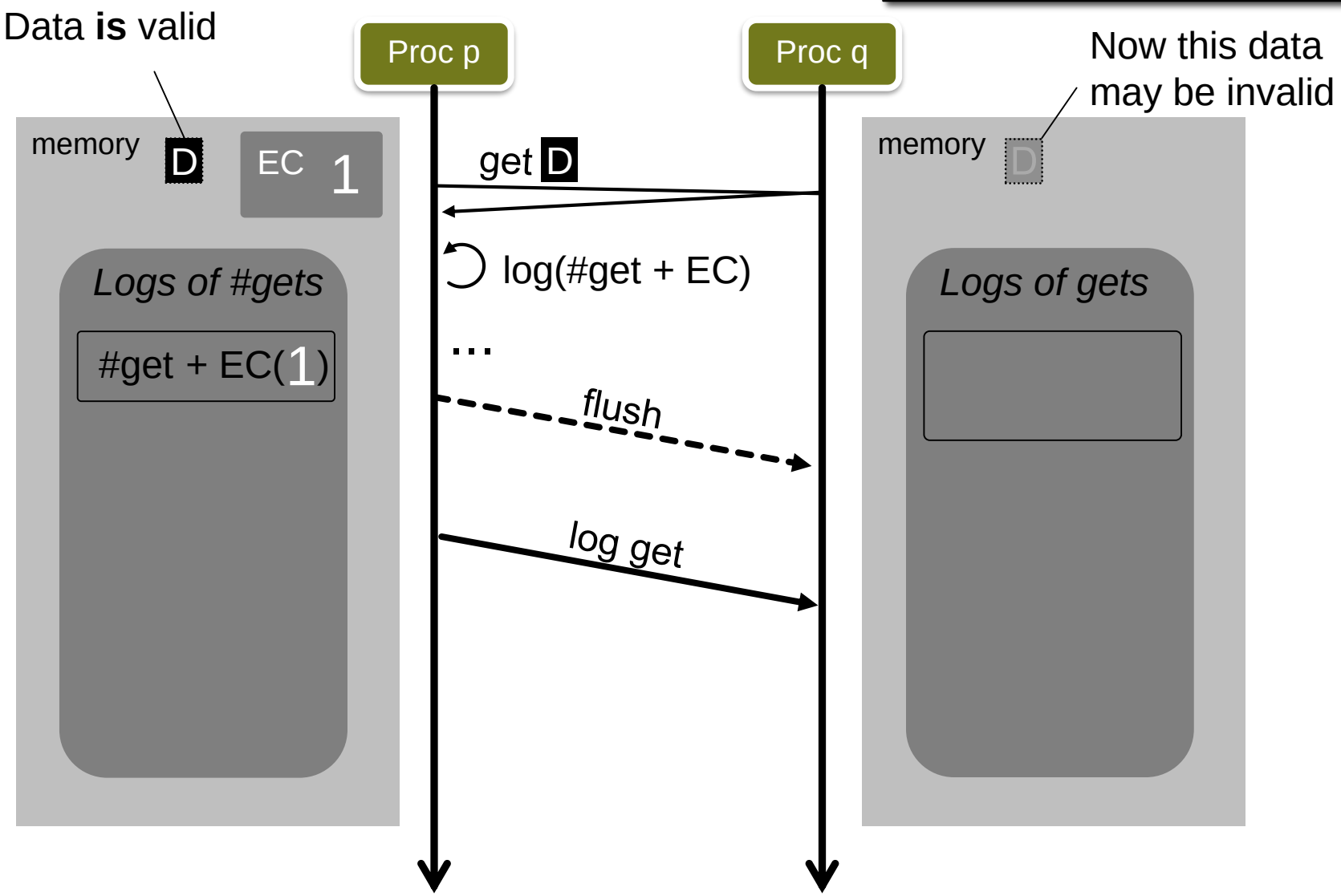
RMA: Logging Gets

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



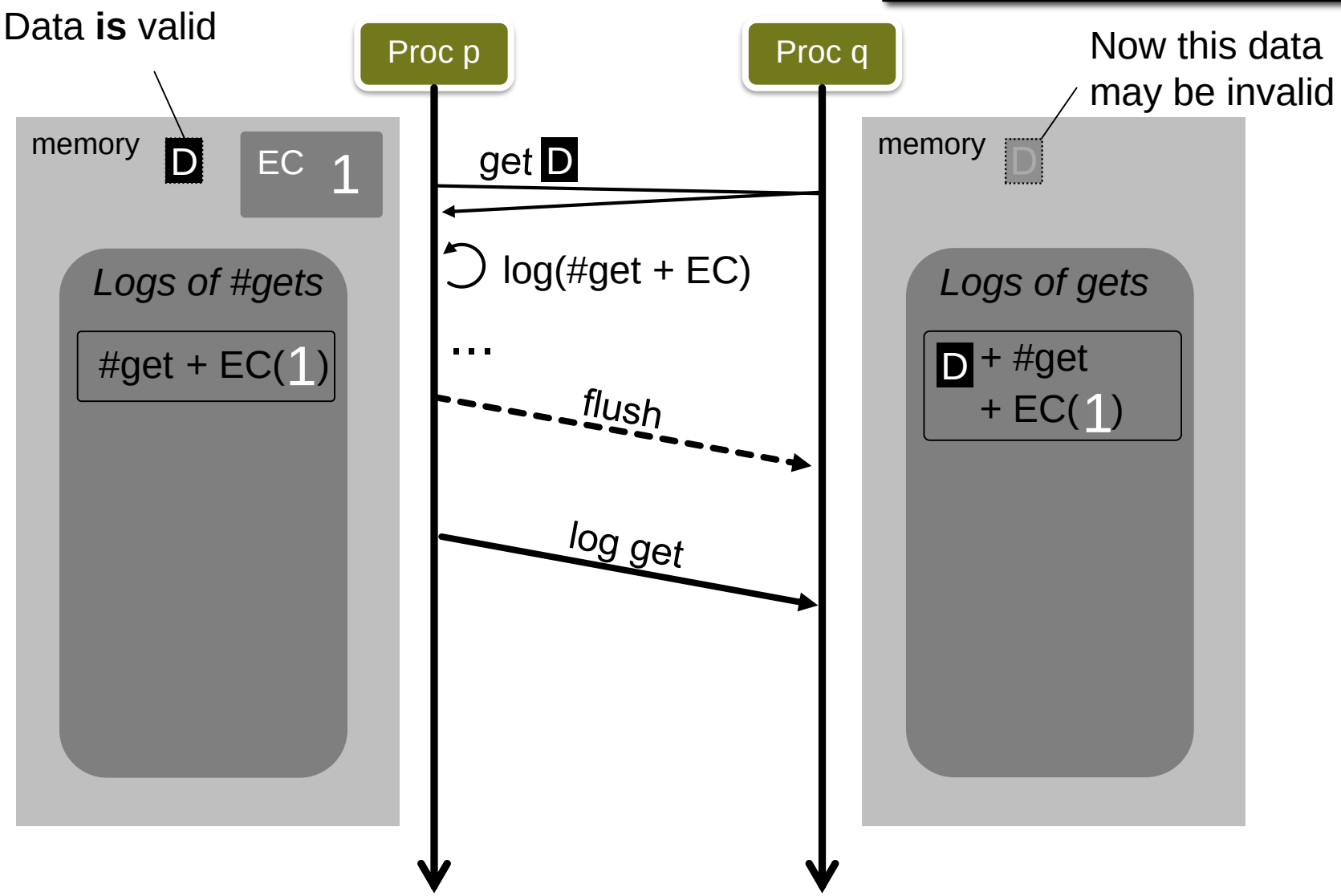
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



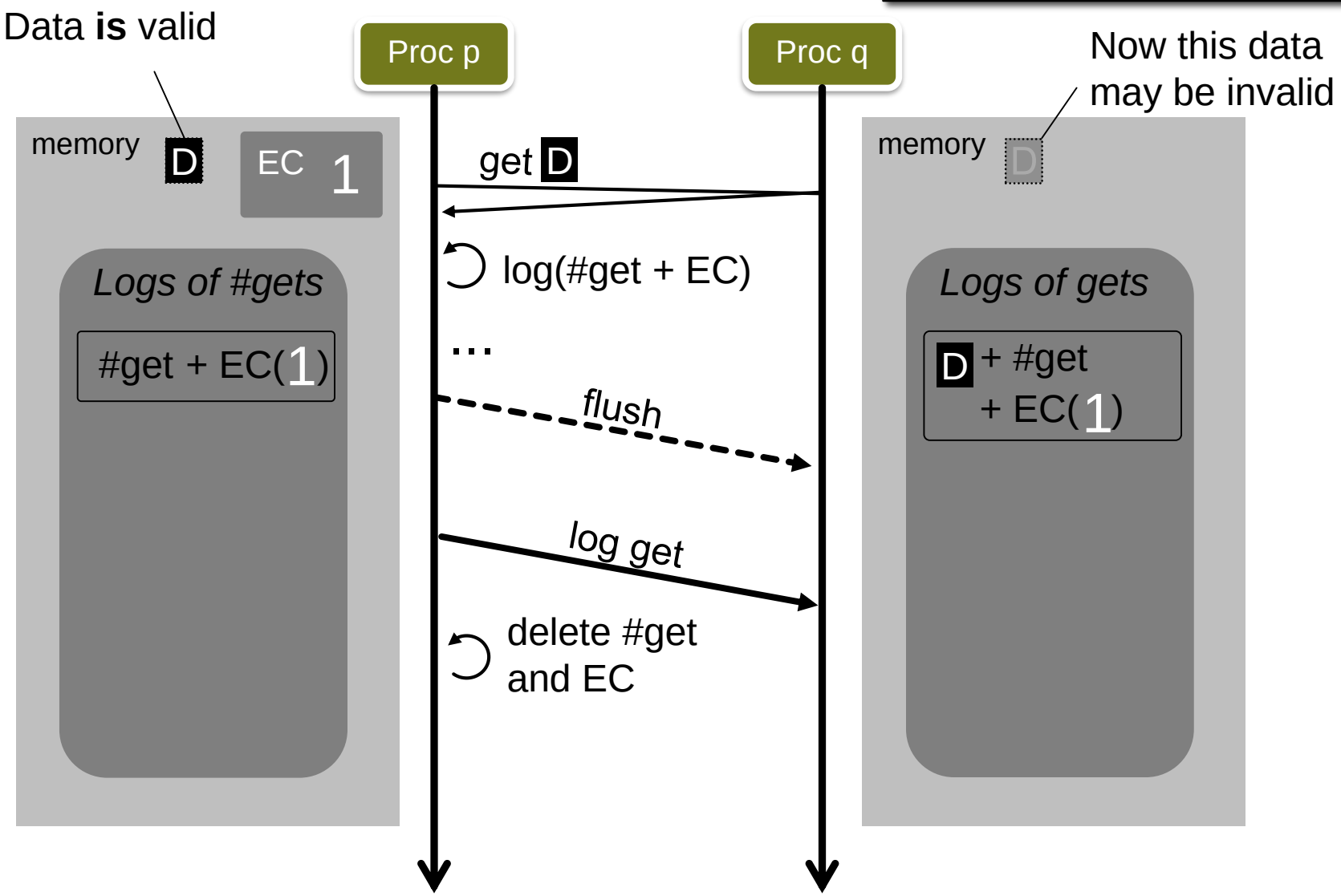
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



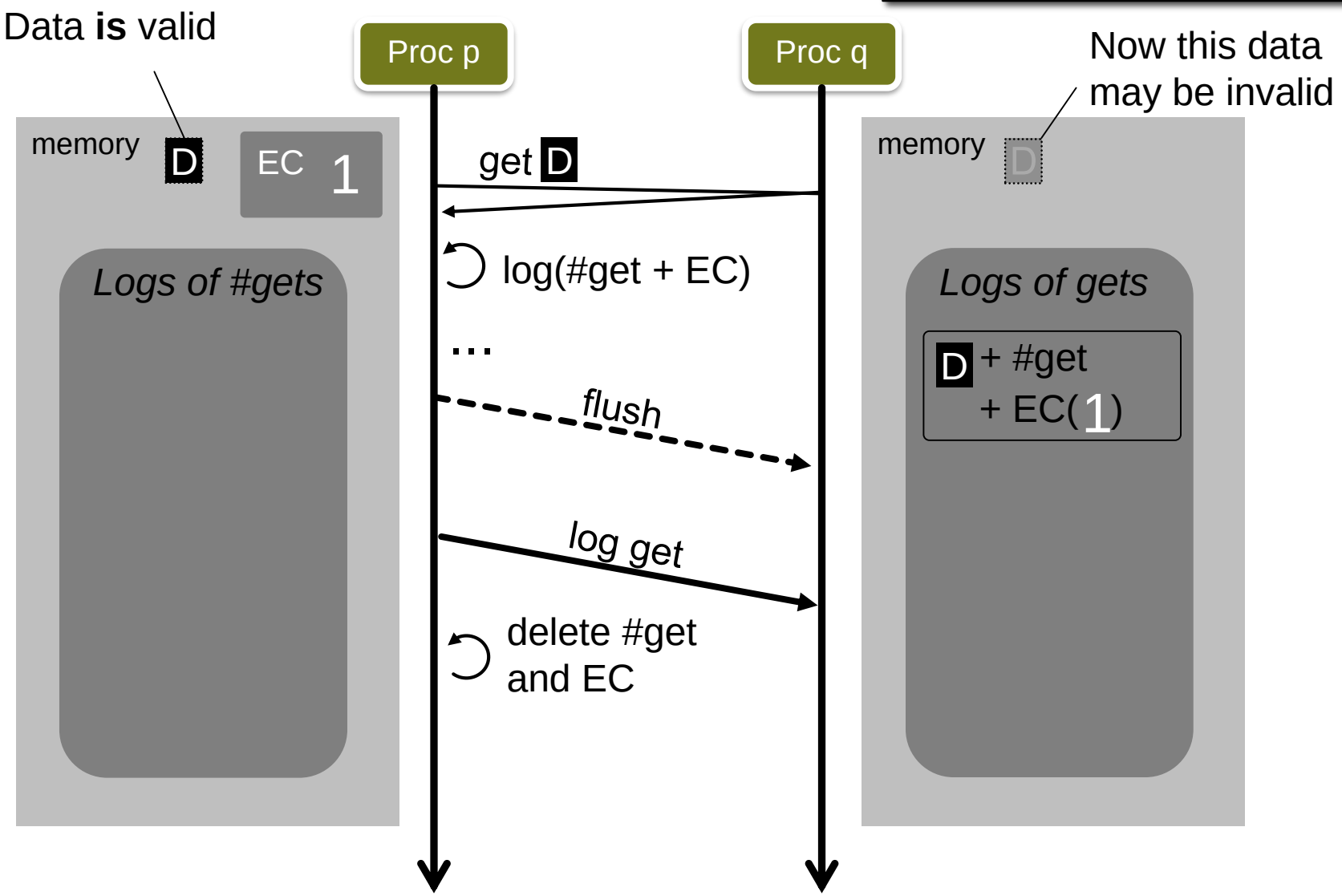
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$

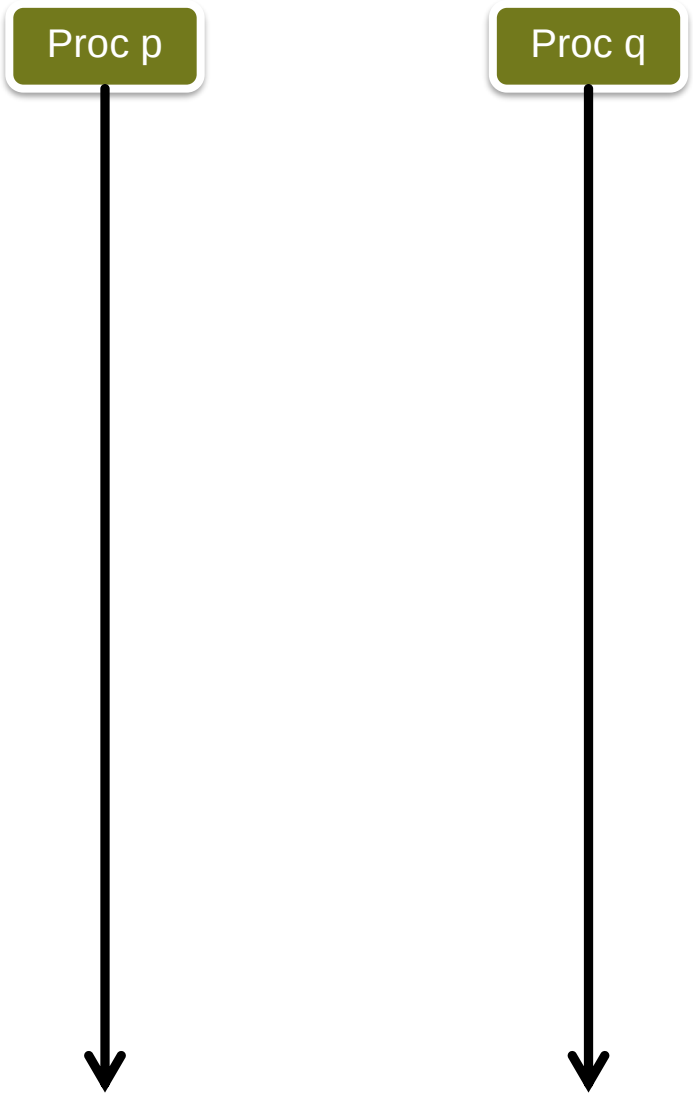


RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



RMA: Logging Gets



Overview of our Research

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

MP vs. RMA

Schemes

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Holistic fault-tolerance library

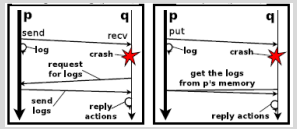
Design

Checkpoints on demand

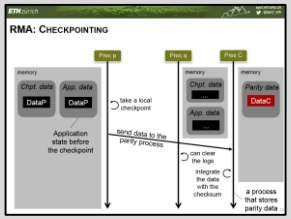
Optimizations

Performance

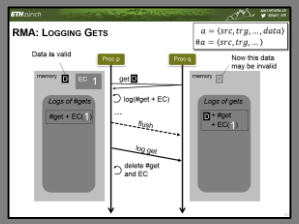
UC in RMA



MP vs. RMA



Checkpointing schemes



Logging accesses

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

Distribution of processes

Decreasing failure prob.

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

MP vs. RMA

Schemes

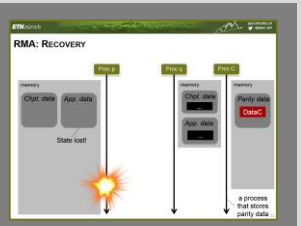
UC in RMA

MP vs. RMA

Checkpointing schemes

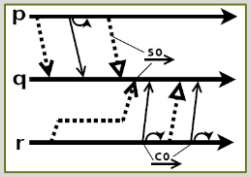
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Holistic fault-tolerance library

Design

Checkpoints on demand

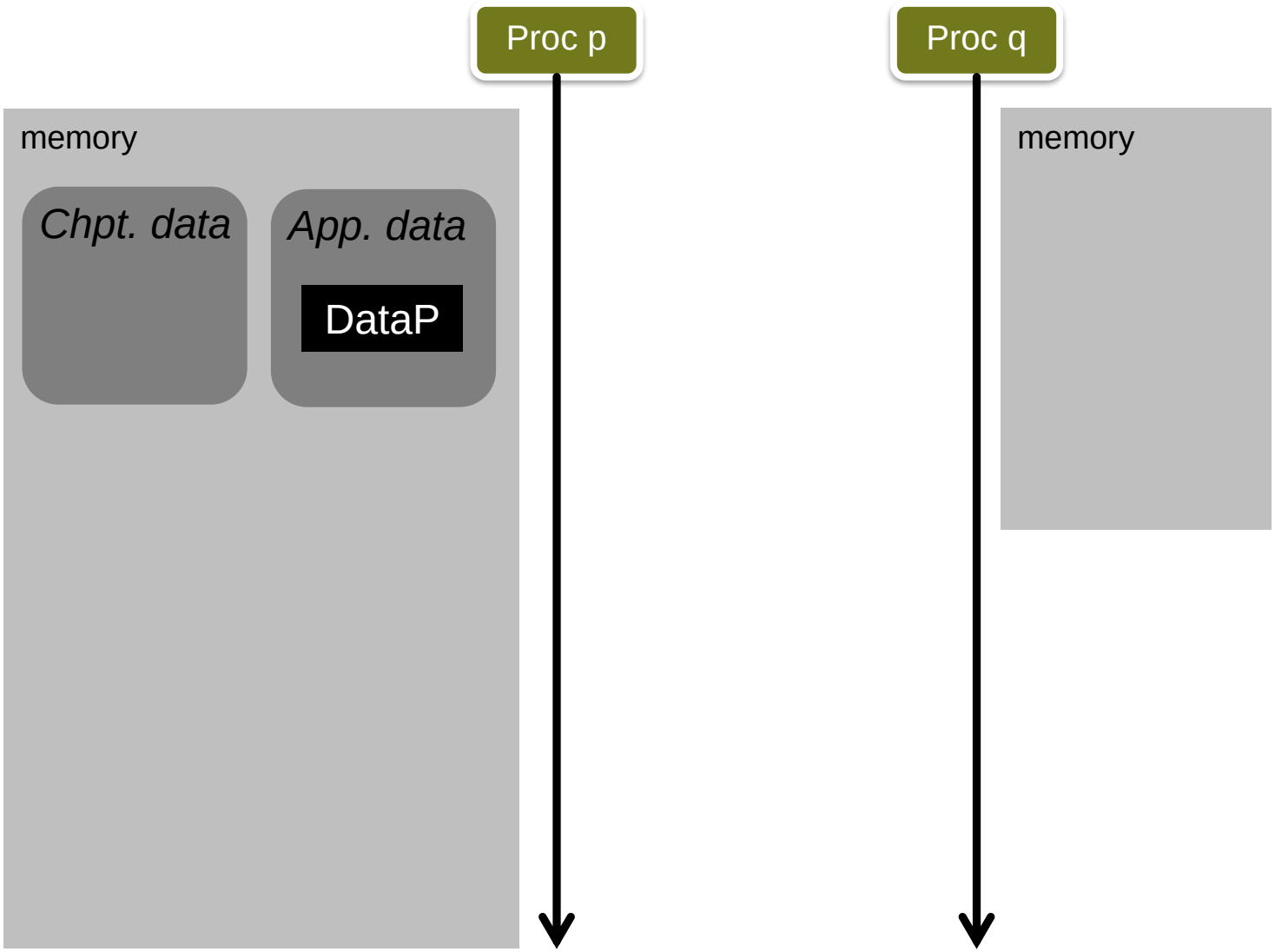
Optimizations

Performance

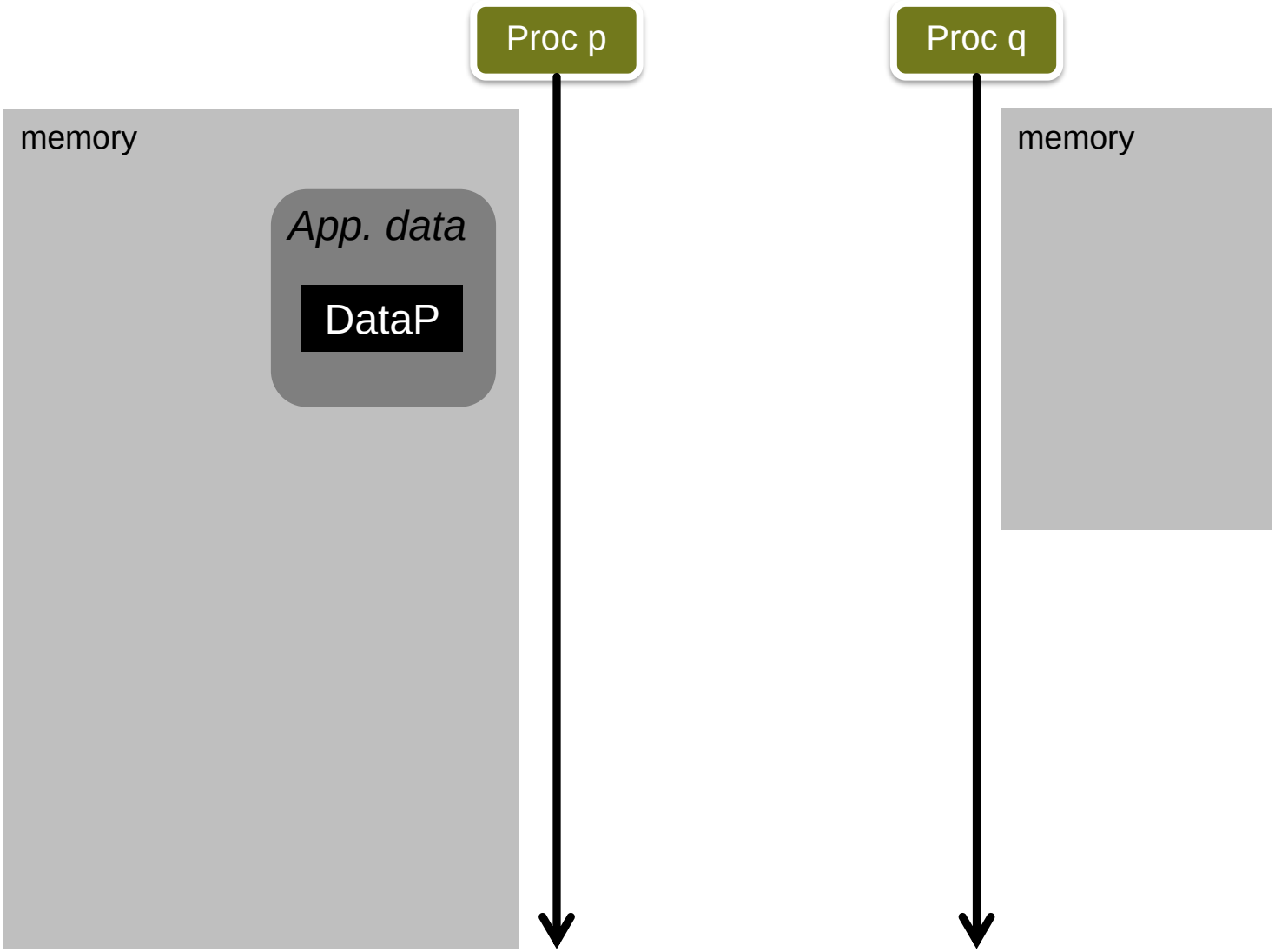
Distribution of processes

Decreasing failure prob.

RMA: RECOVERY

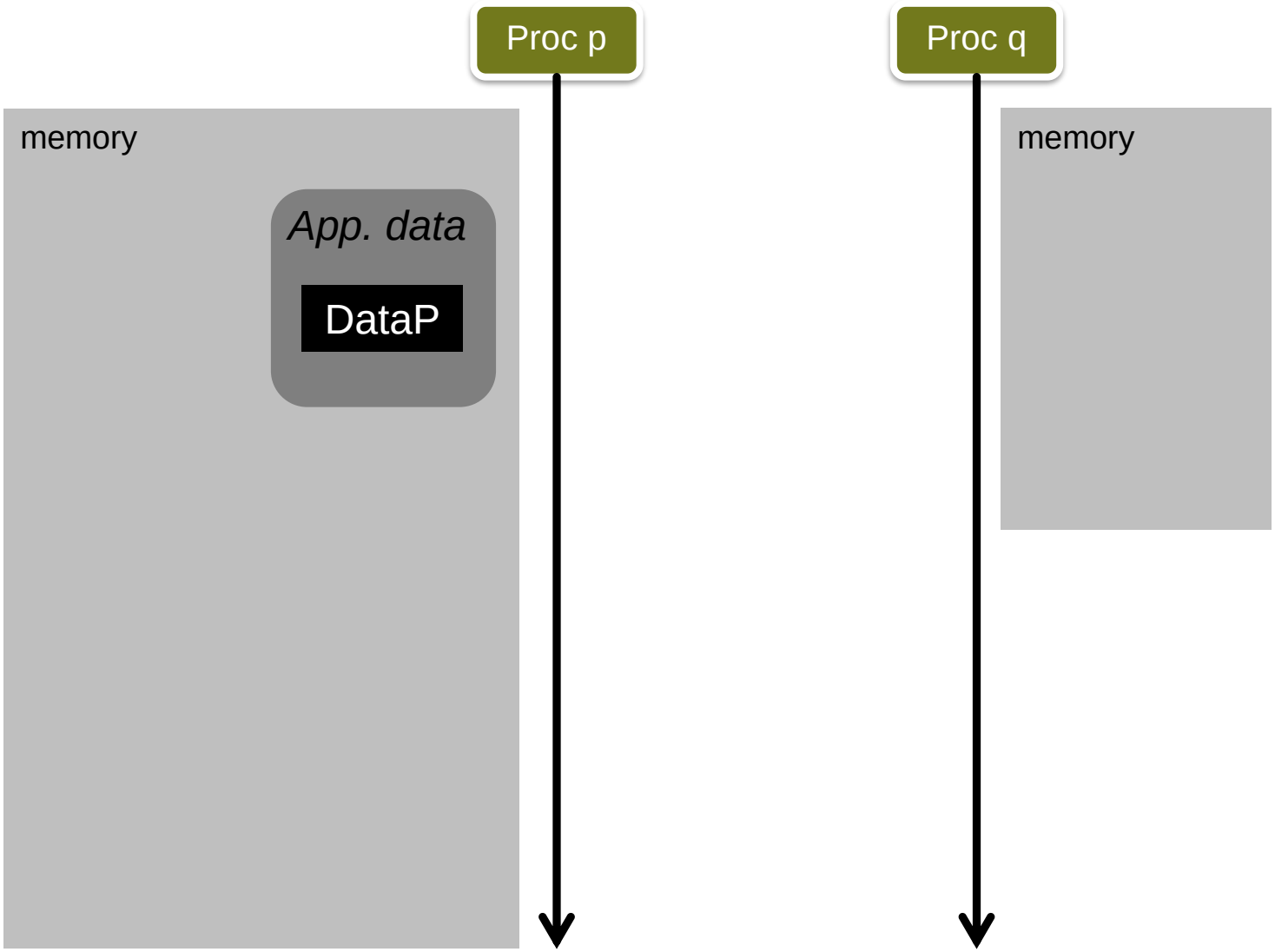


RMA: RECOVERY



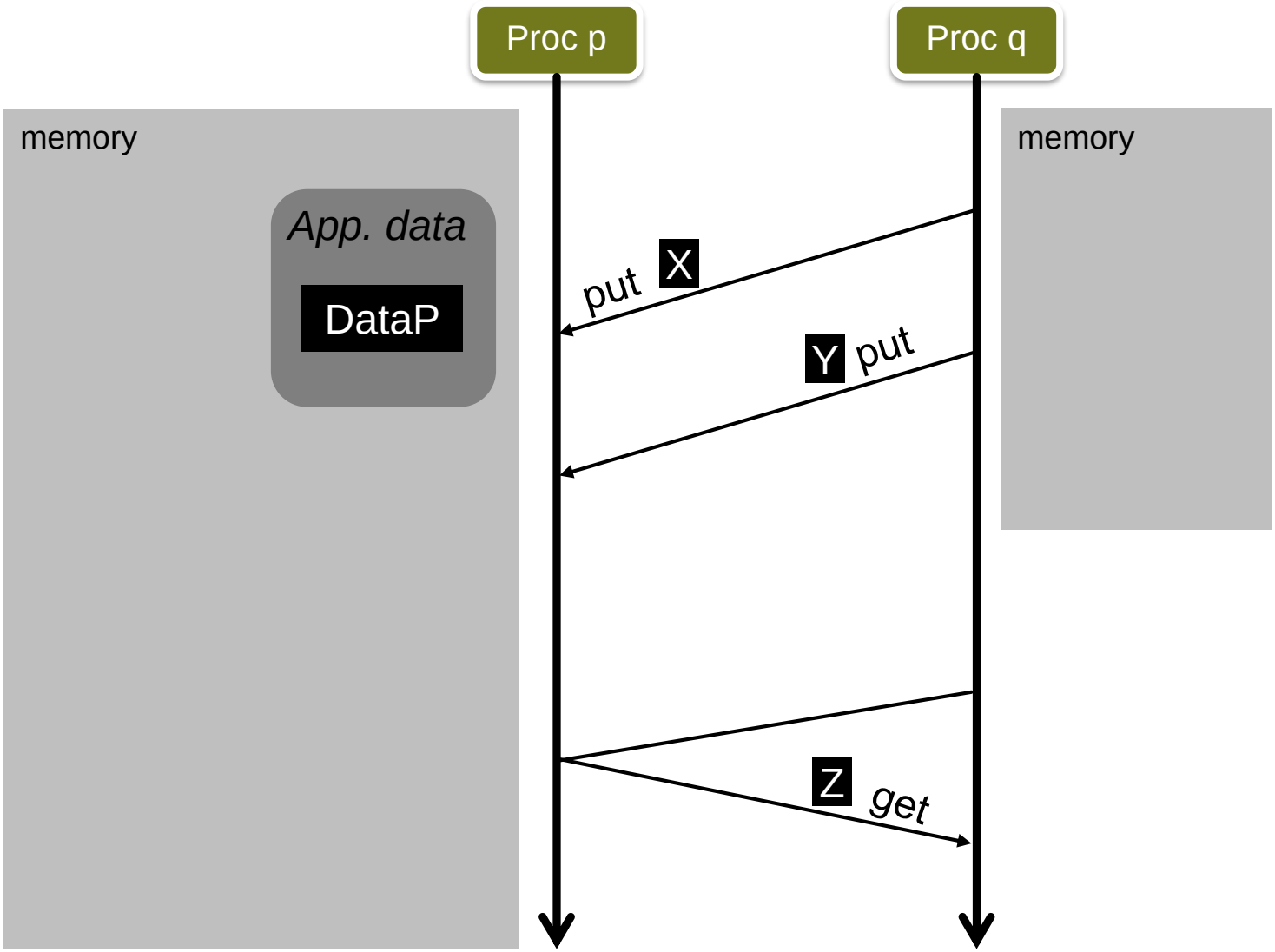
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



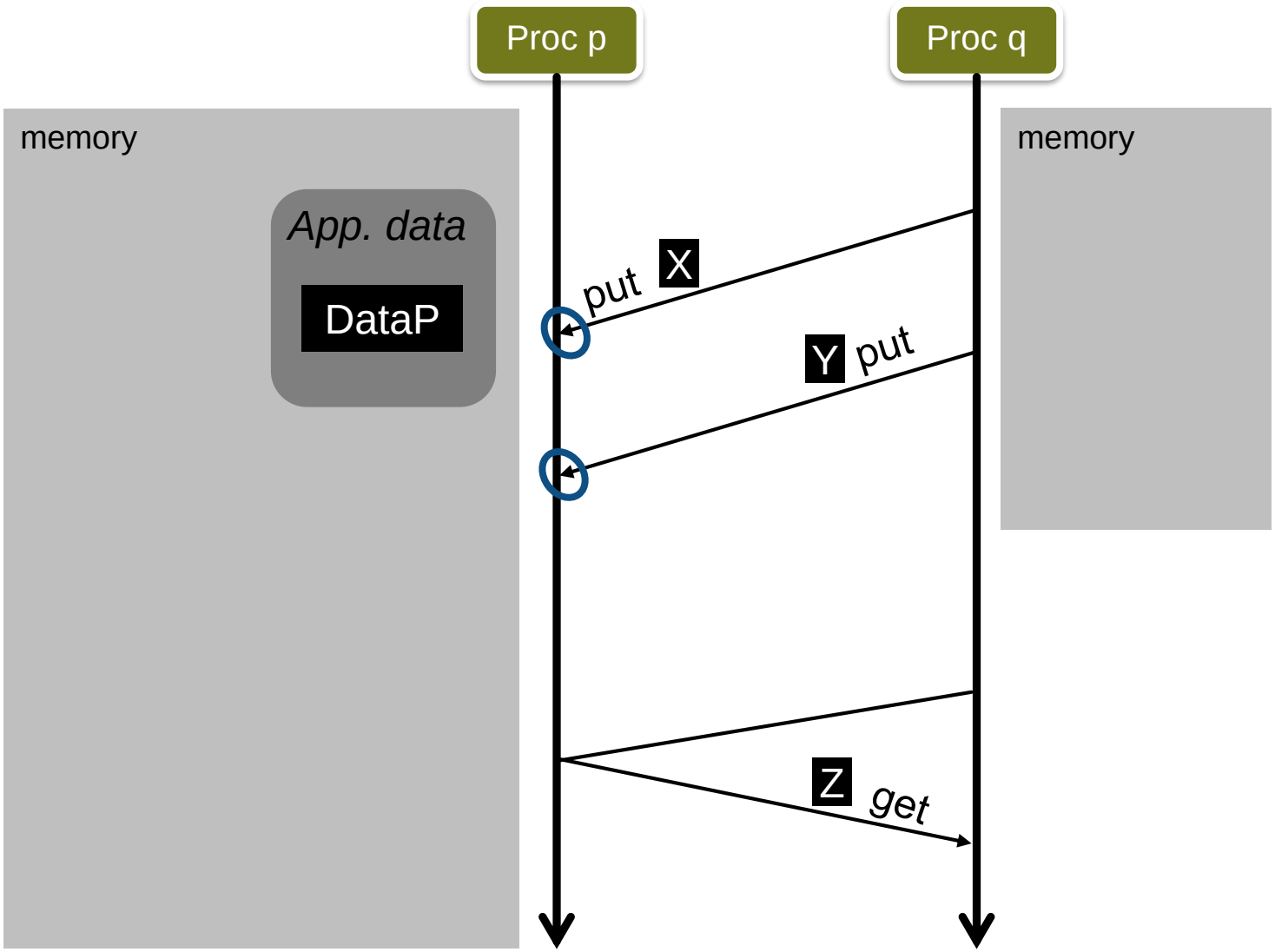
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



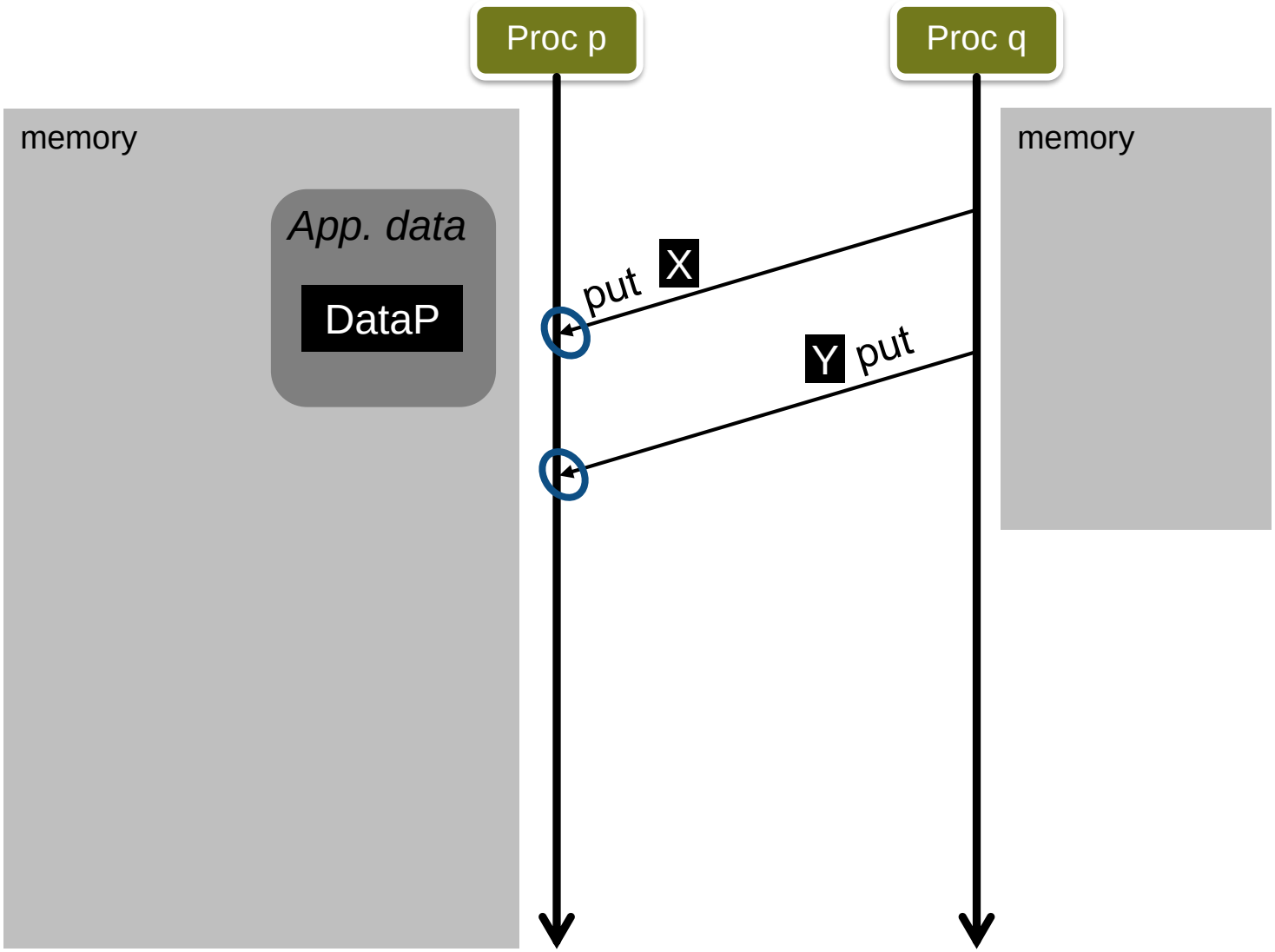
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



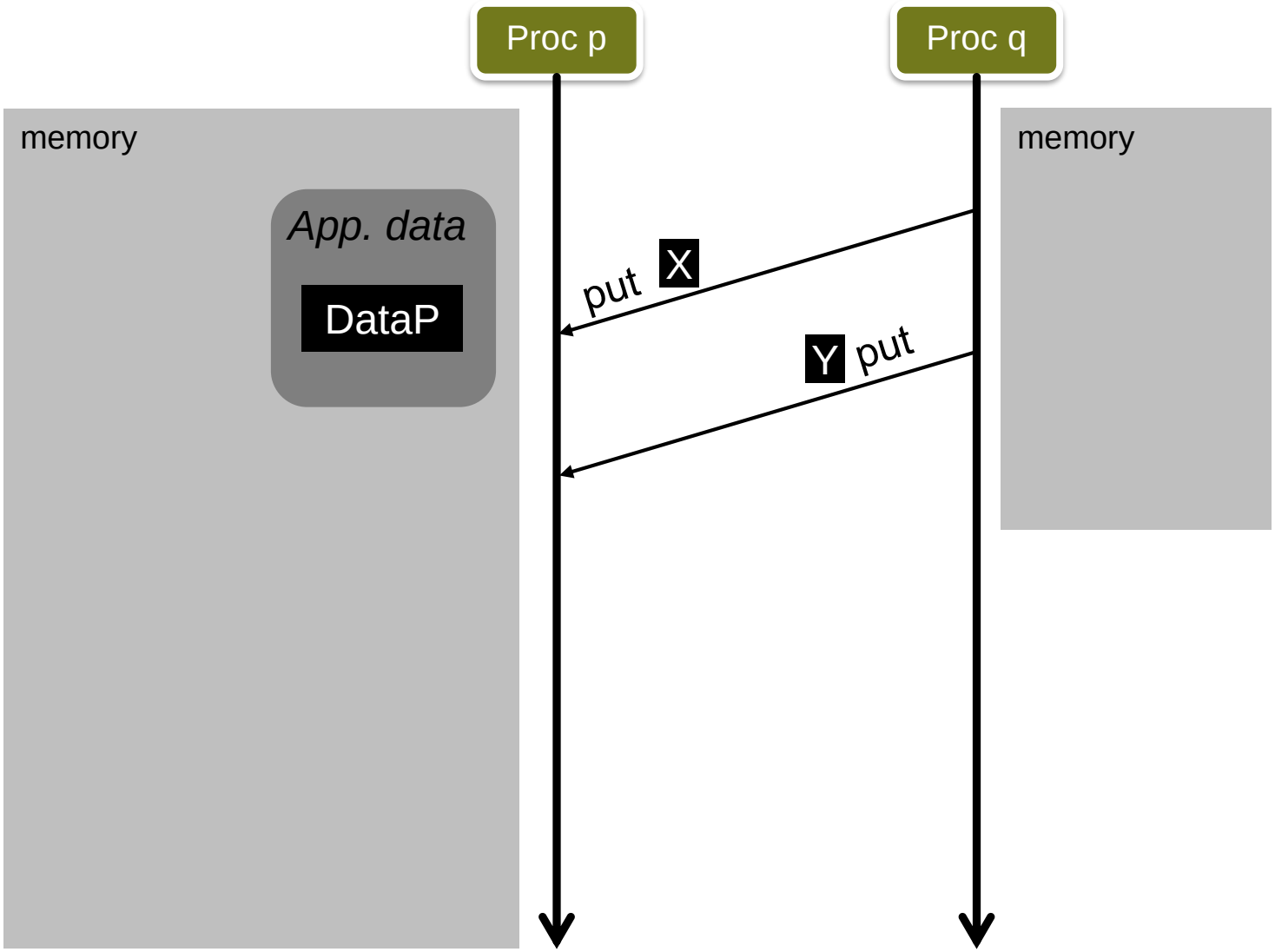
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



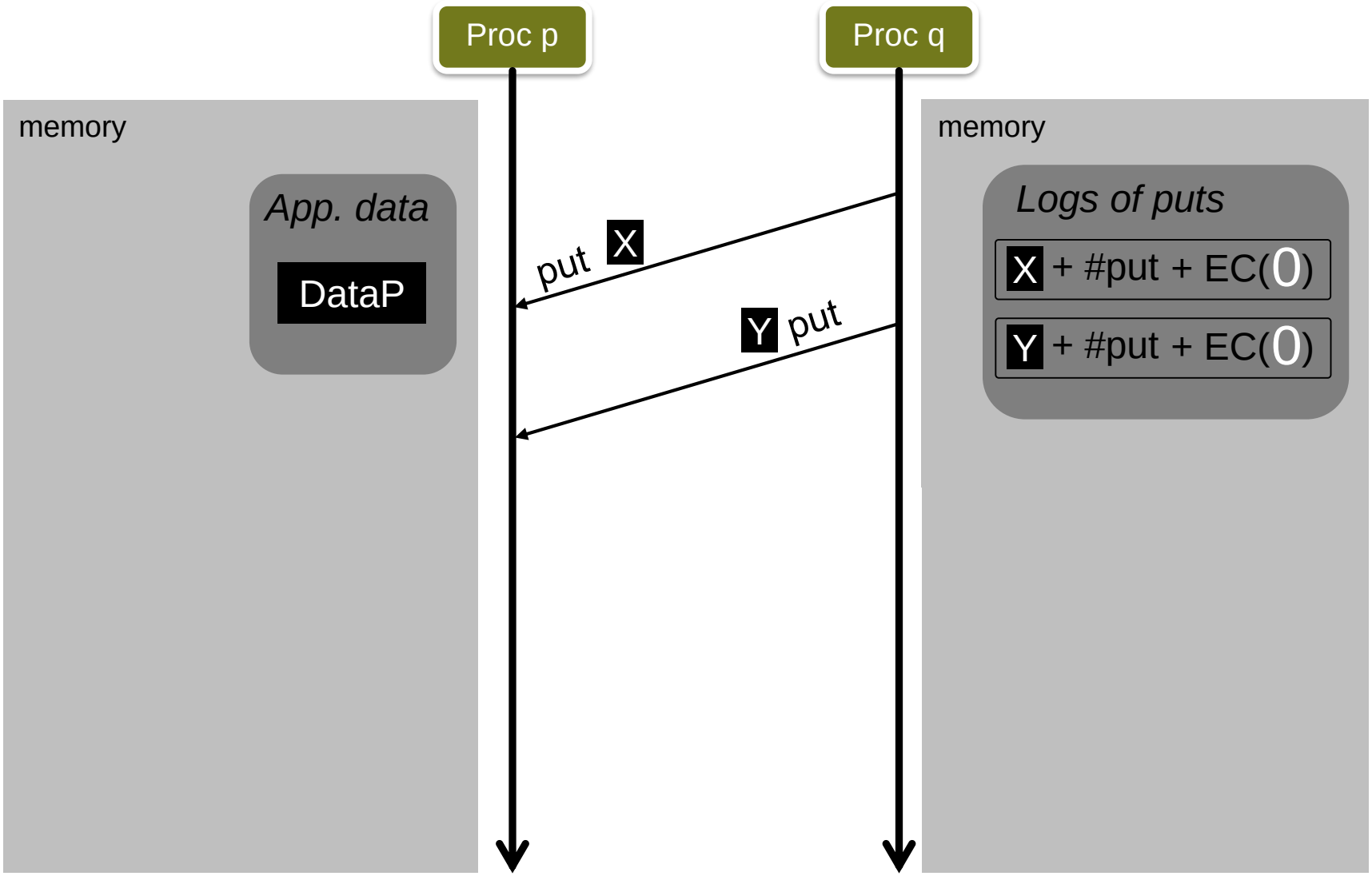
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



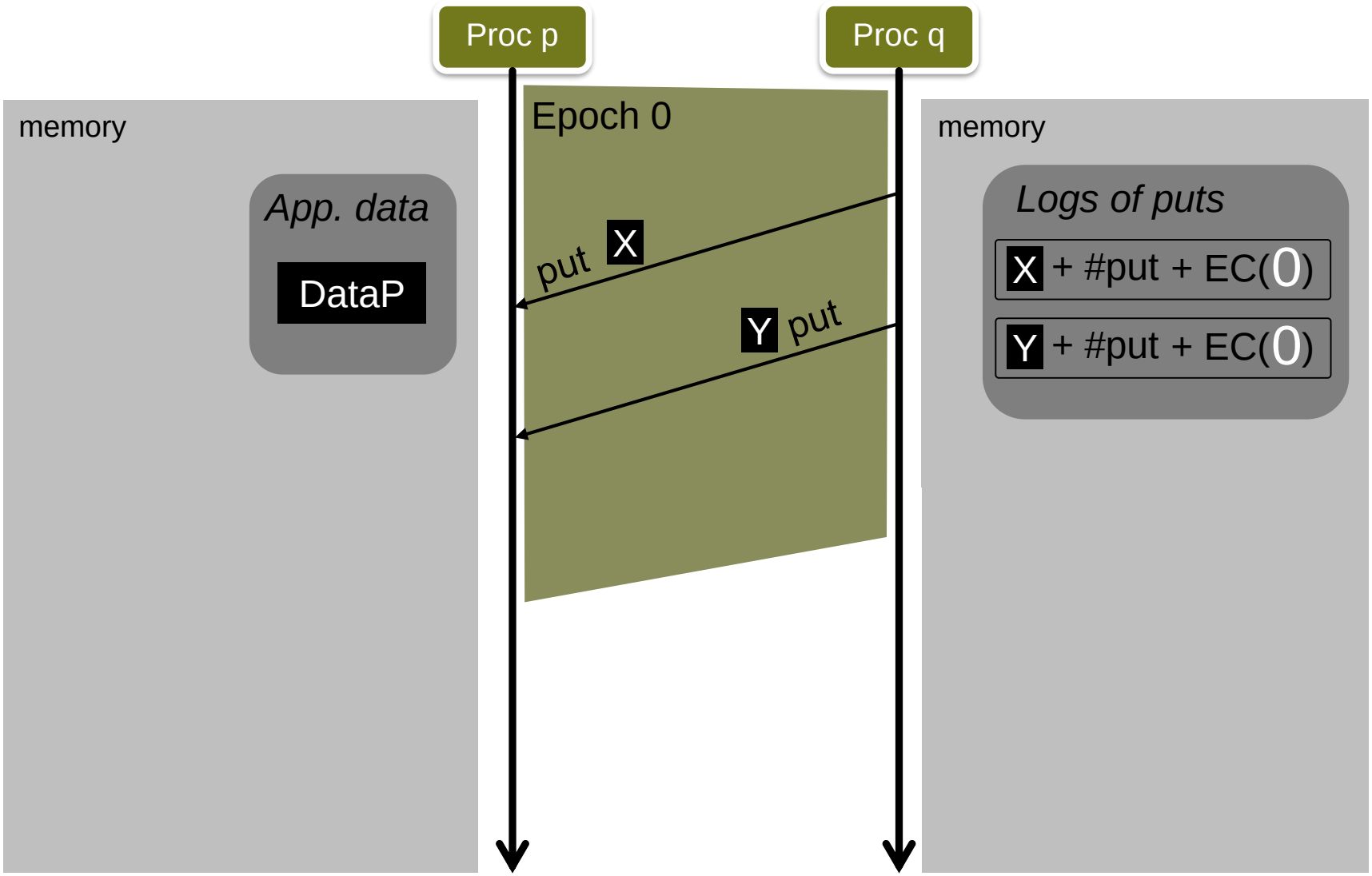
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



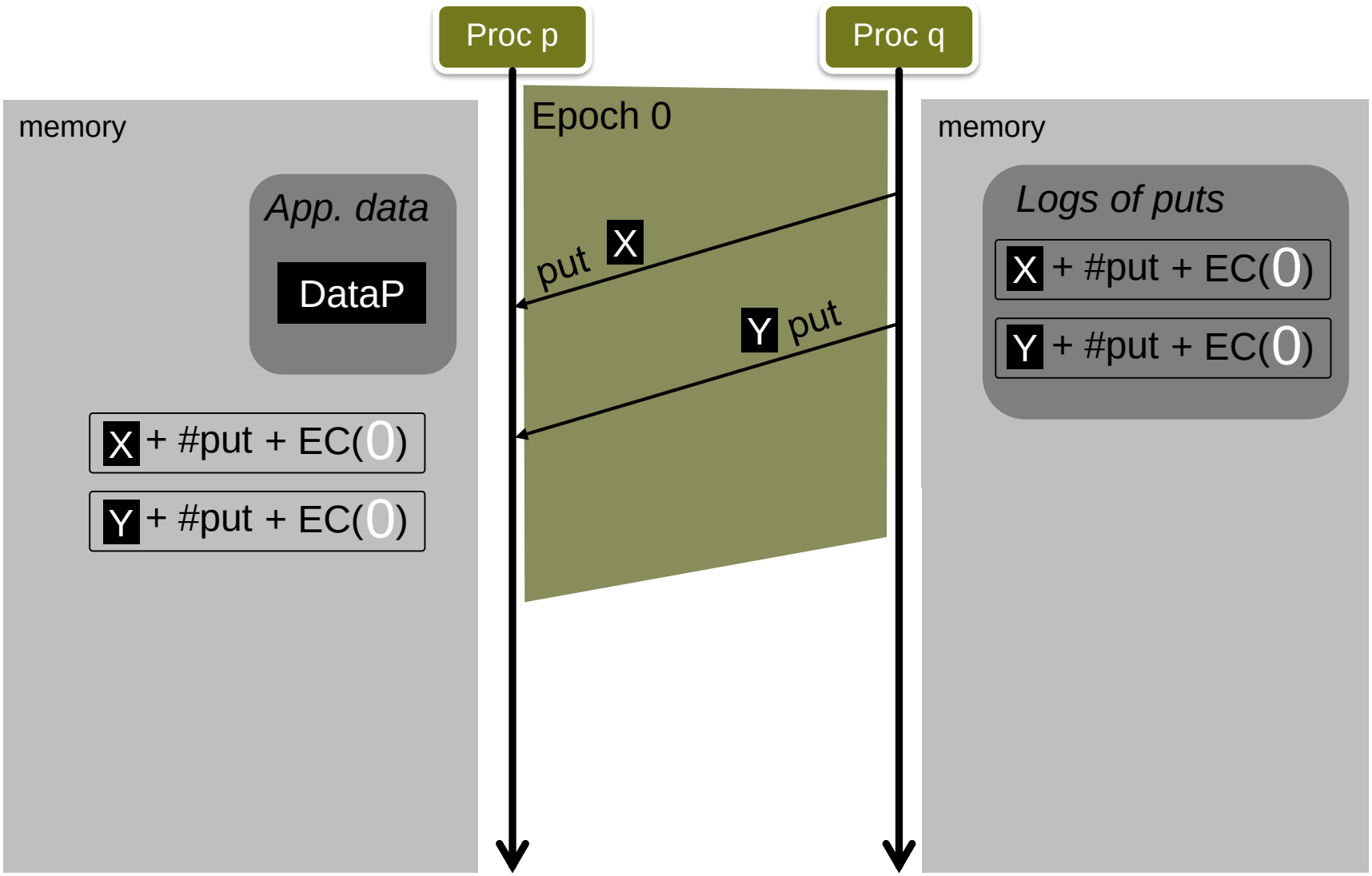
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



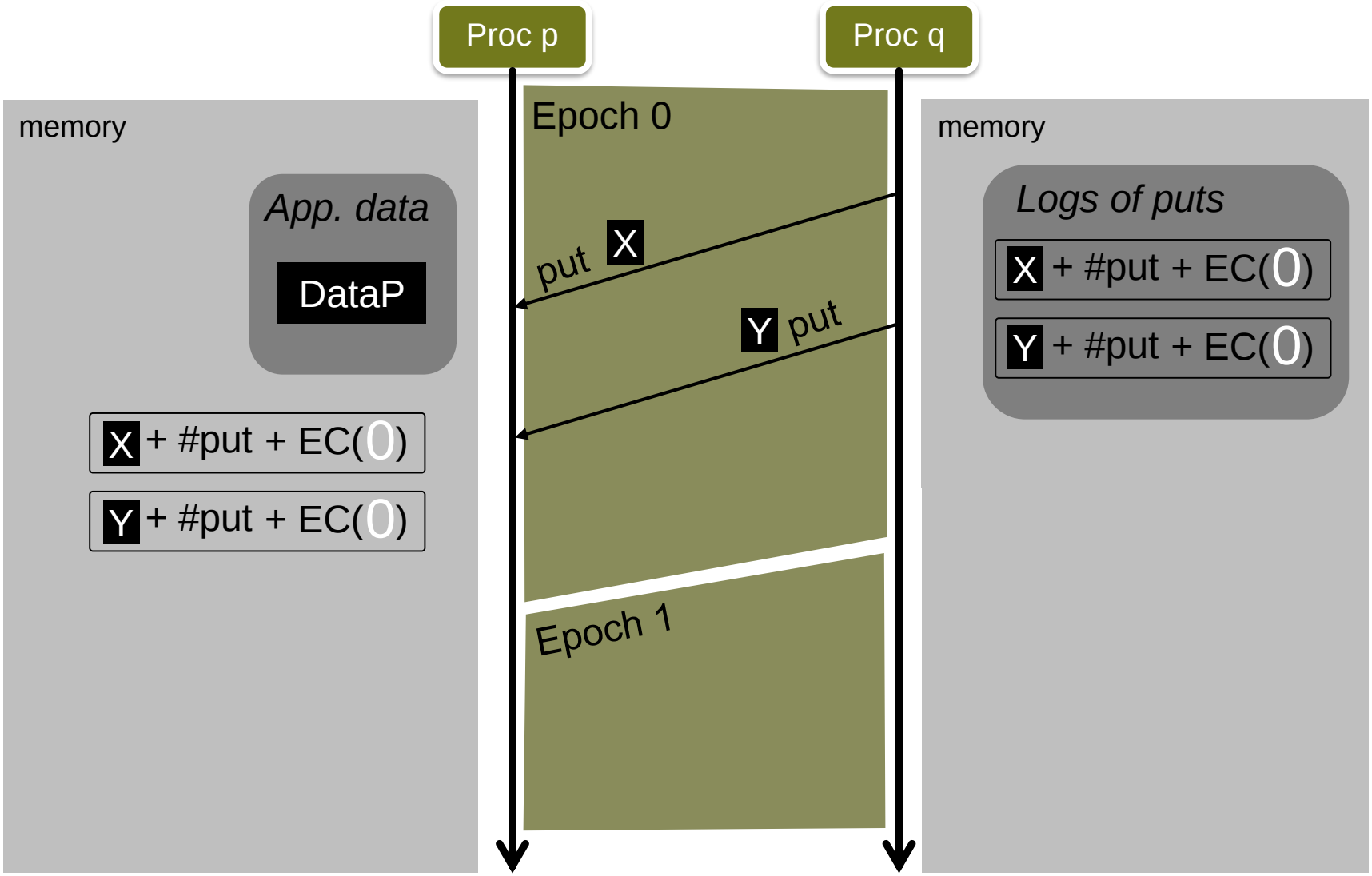
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



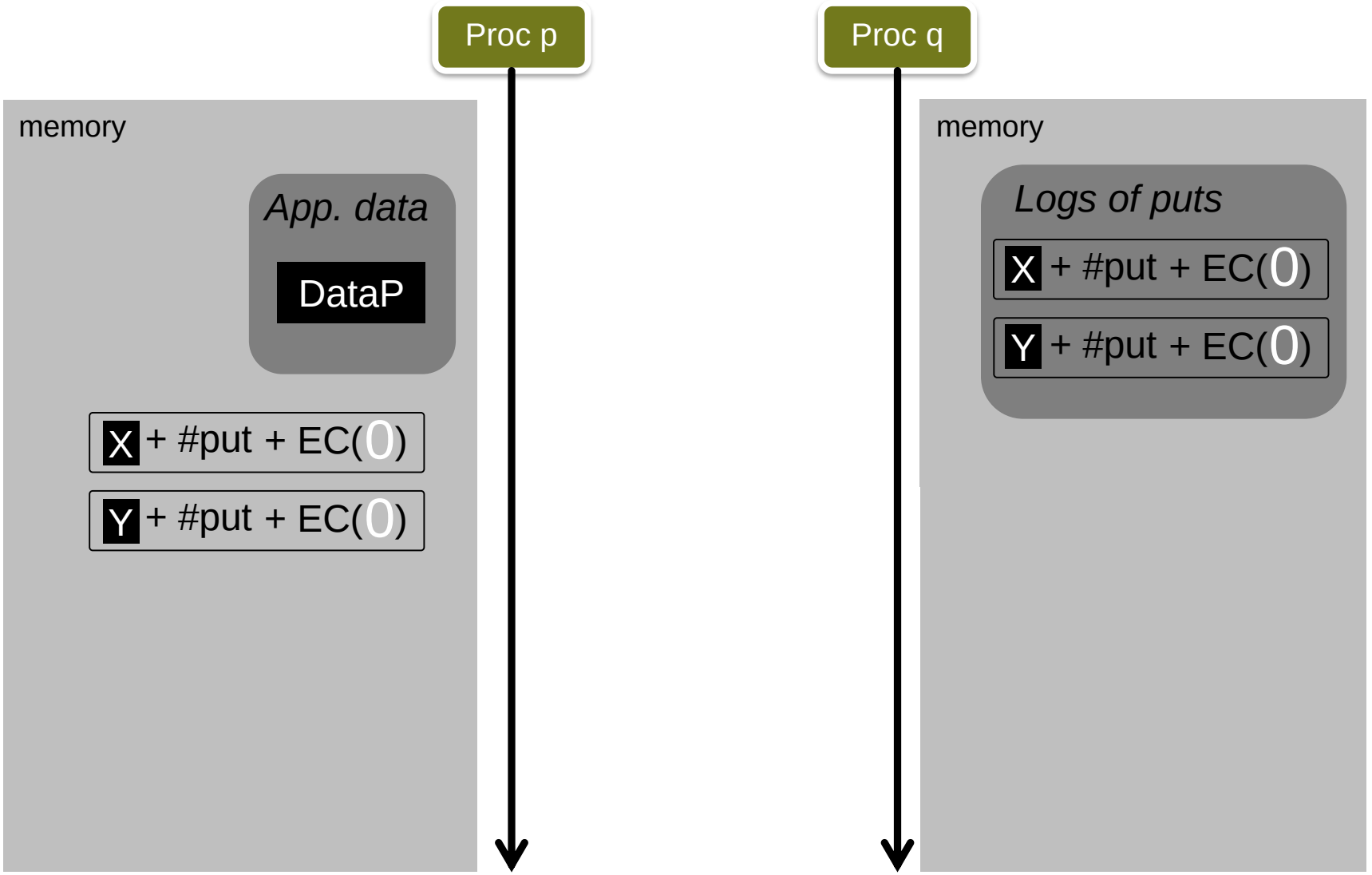
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



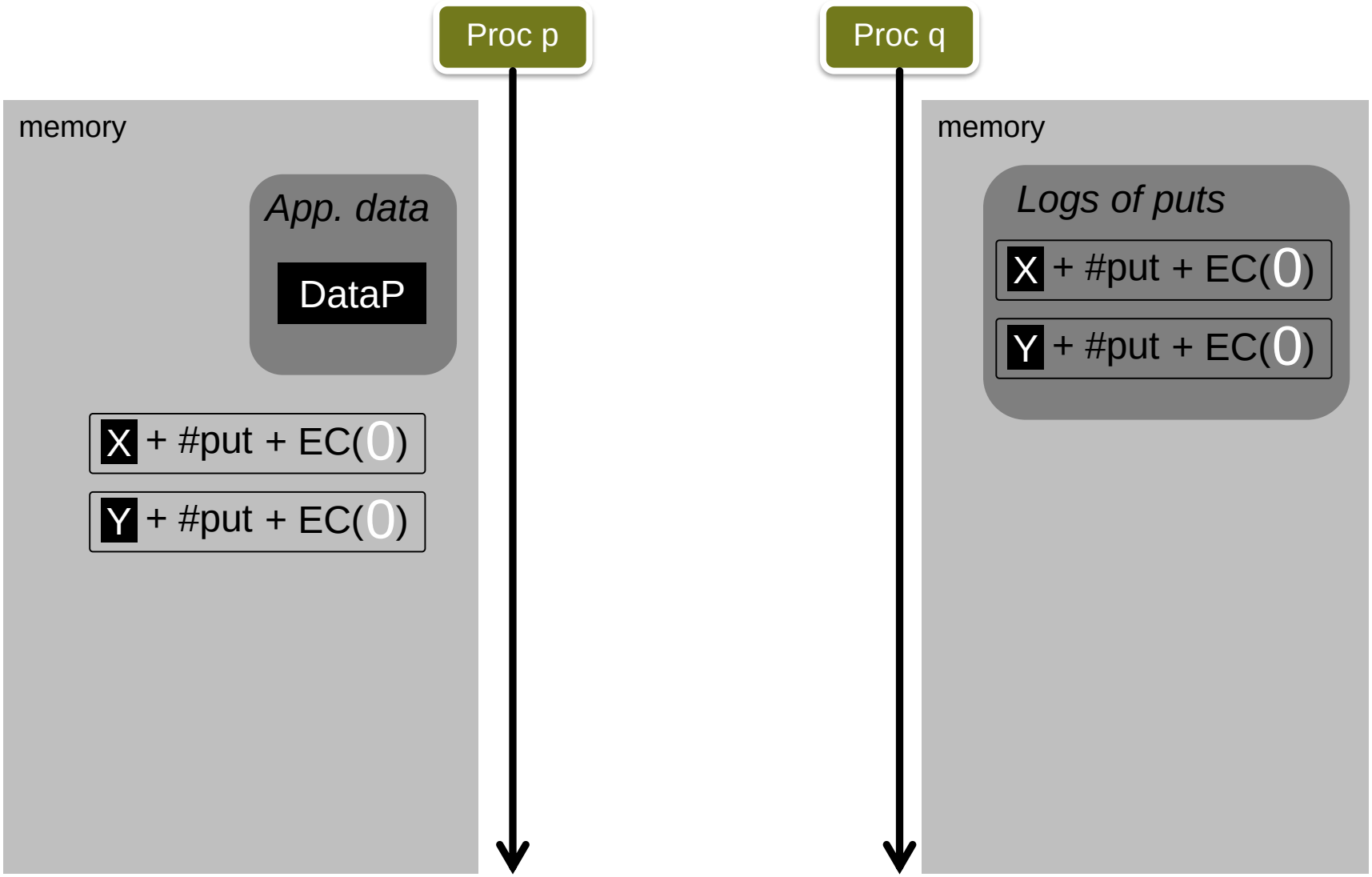
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



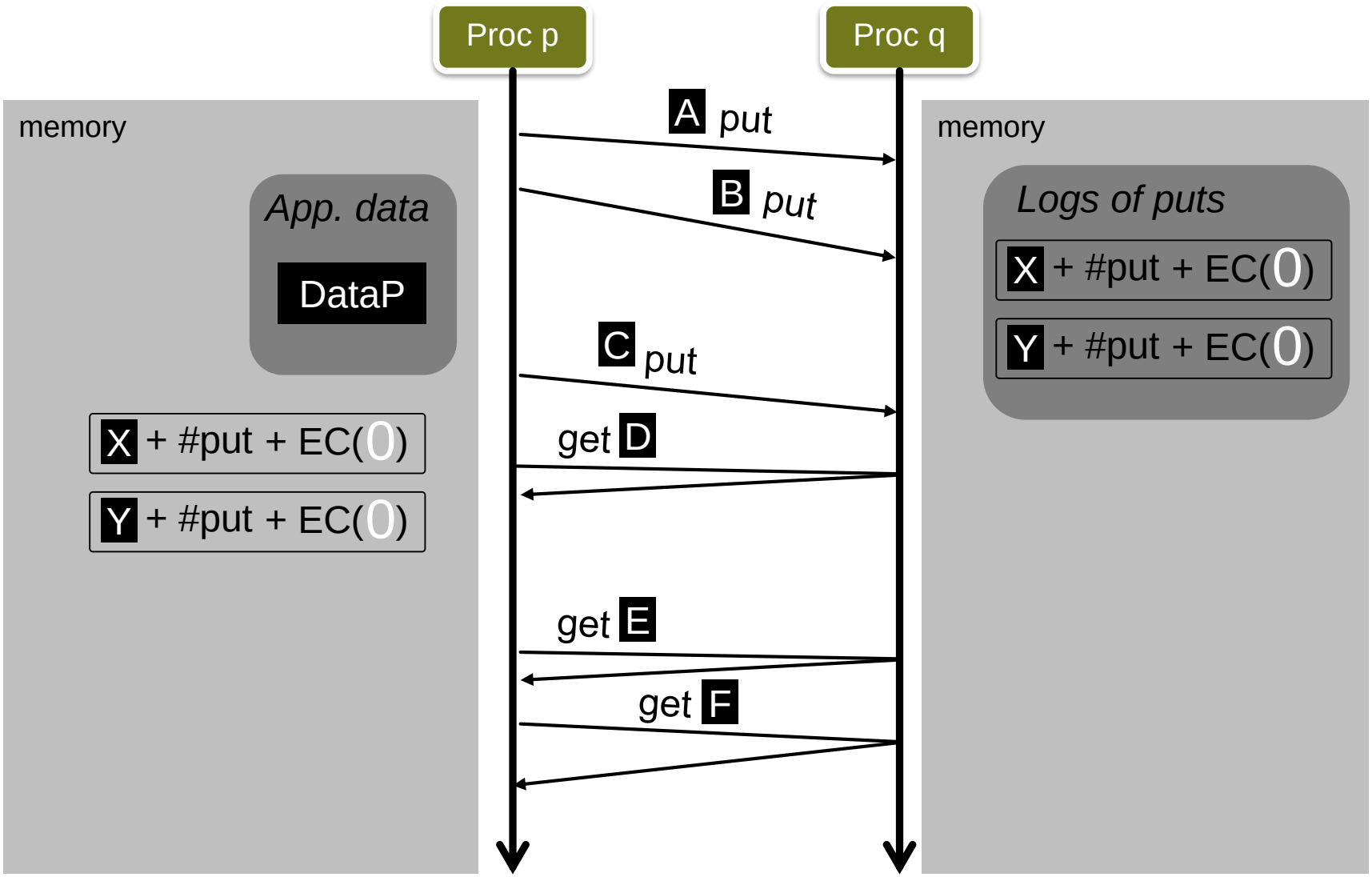
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



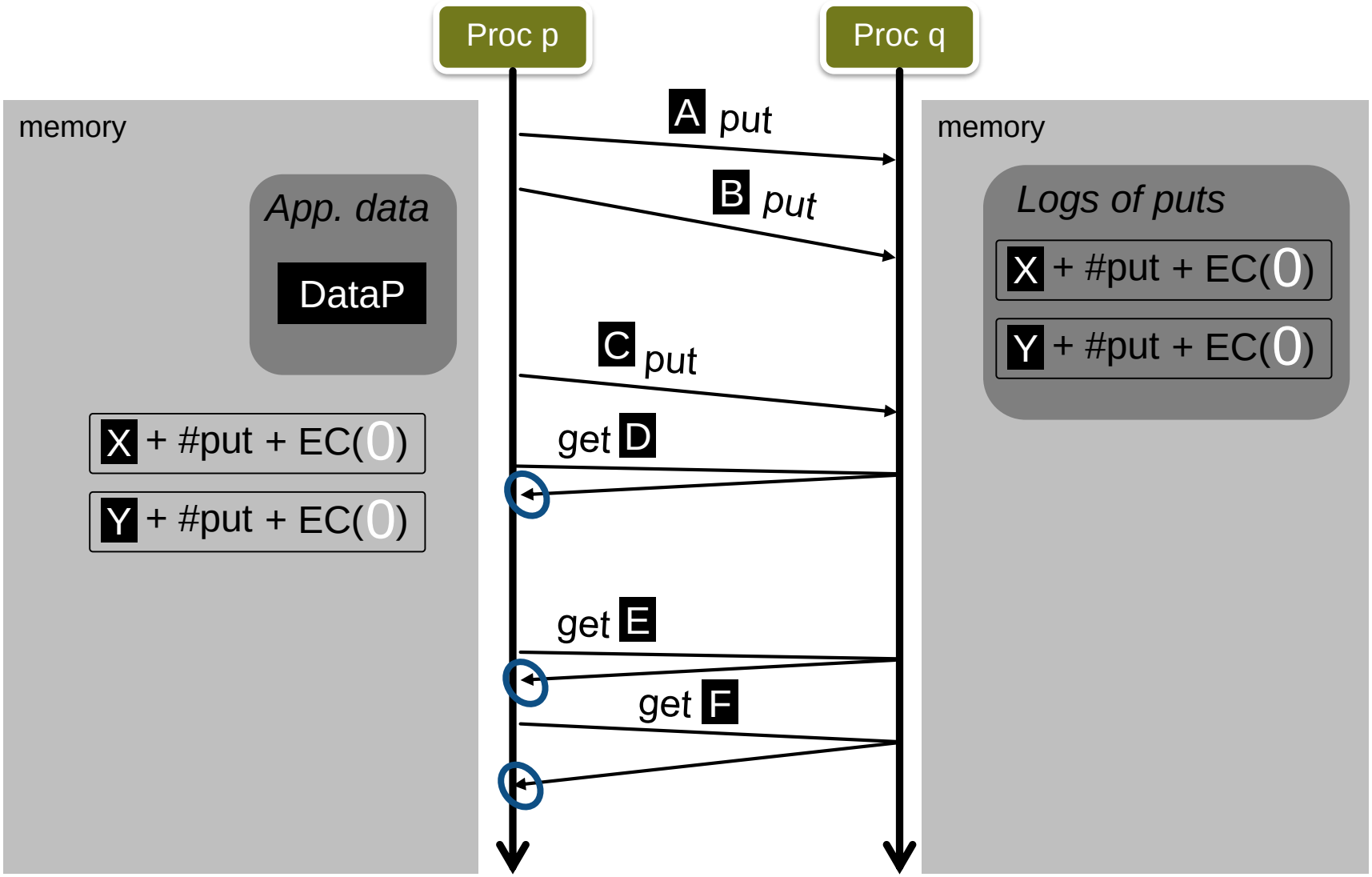
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



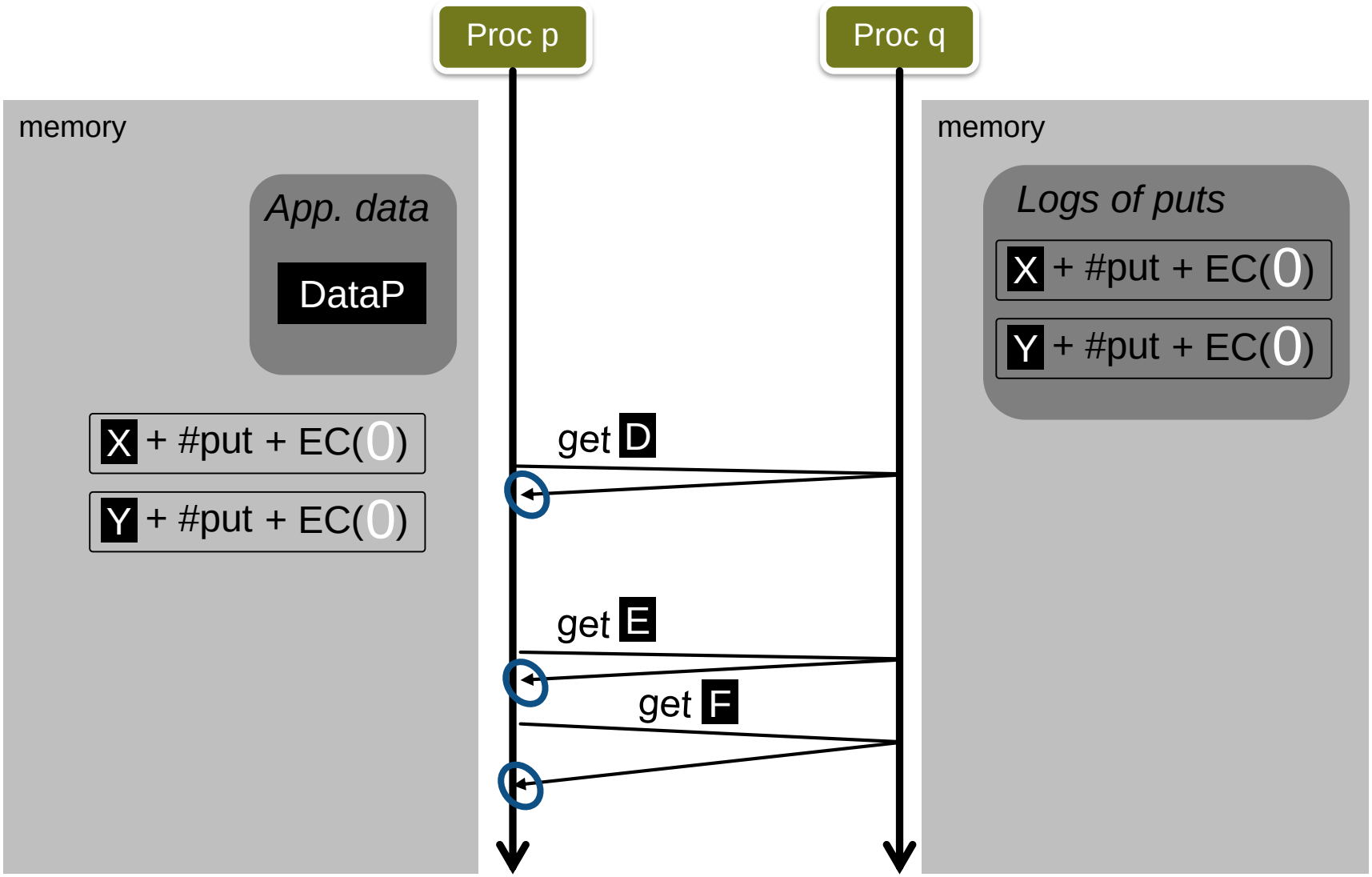
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



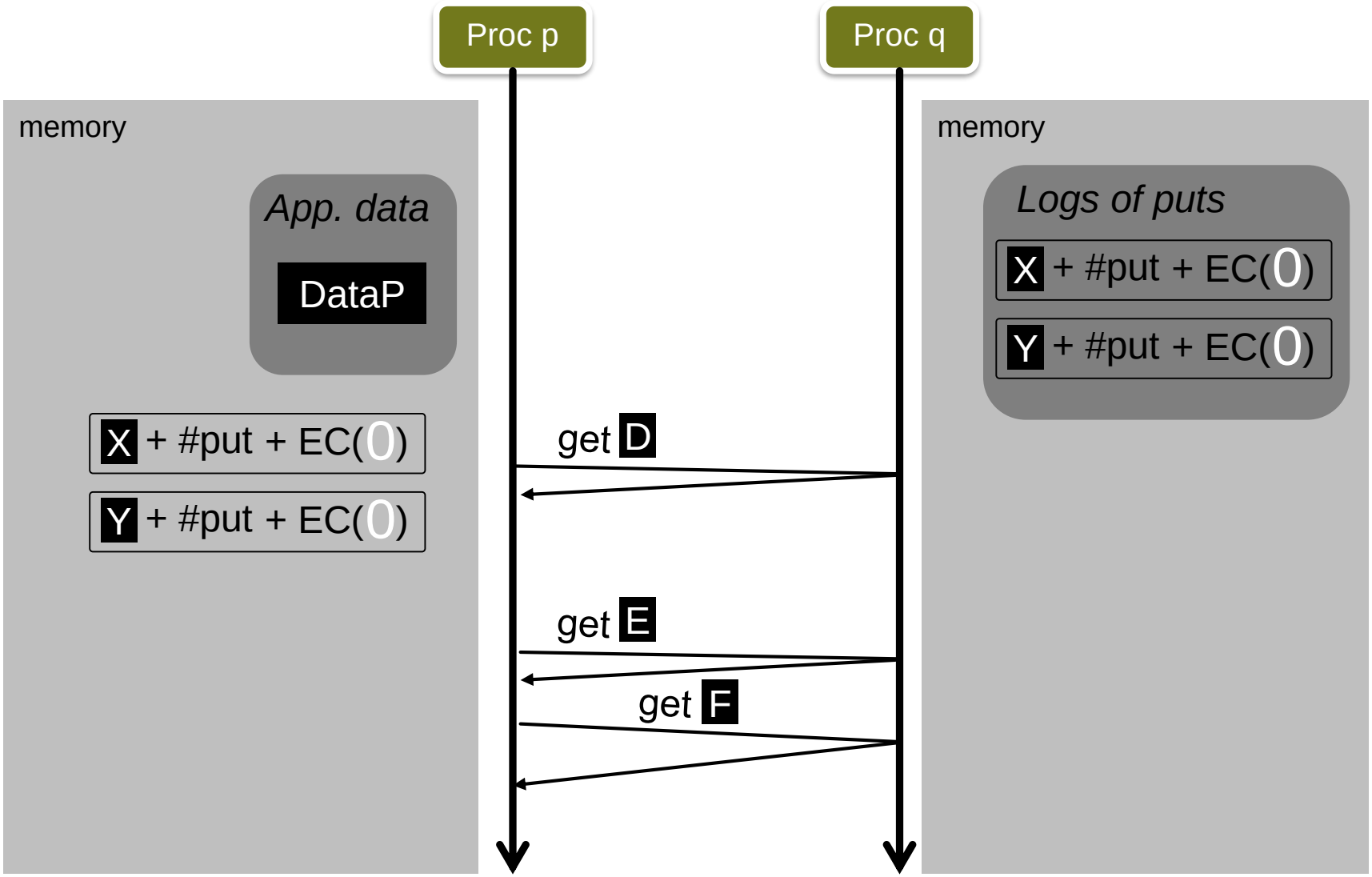
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



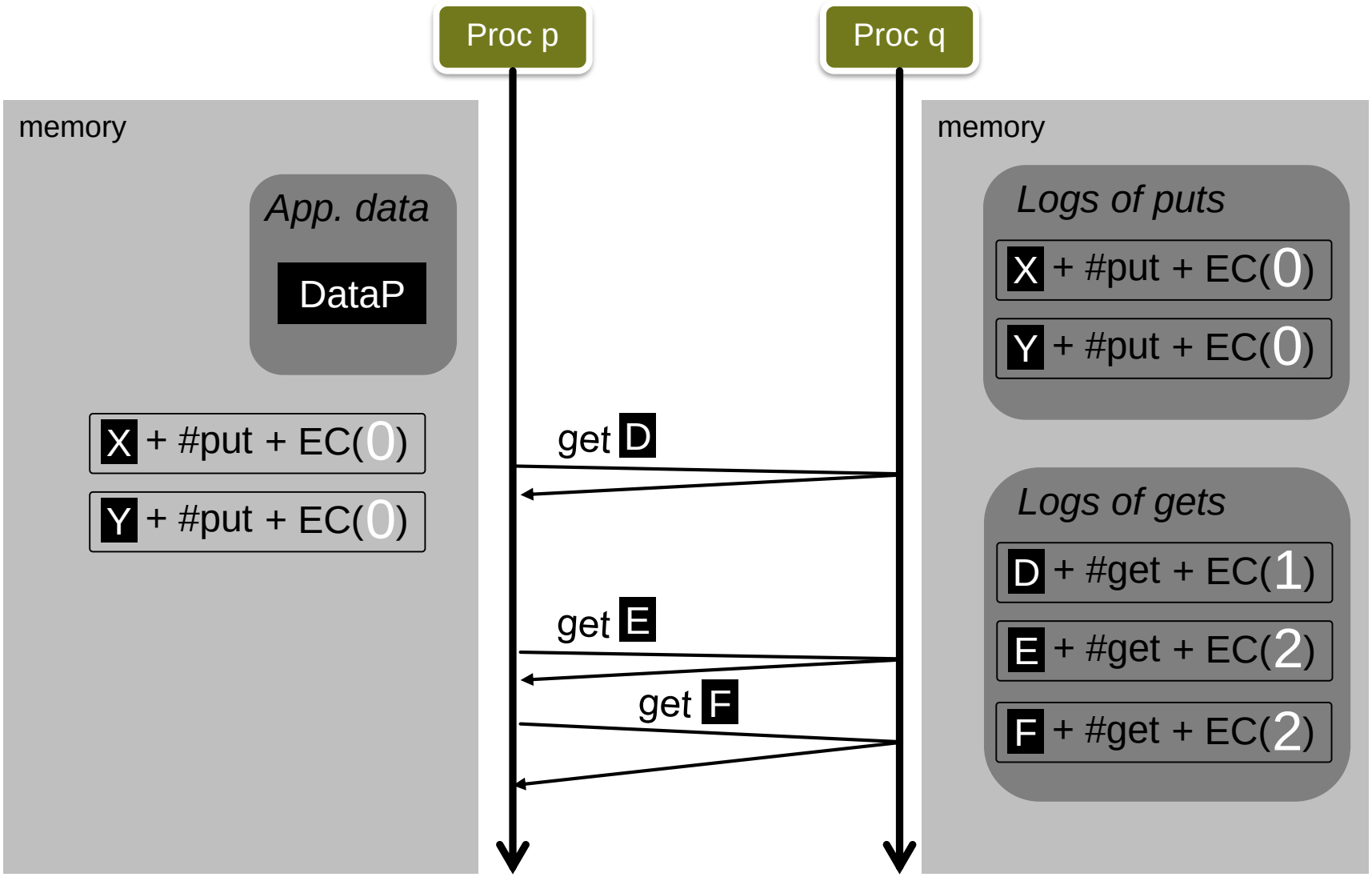
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



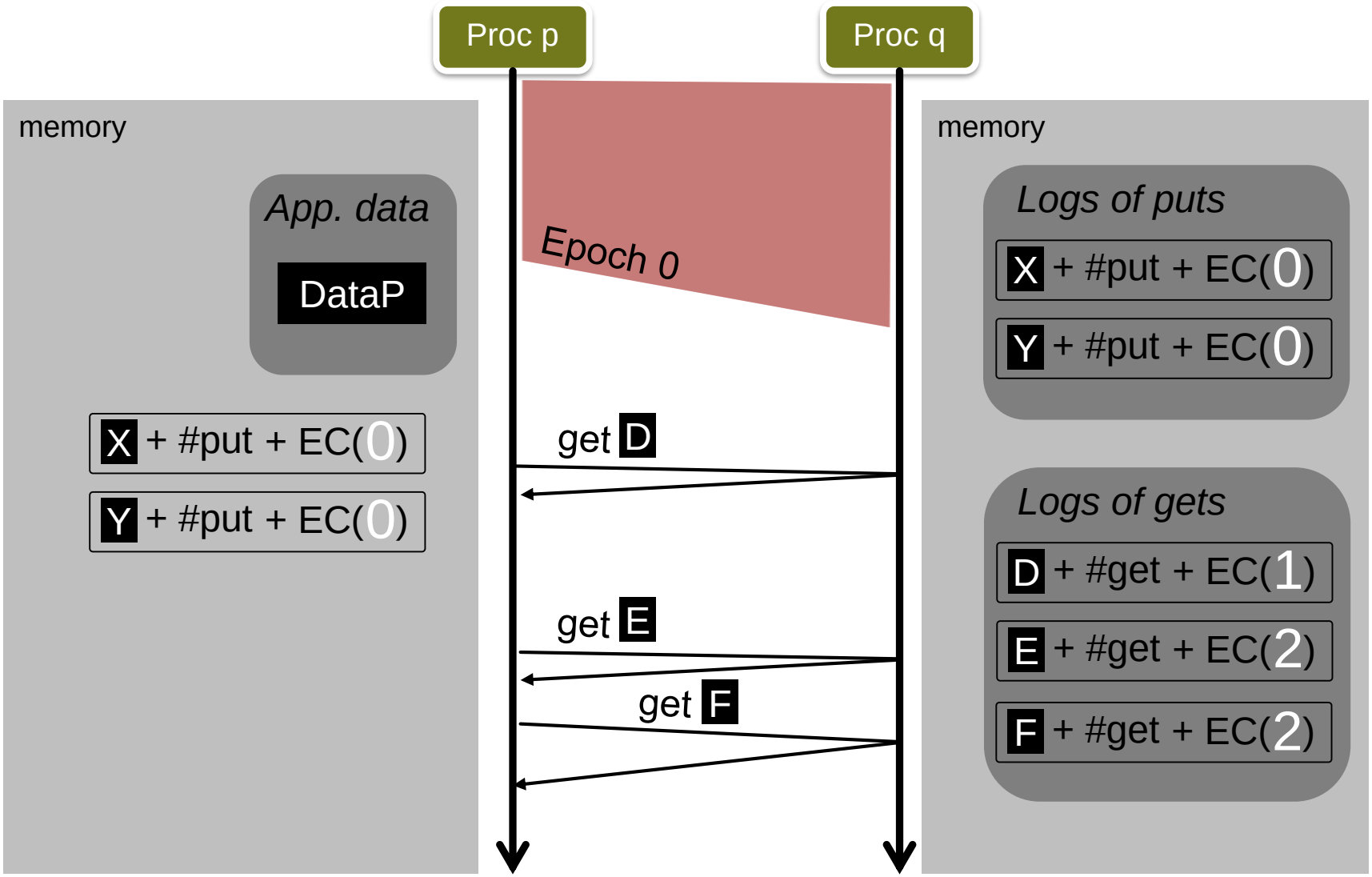
RMA: Recovery

Stage 2: replay actions beyond the checkpoint



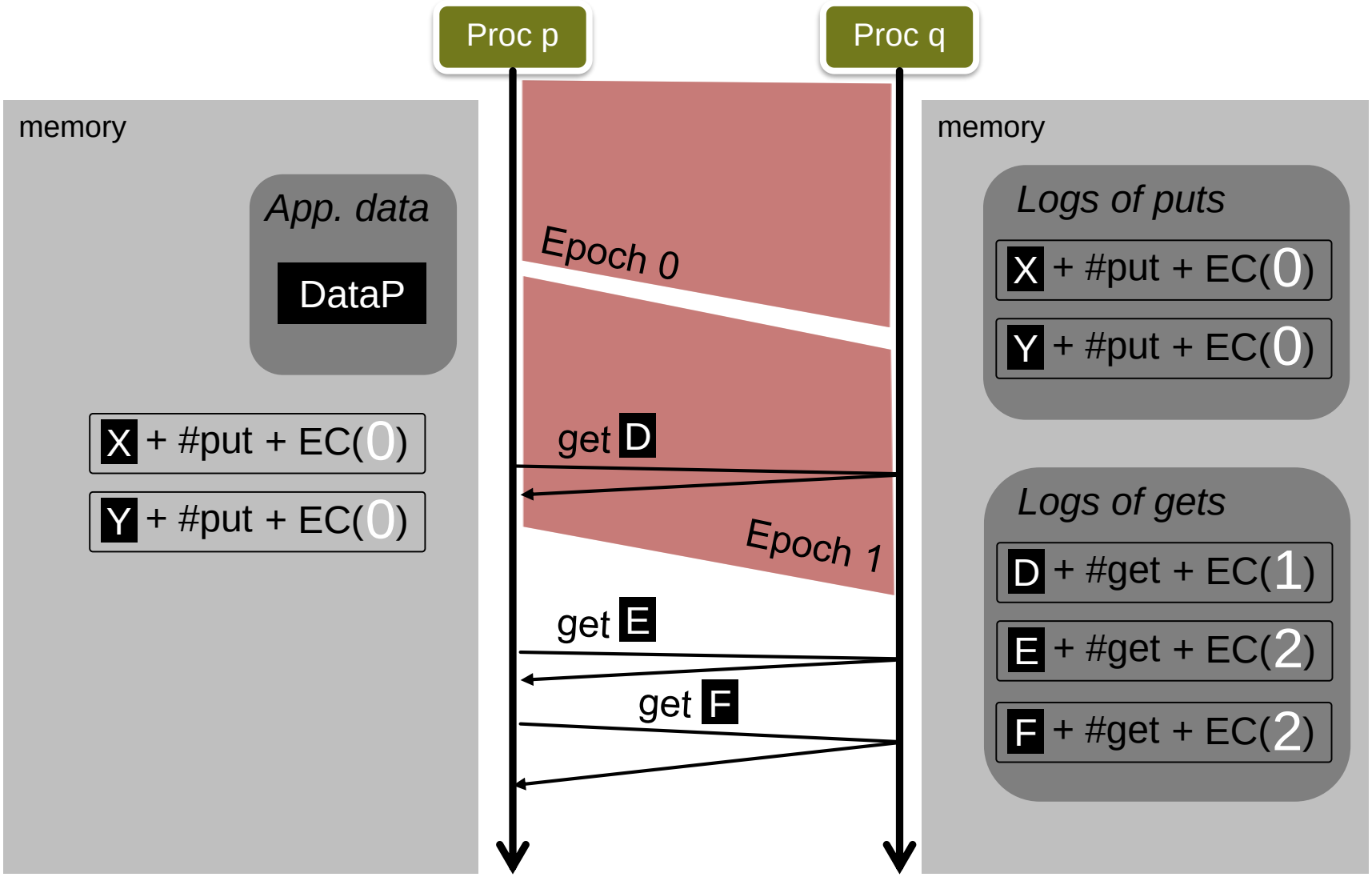
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



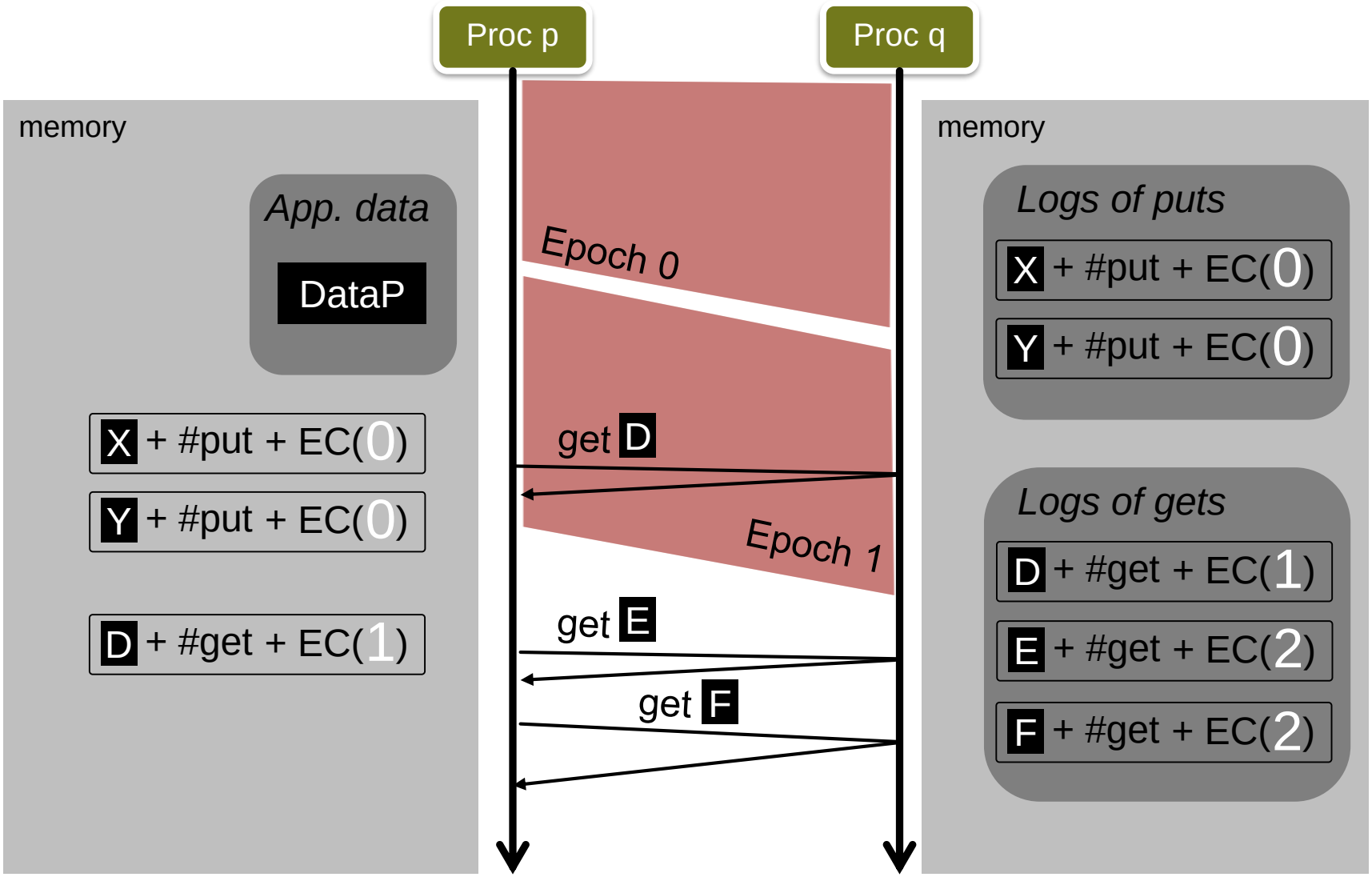
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



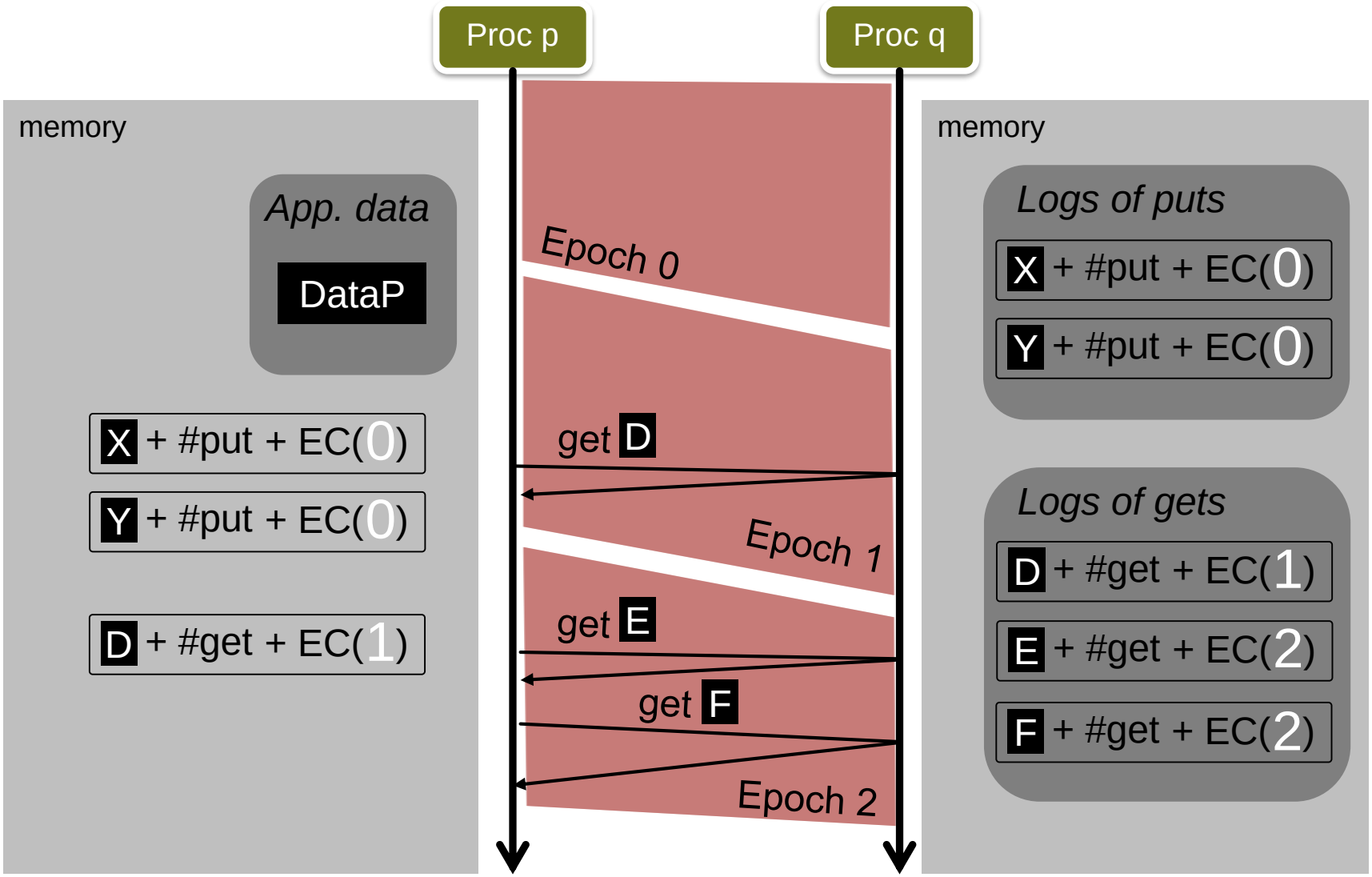
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



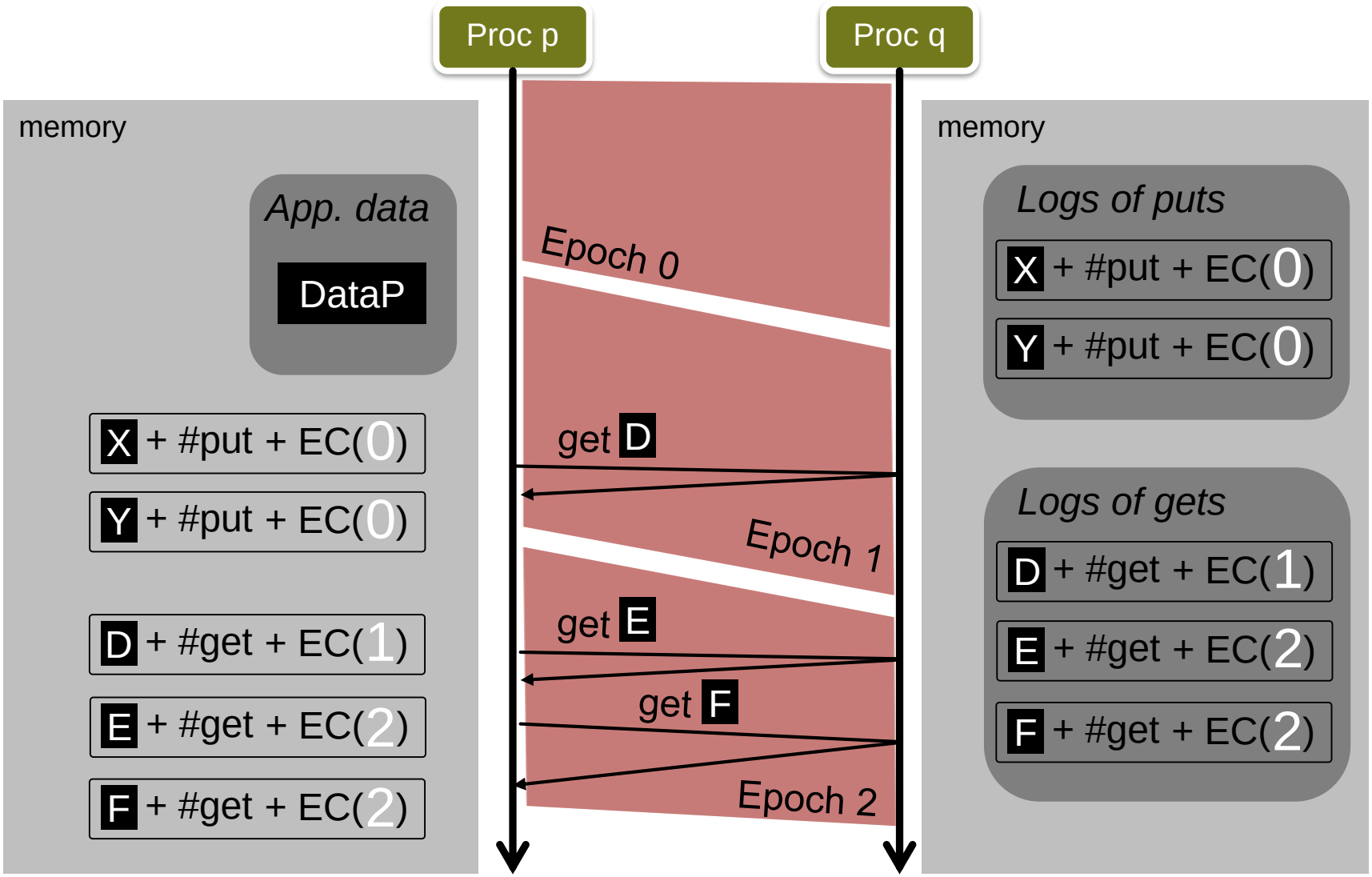
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



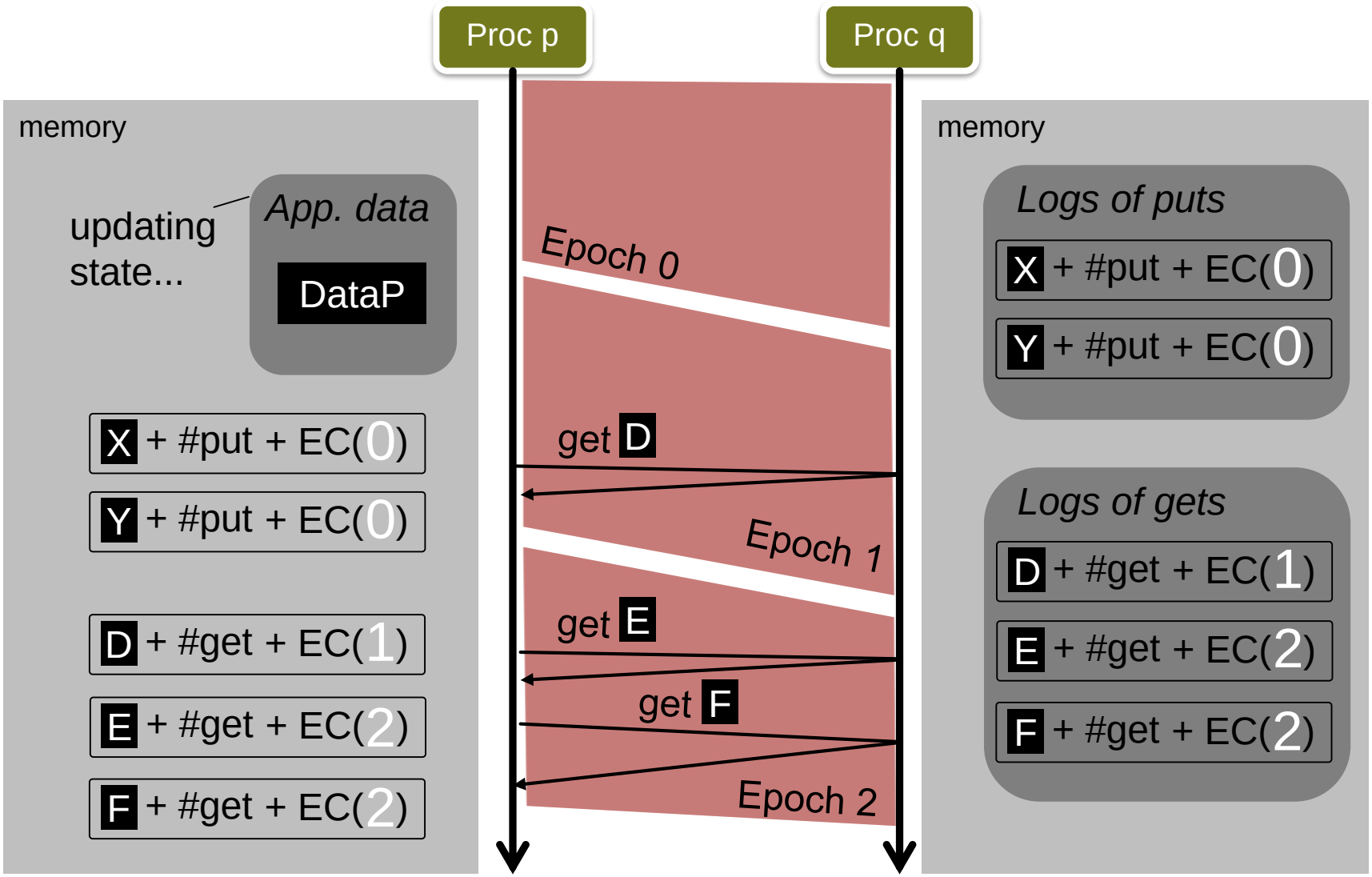
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



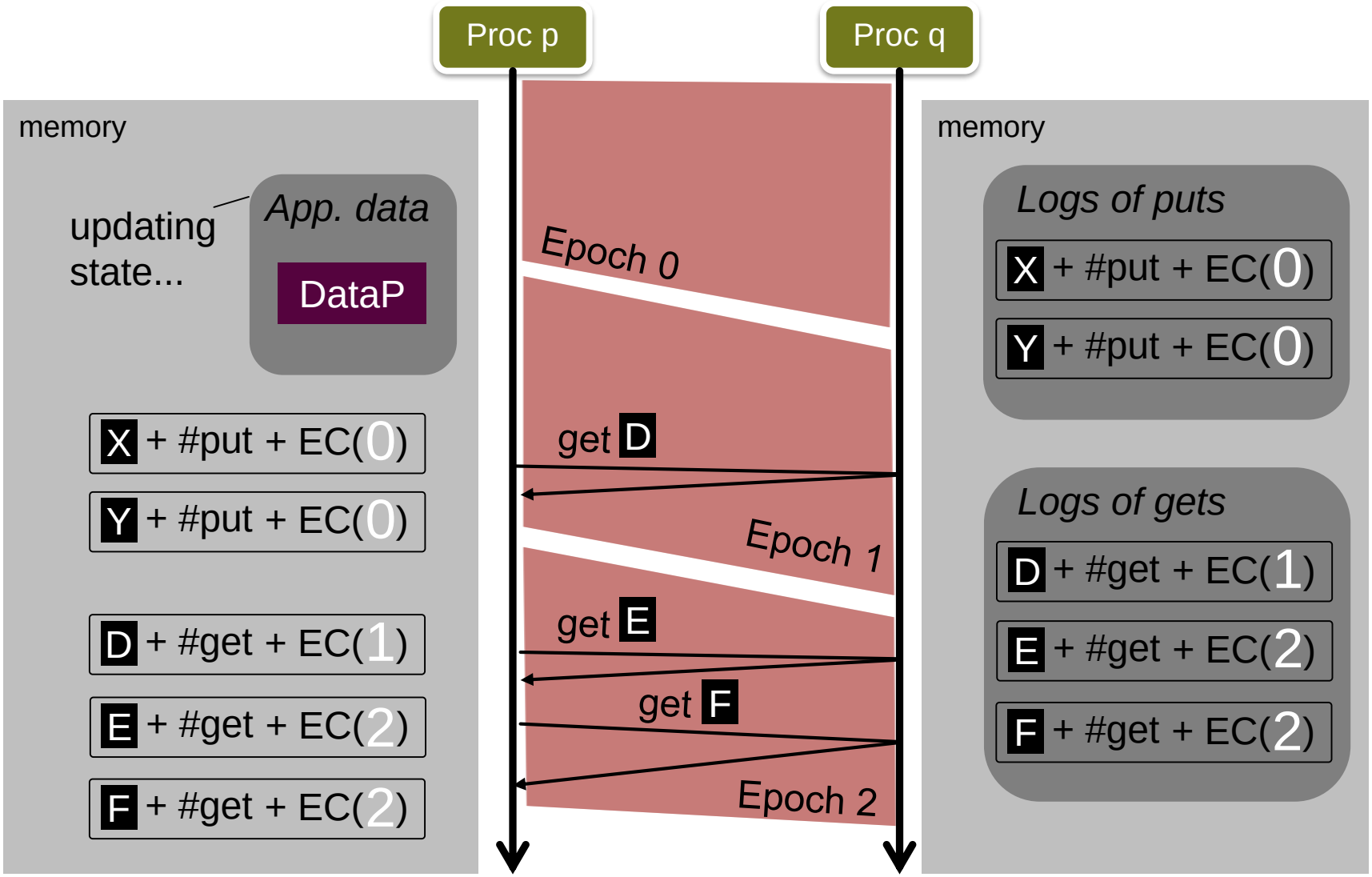
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



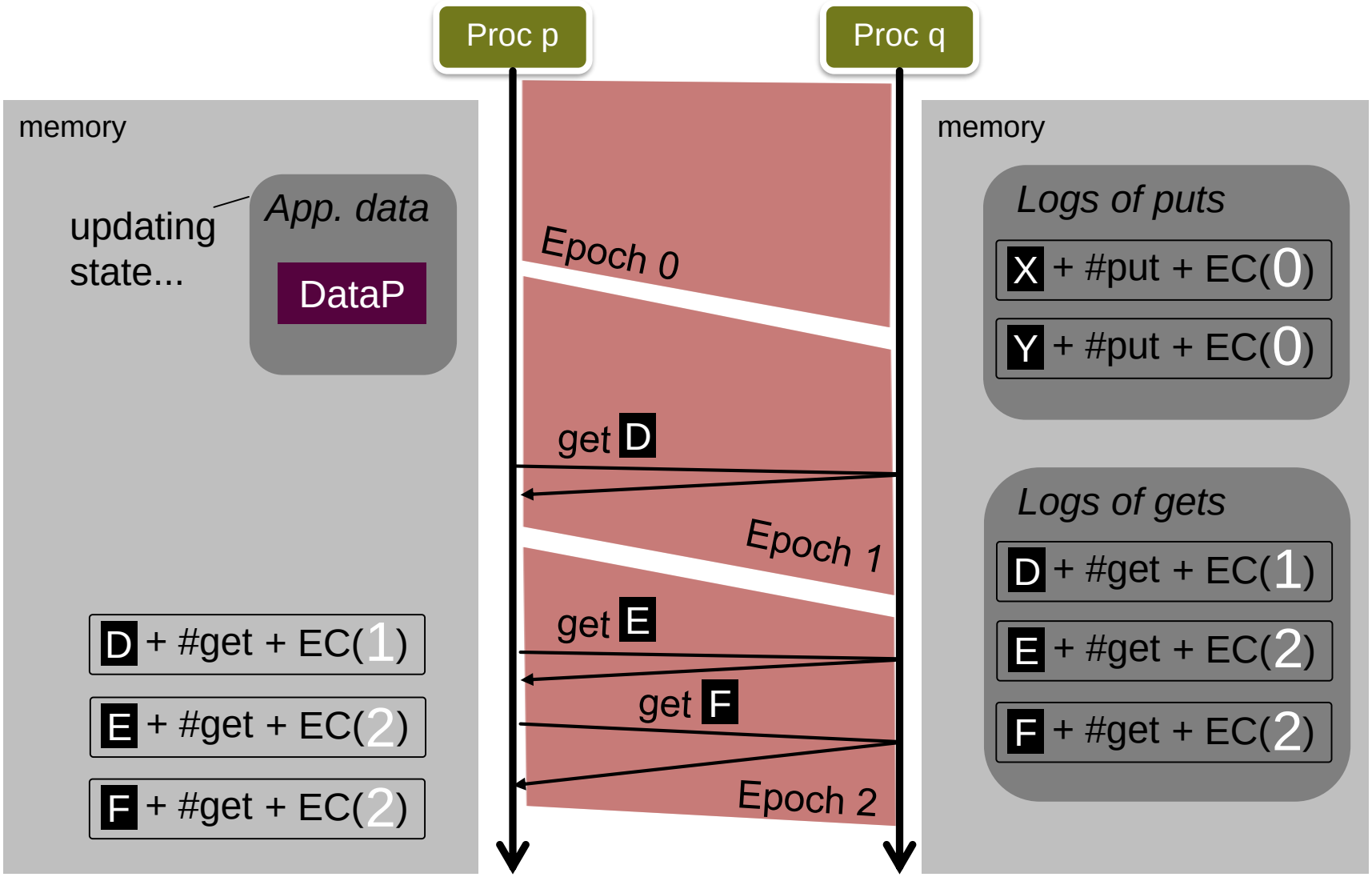
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



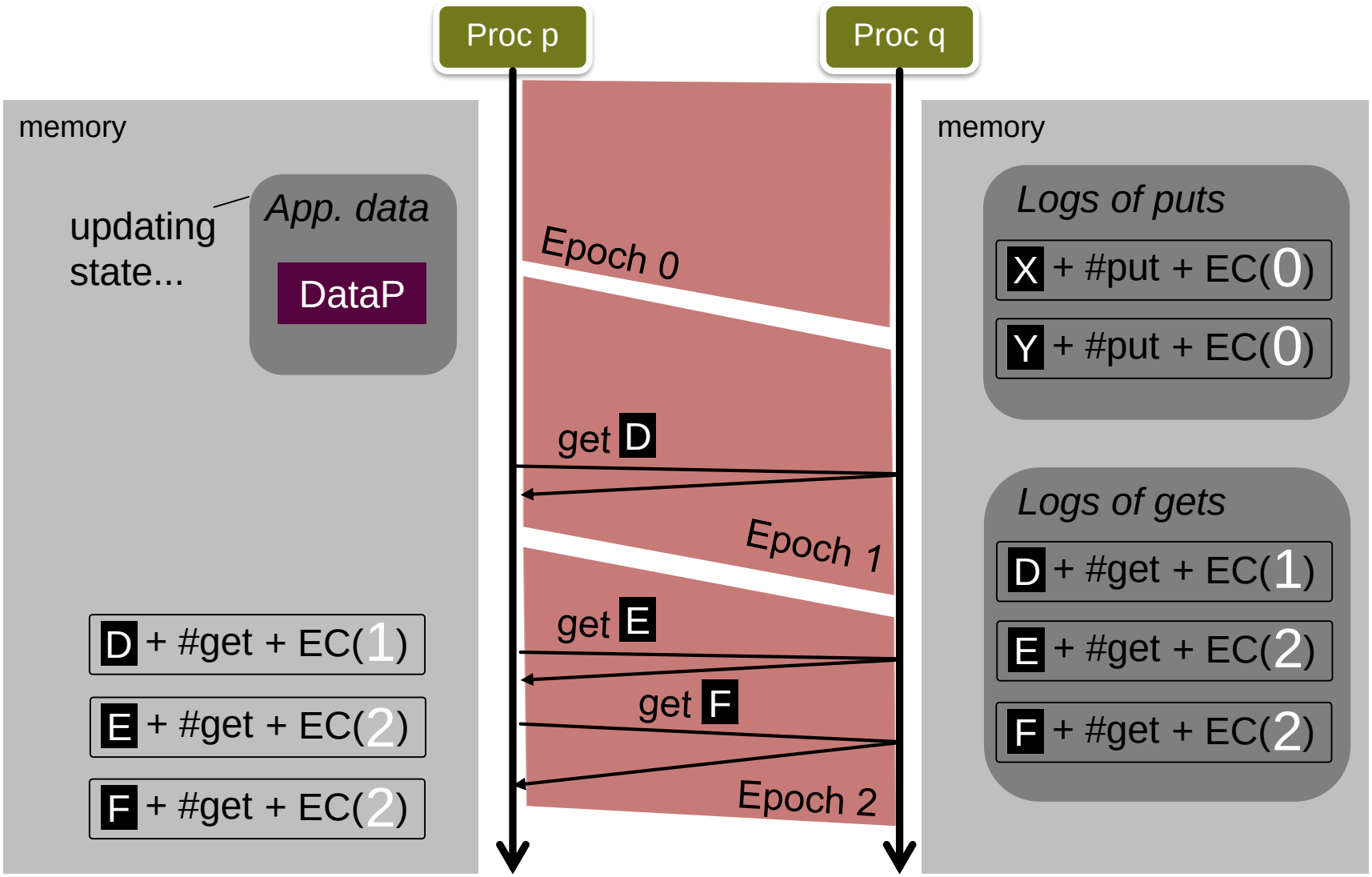
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



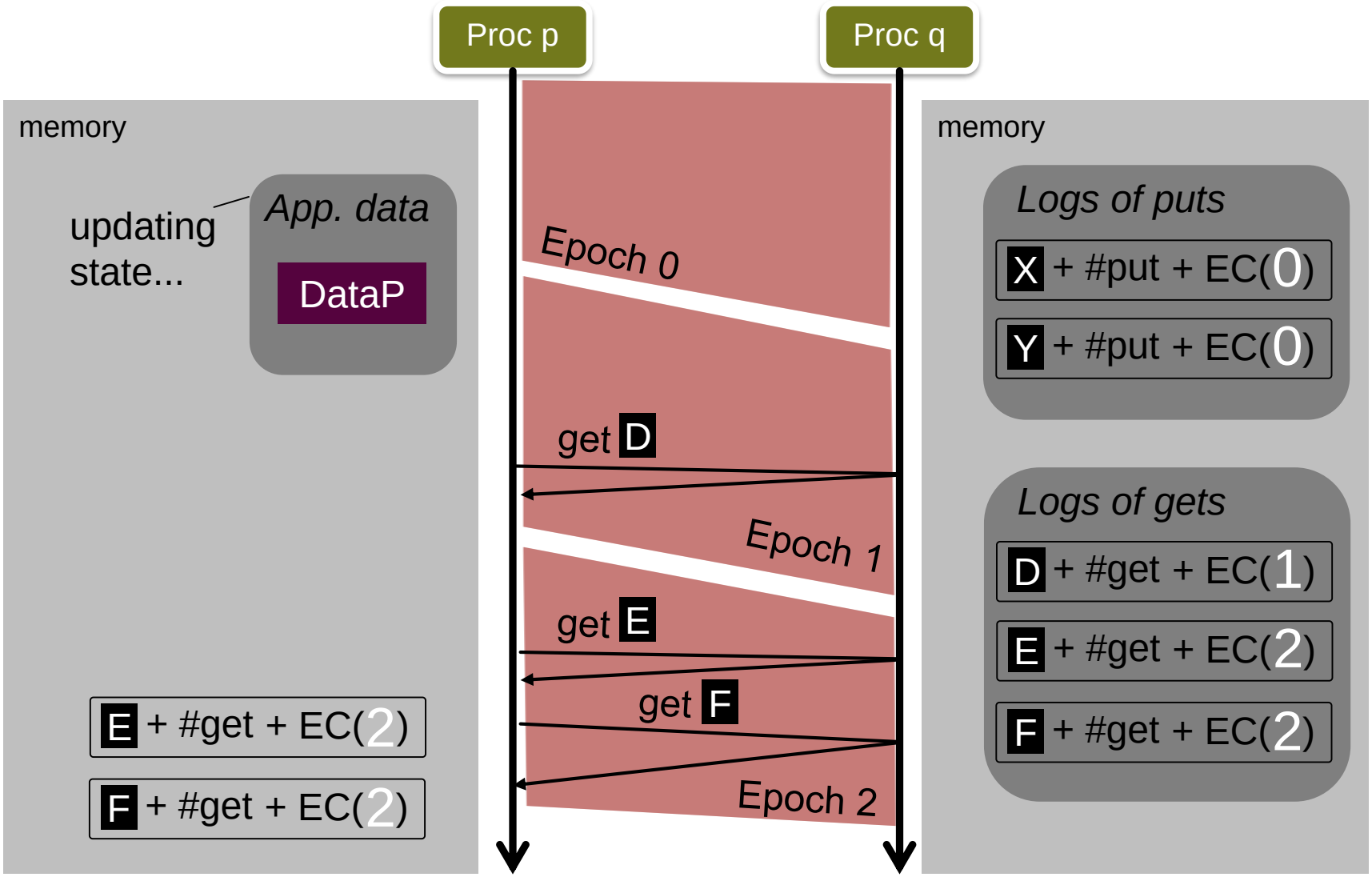
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



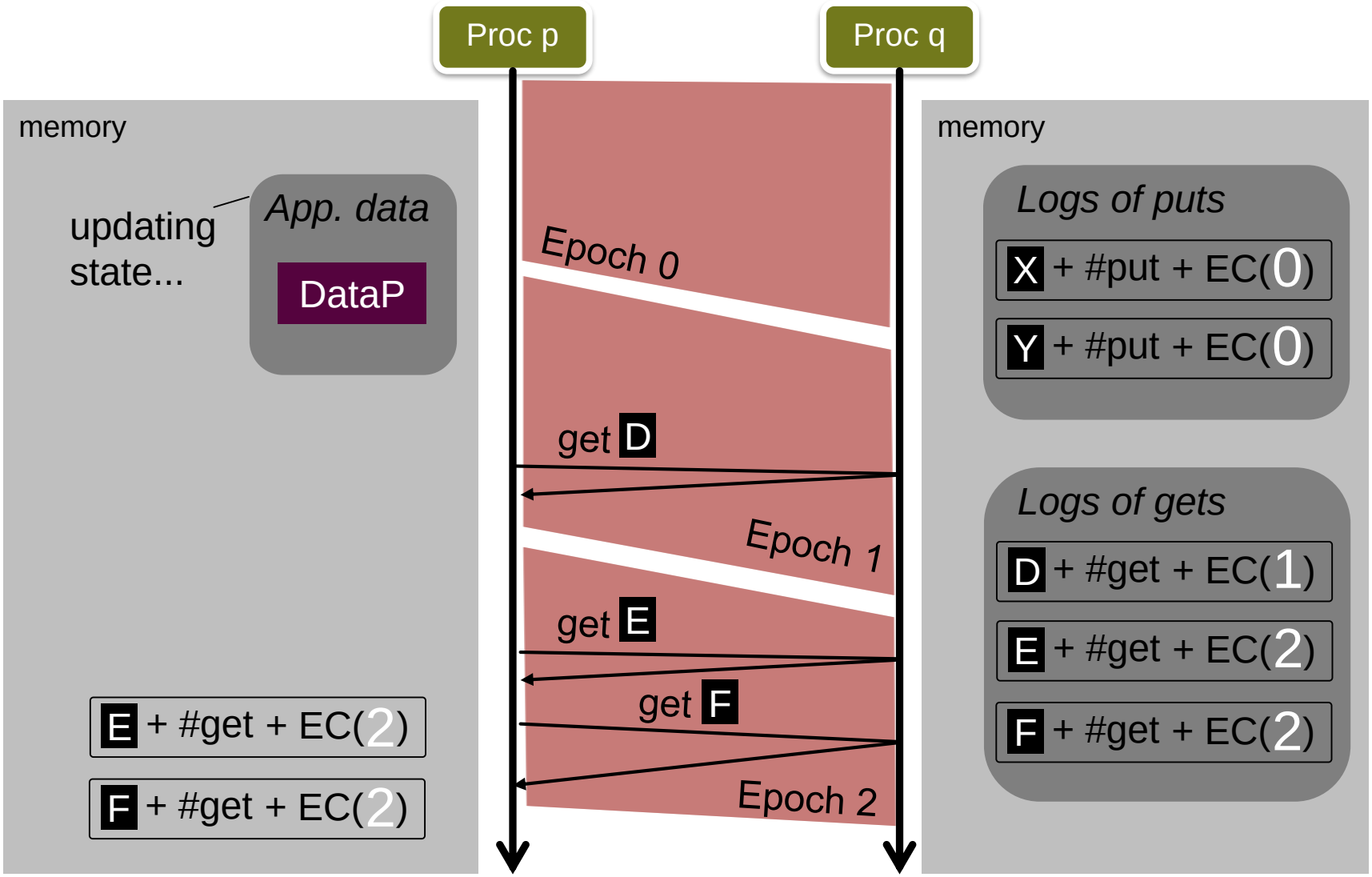
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



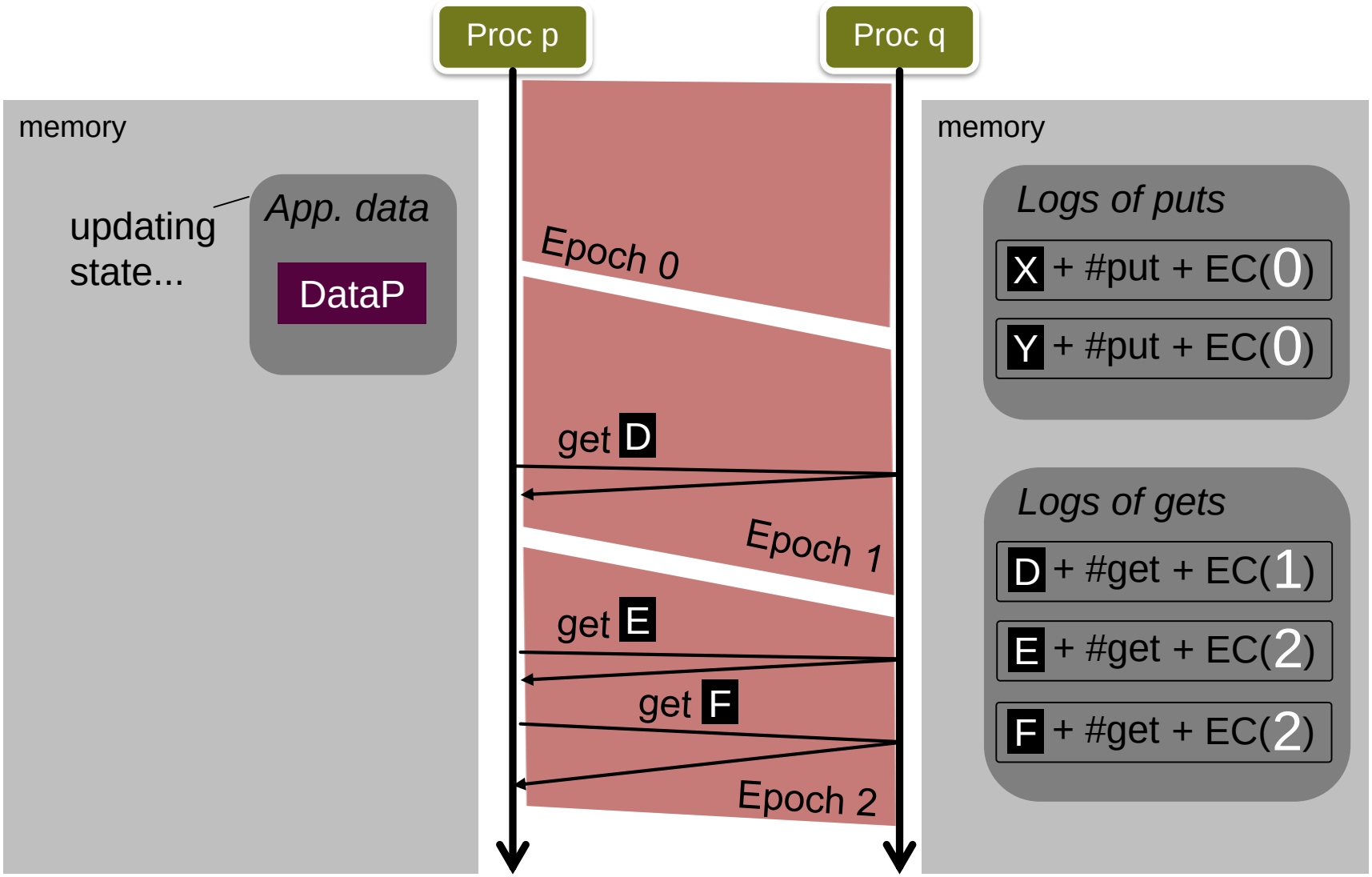
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



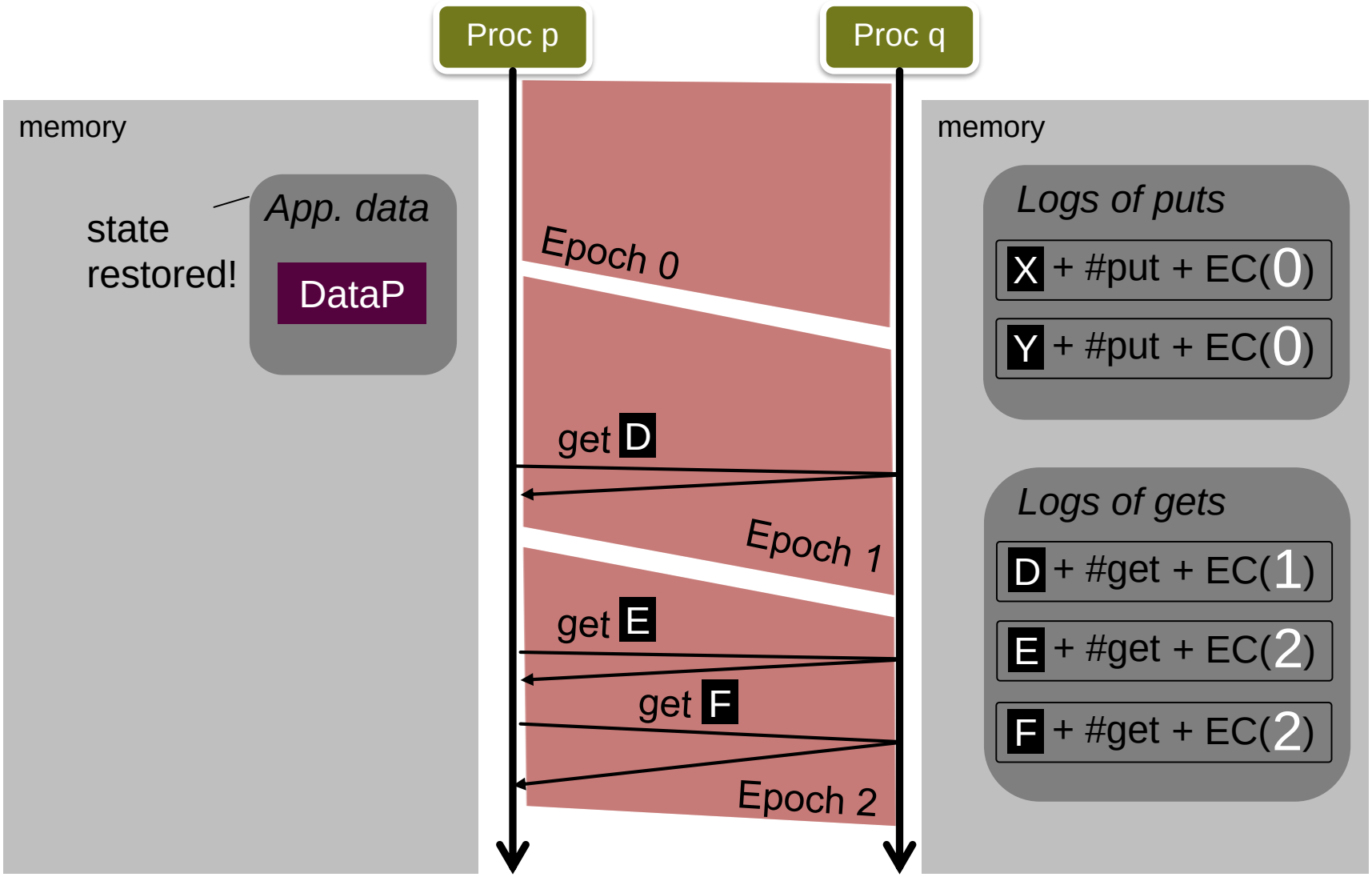
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

MP vs. RMA

Schemes

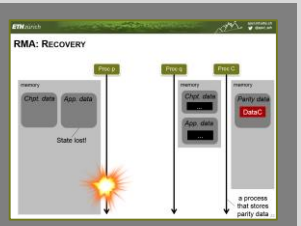
UC in RMA

MP vs. RMA

Checkpointing schemes

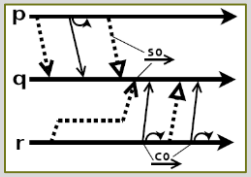
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

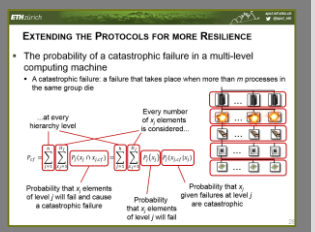
Performance

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes



Decreasing failure prob.

Extending the Protocols for more Resilience

Extending the Protocols for more Resilience

- Today's supercomputers have a hierarchical layout

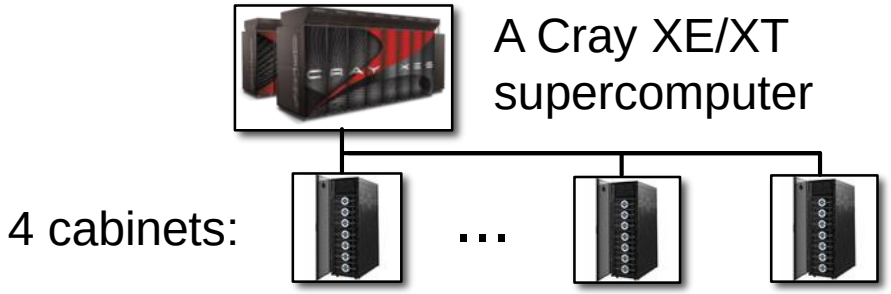
Extending the Protocols for more Resilience

- Today's supercomputers have a hierarchical layout

A Cray XE/XT supercomputer

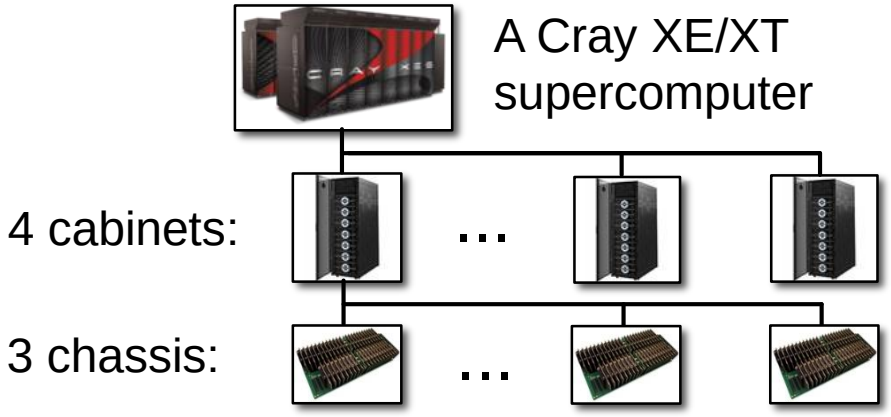
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



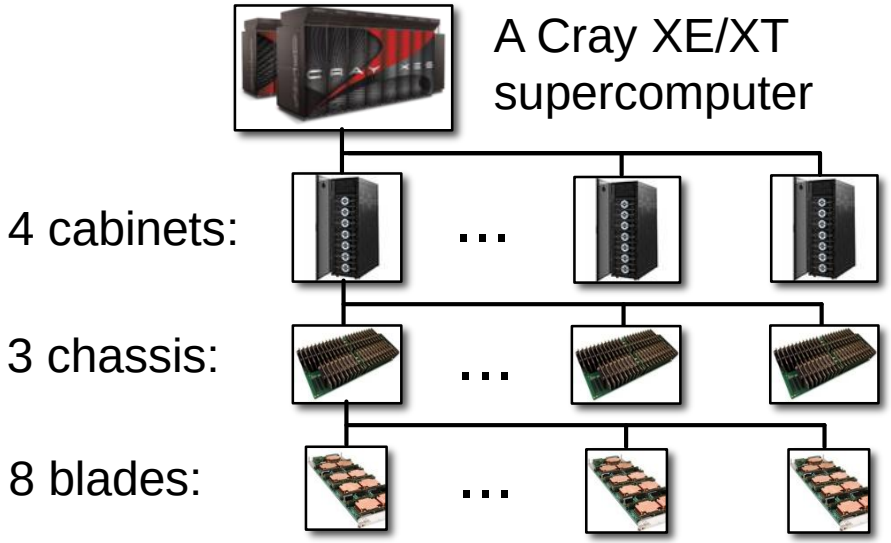
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



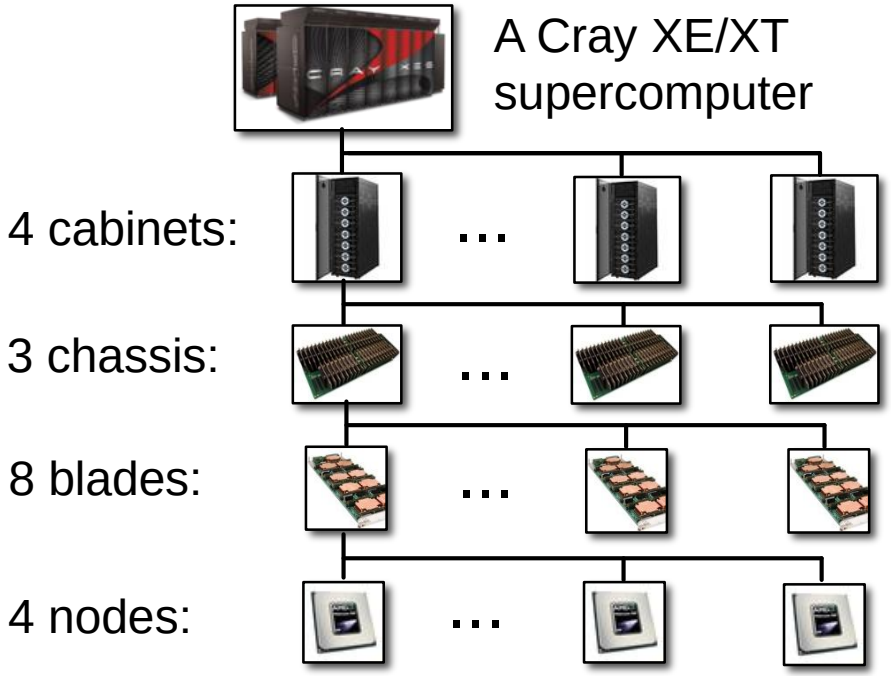
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



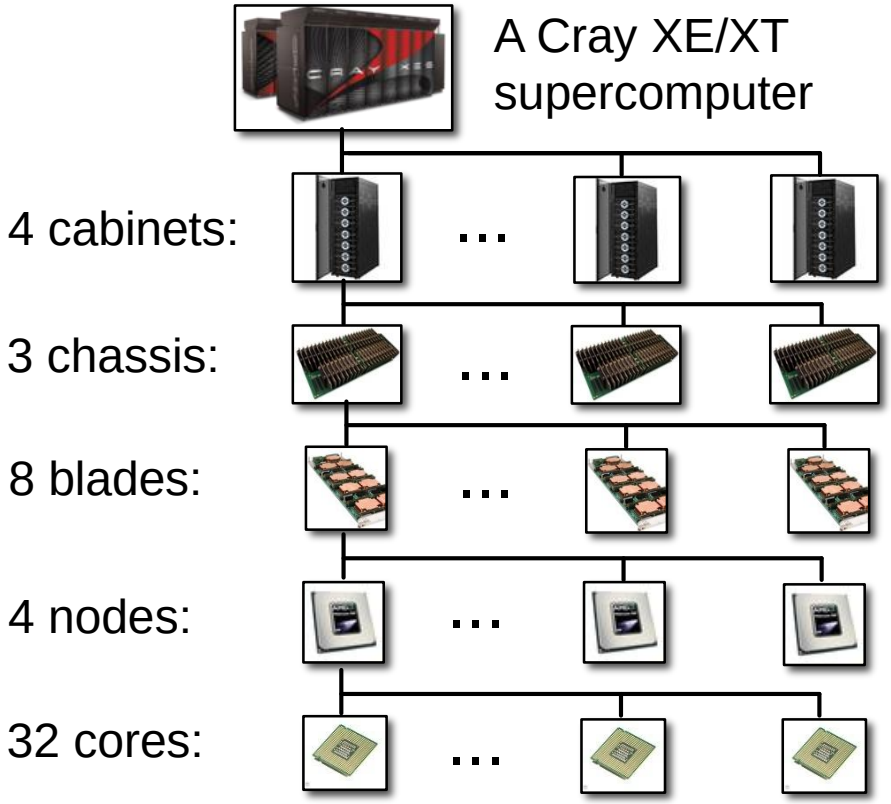
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



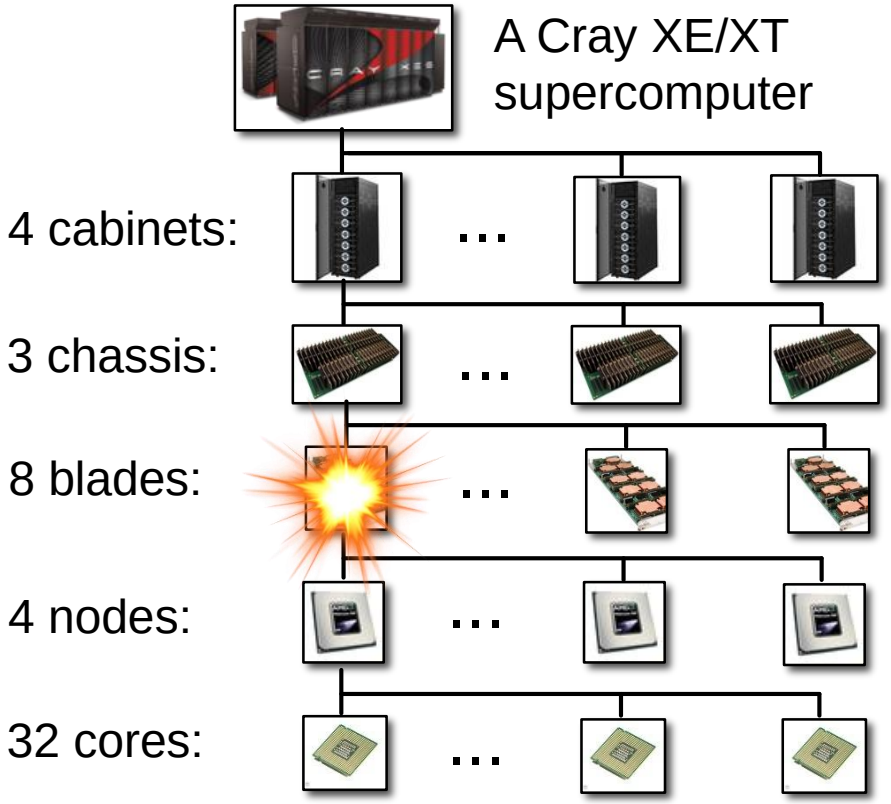
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



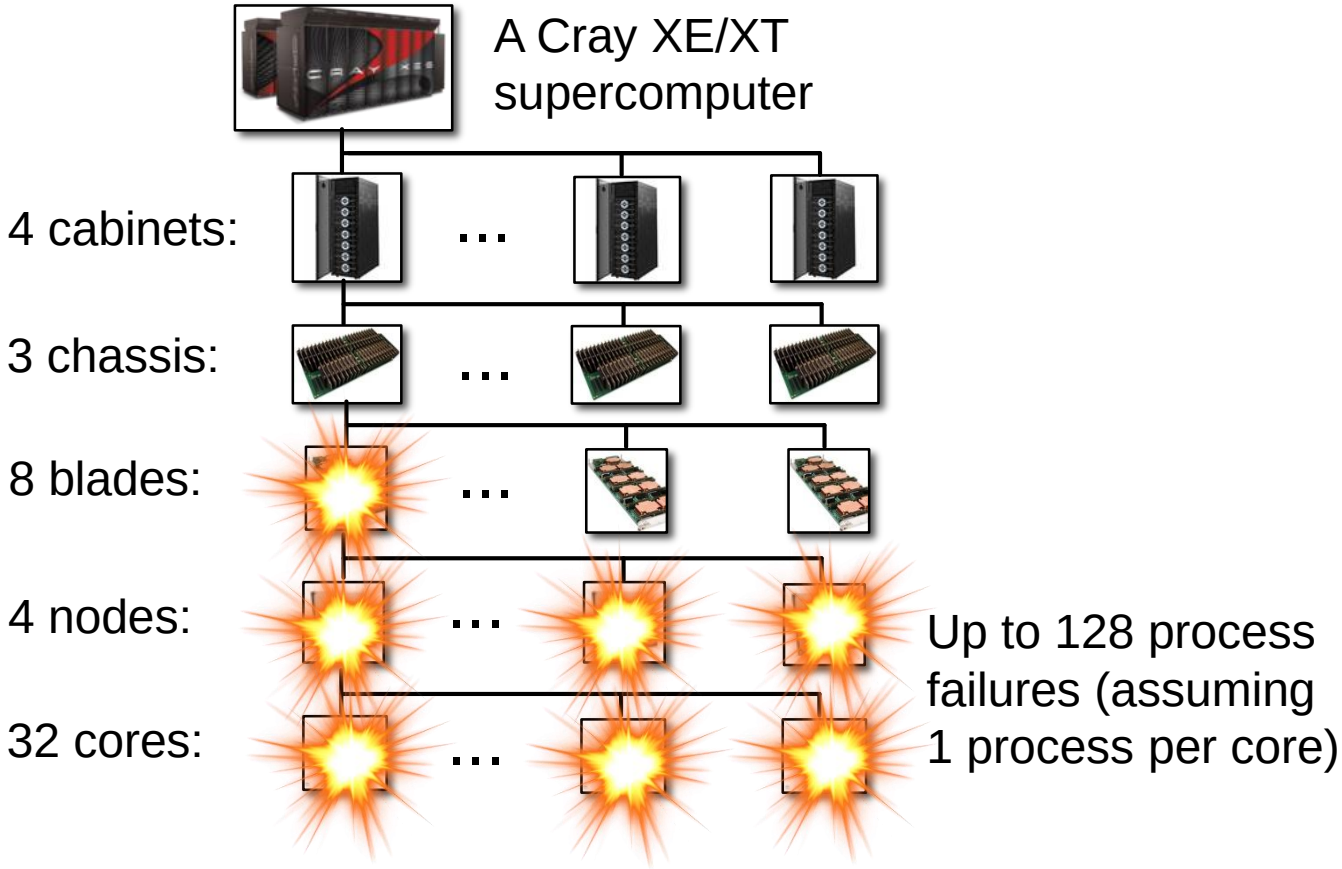
Extending the Protocols for more Resilience

- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes



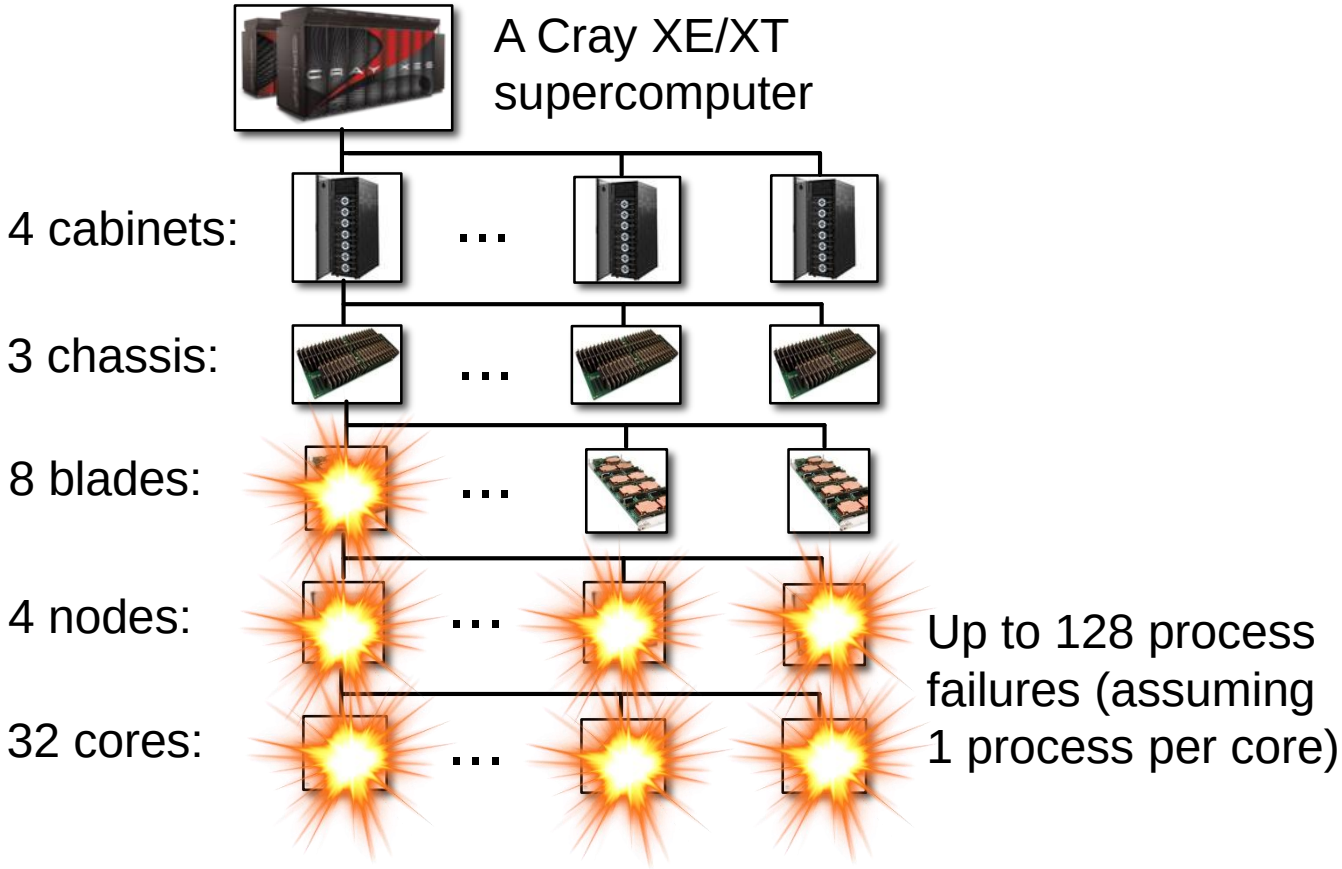
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes



Extending the Protocols for more Resilience

- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes
- Introduced protocols usually cannot handle > 1 process crash



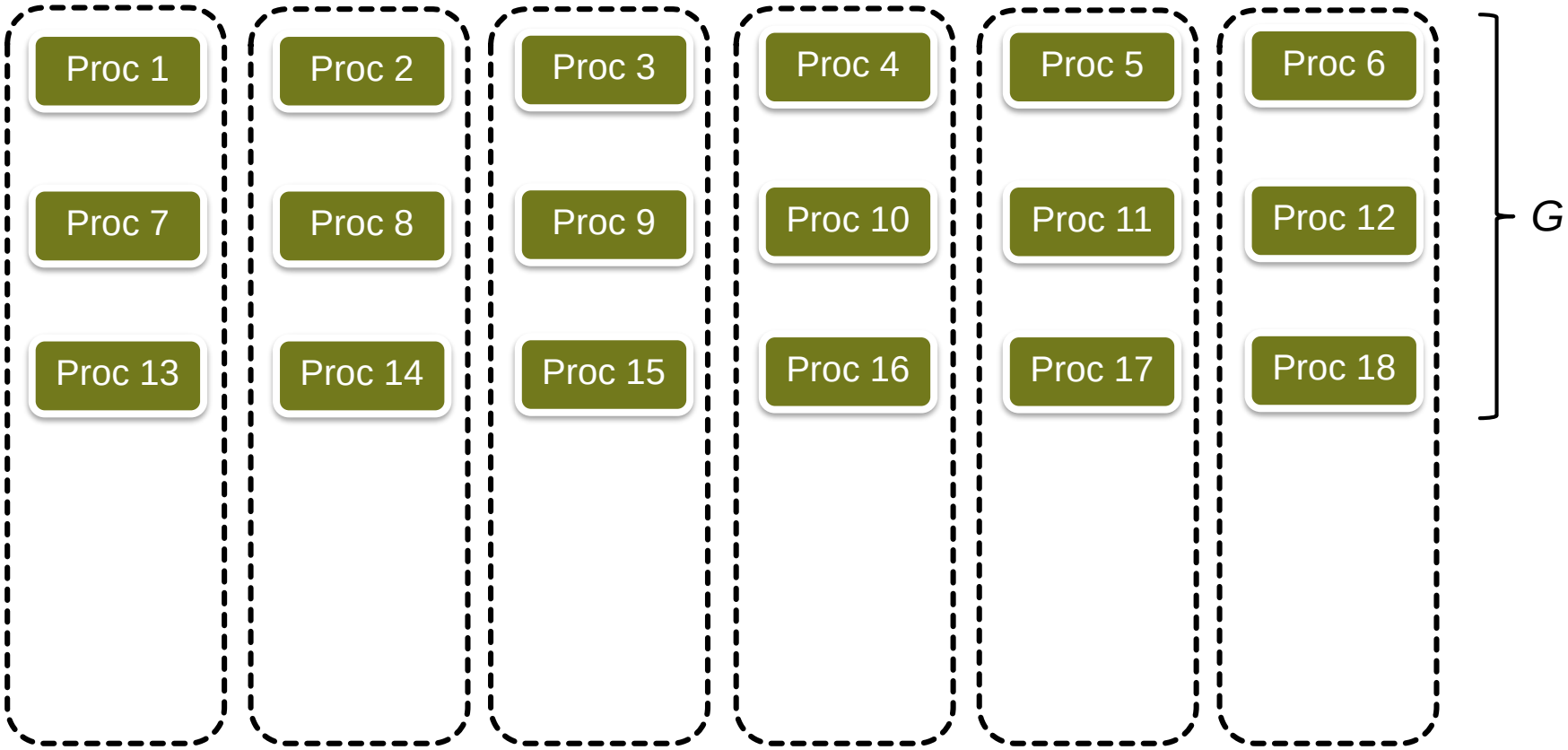
Extending the Protocols for more Resilience

Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each

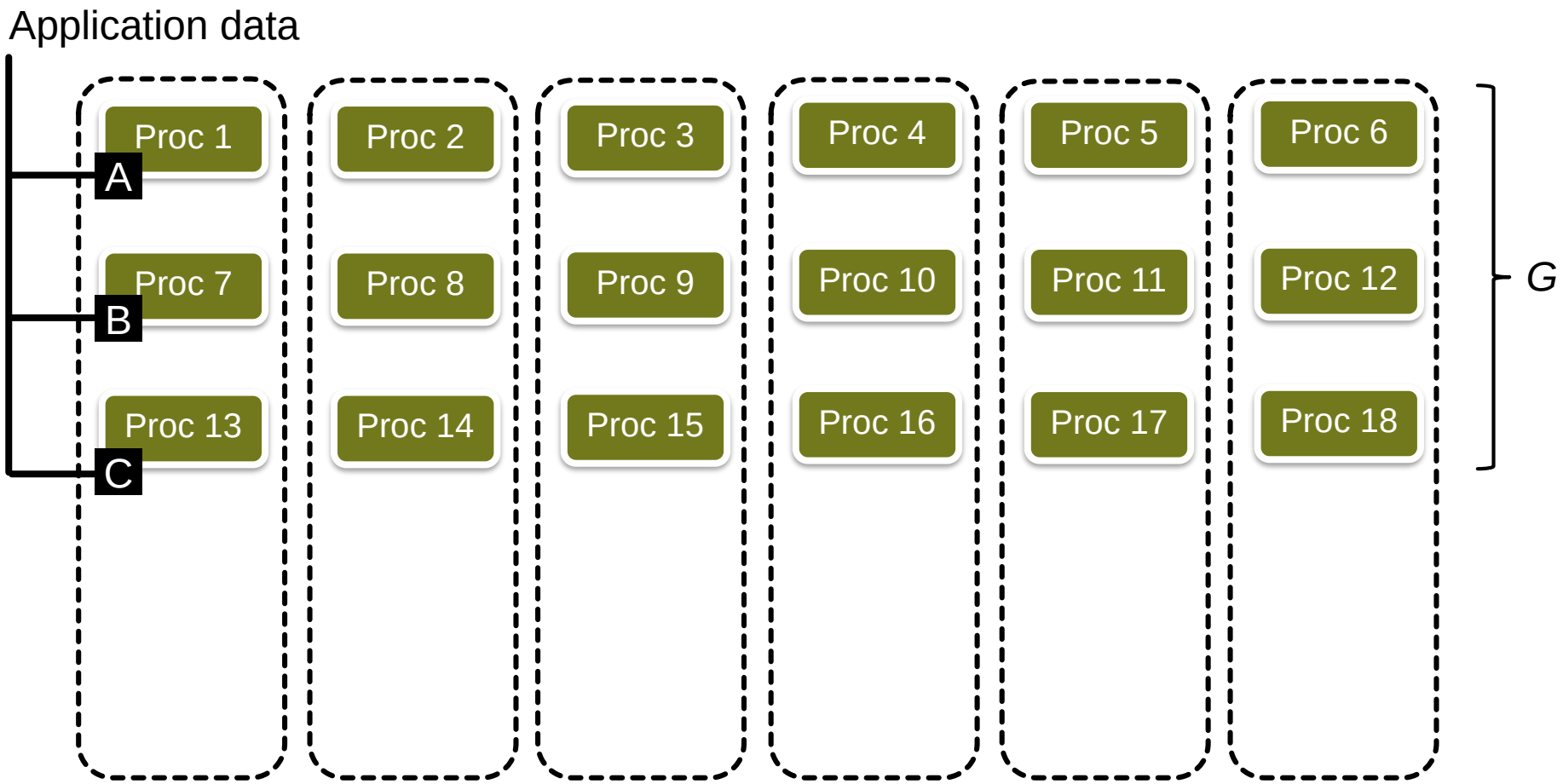
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each



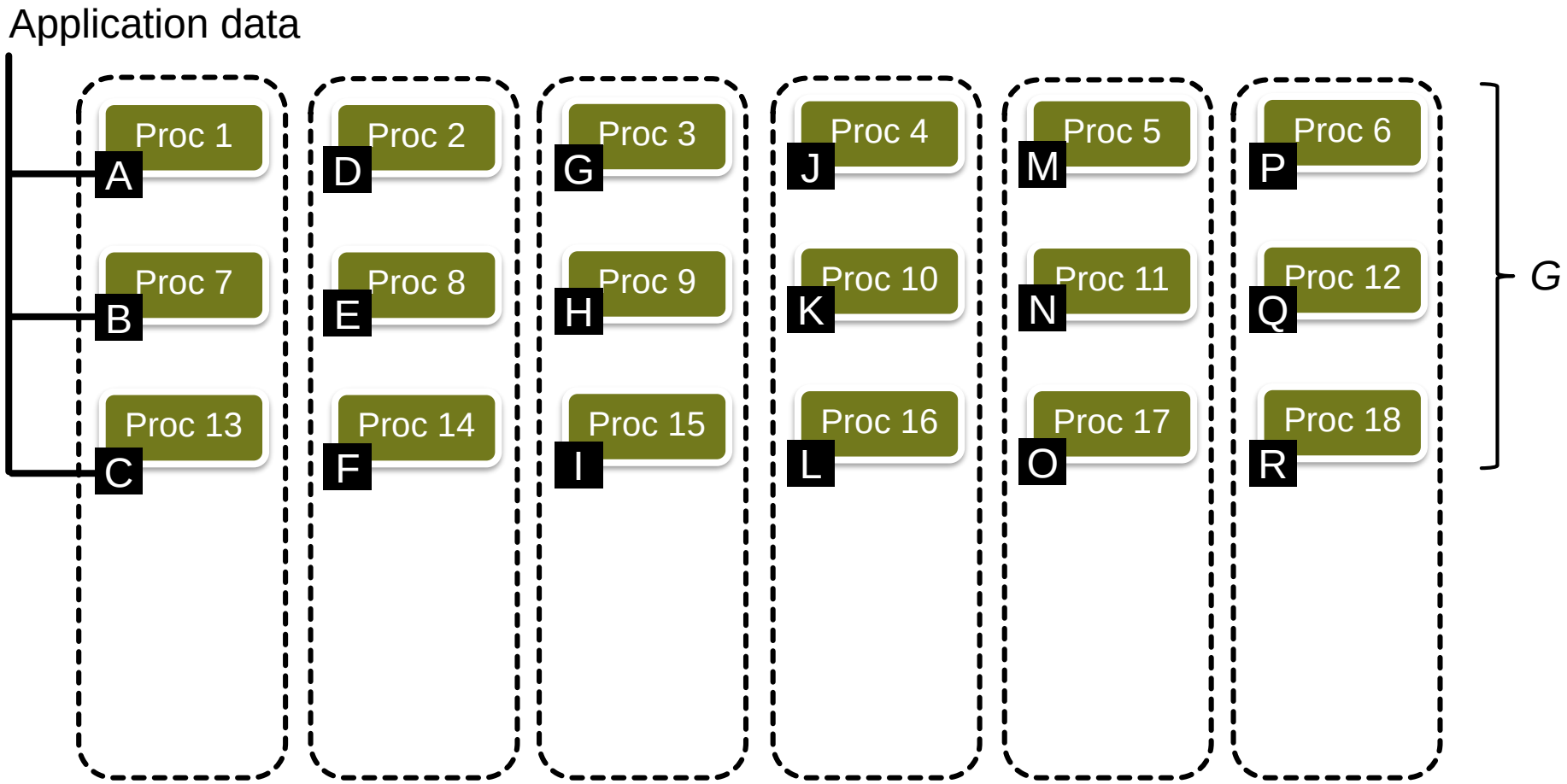
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each



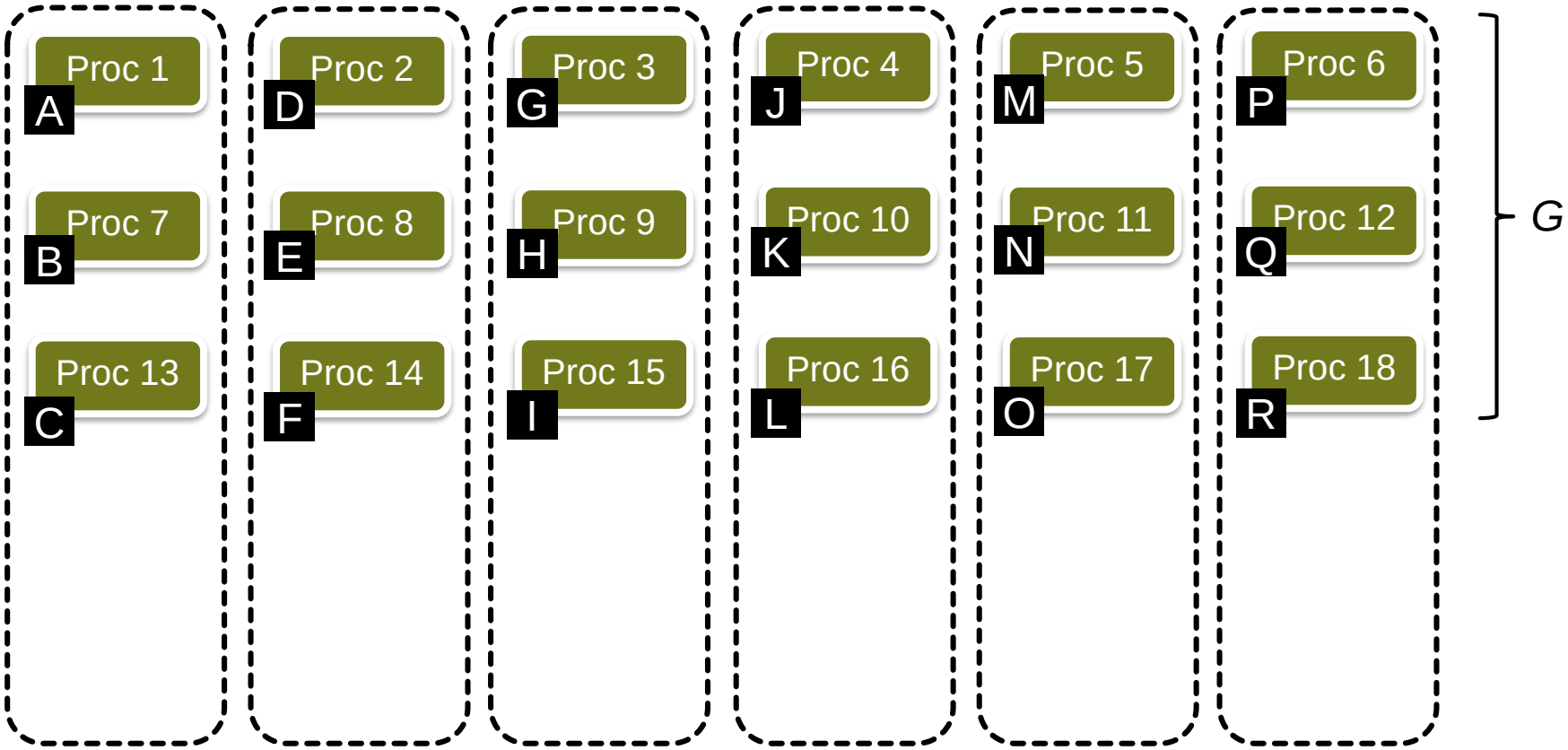
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each



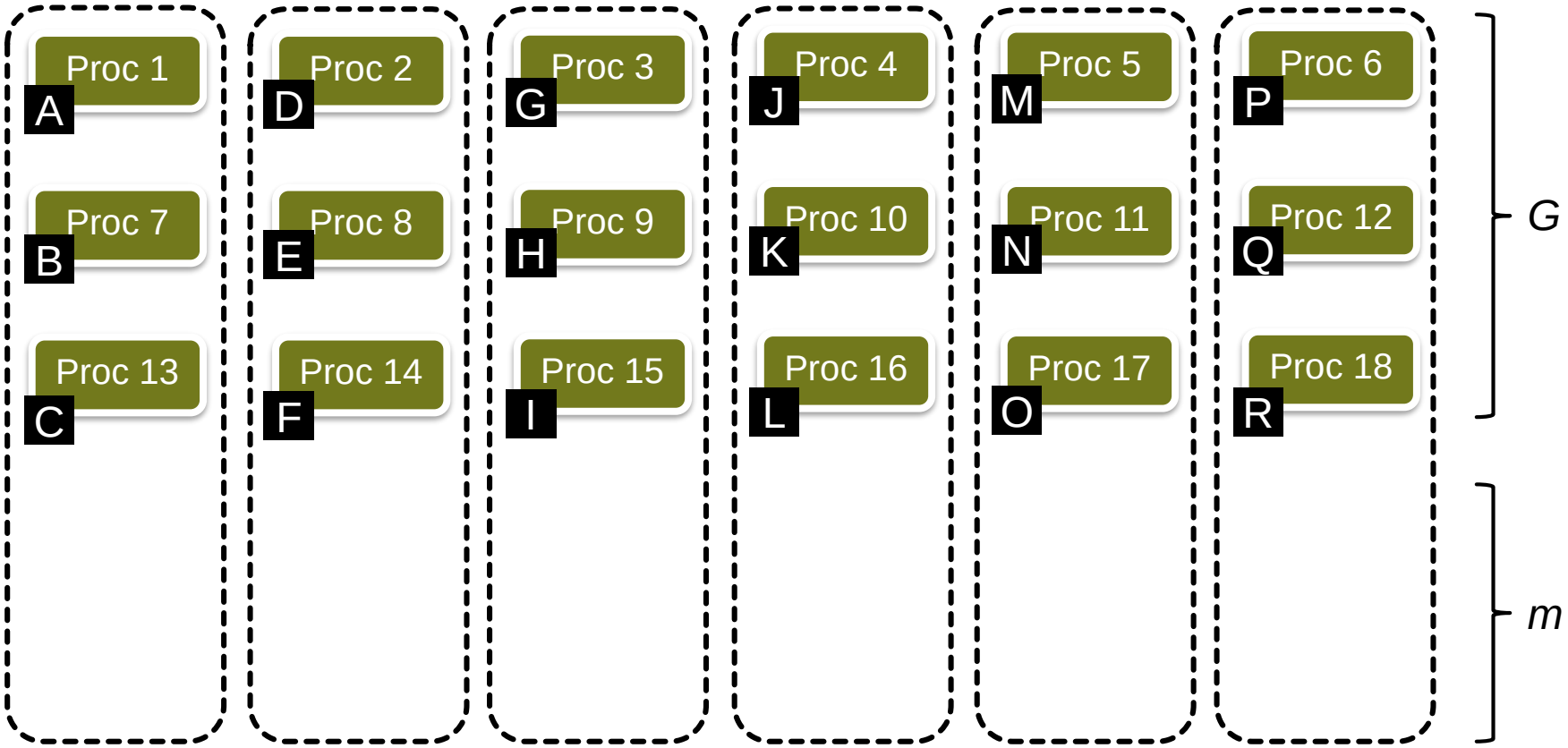
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each



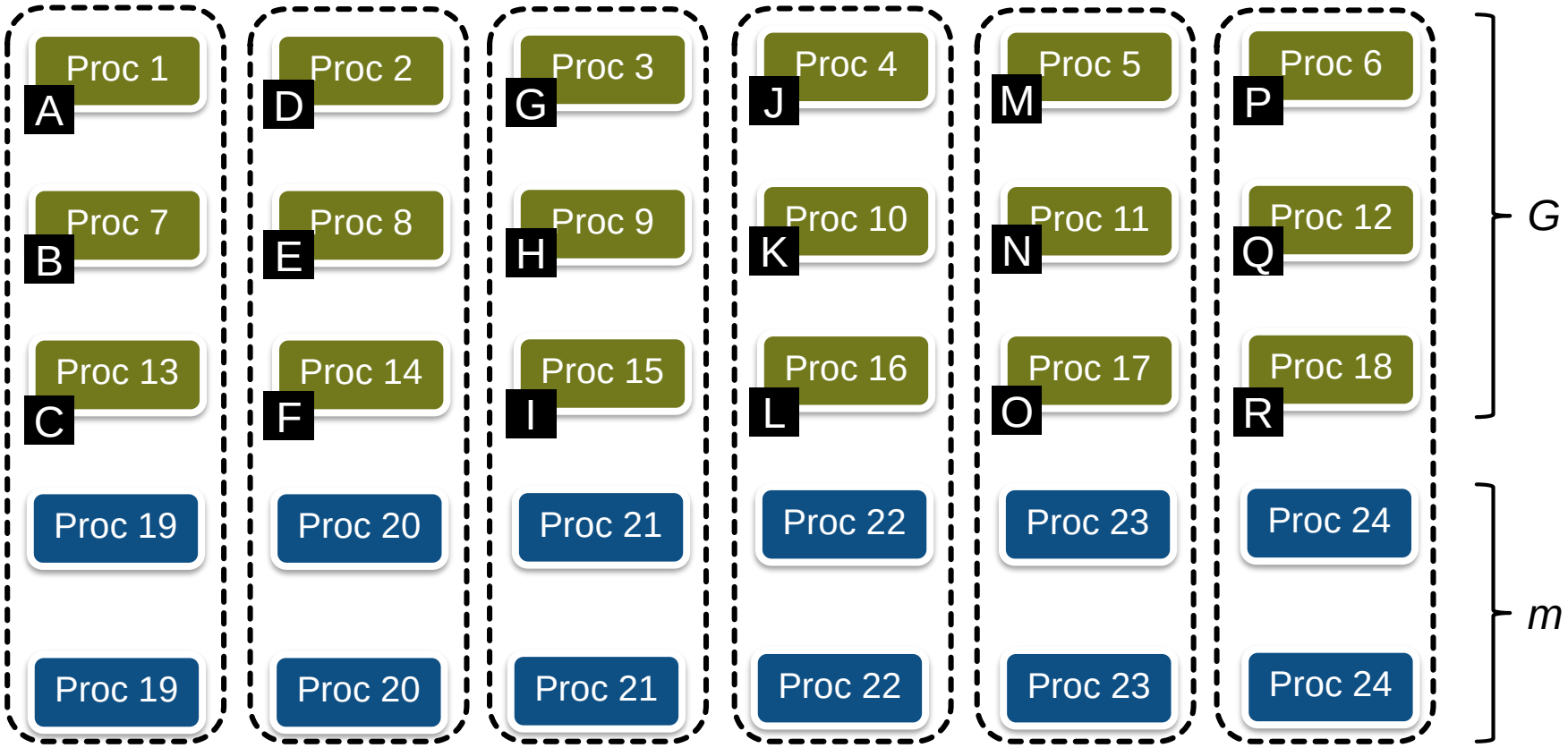
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



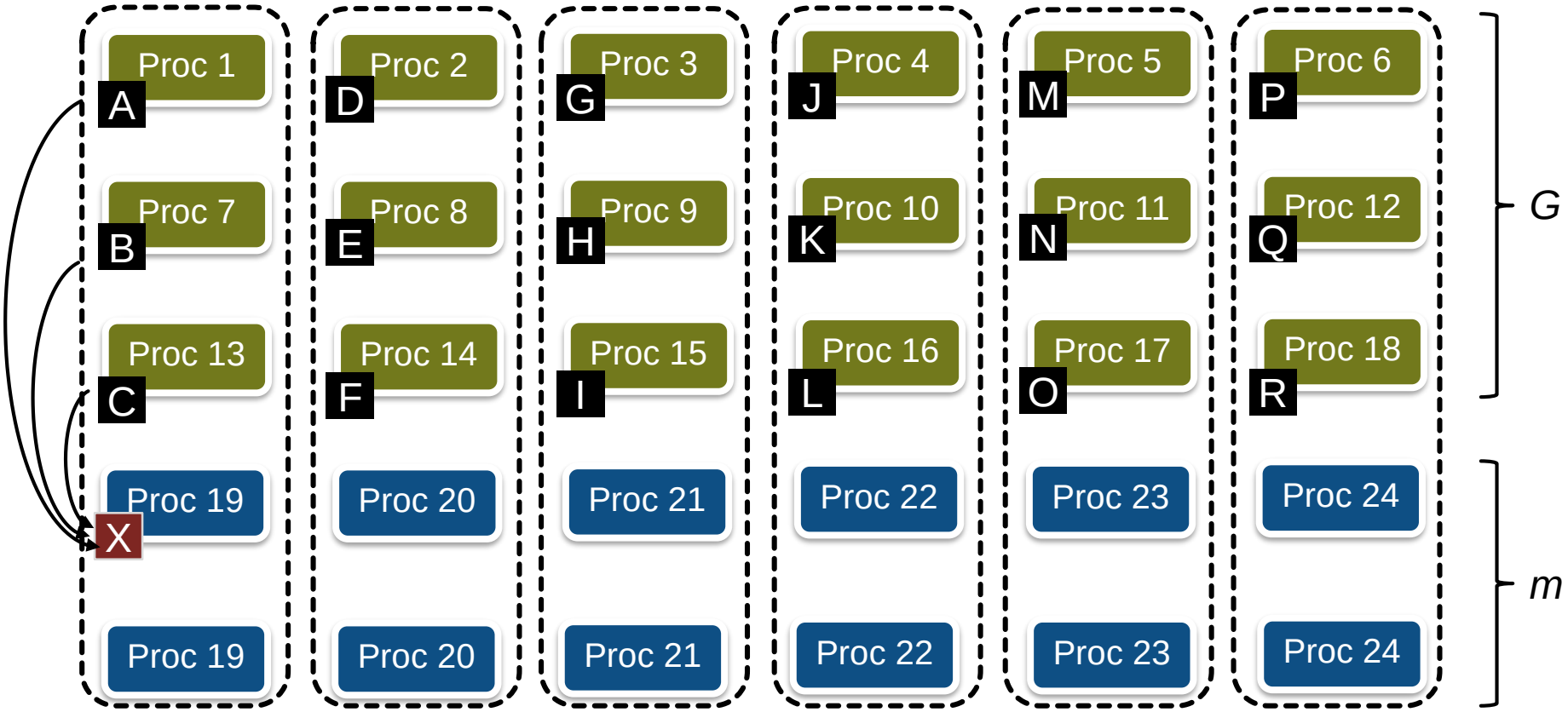
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



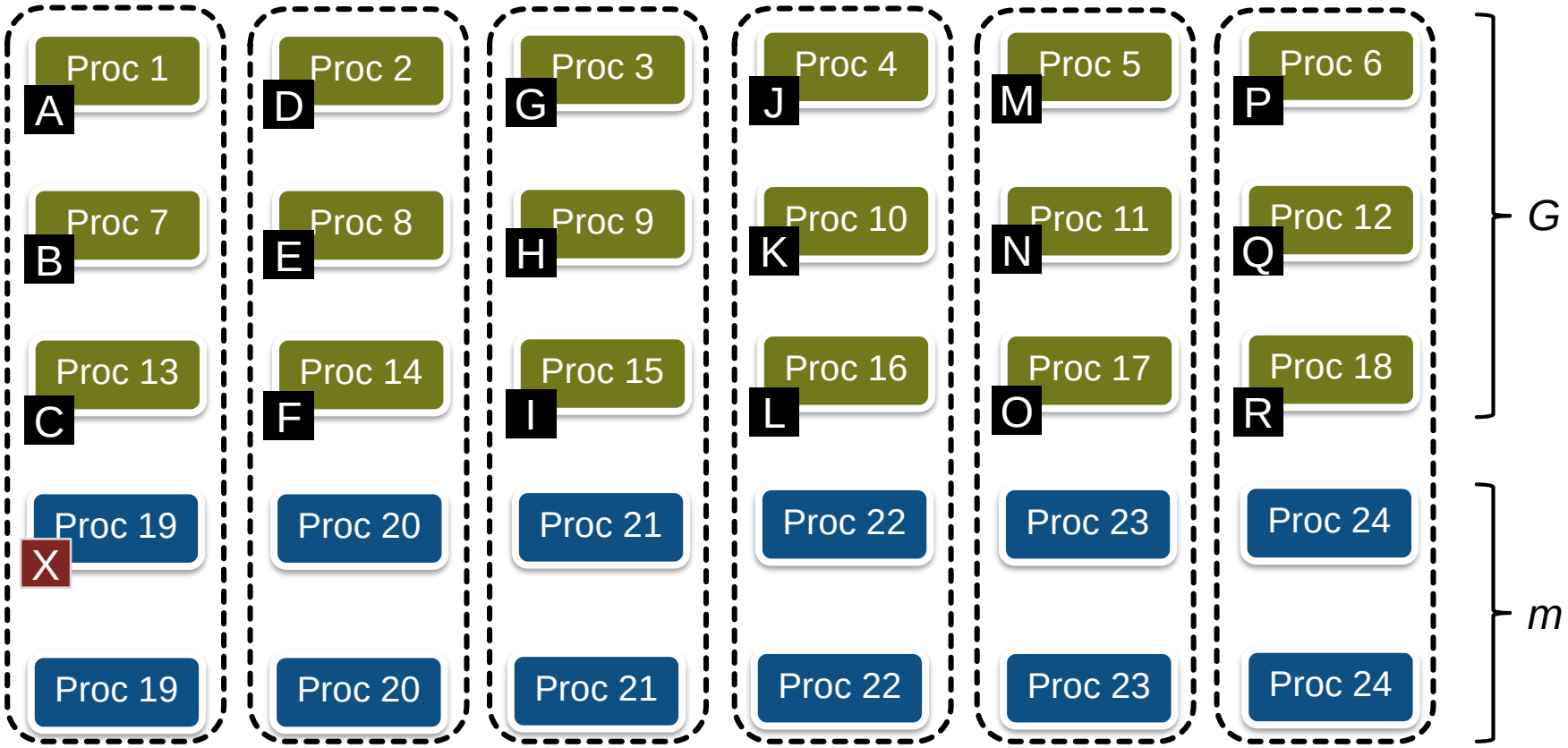
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



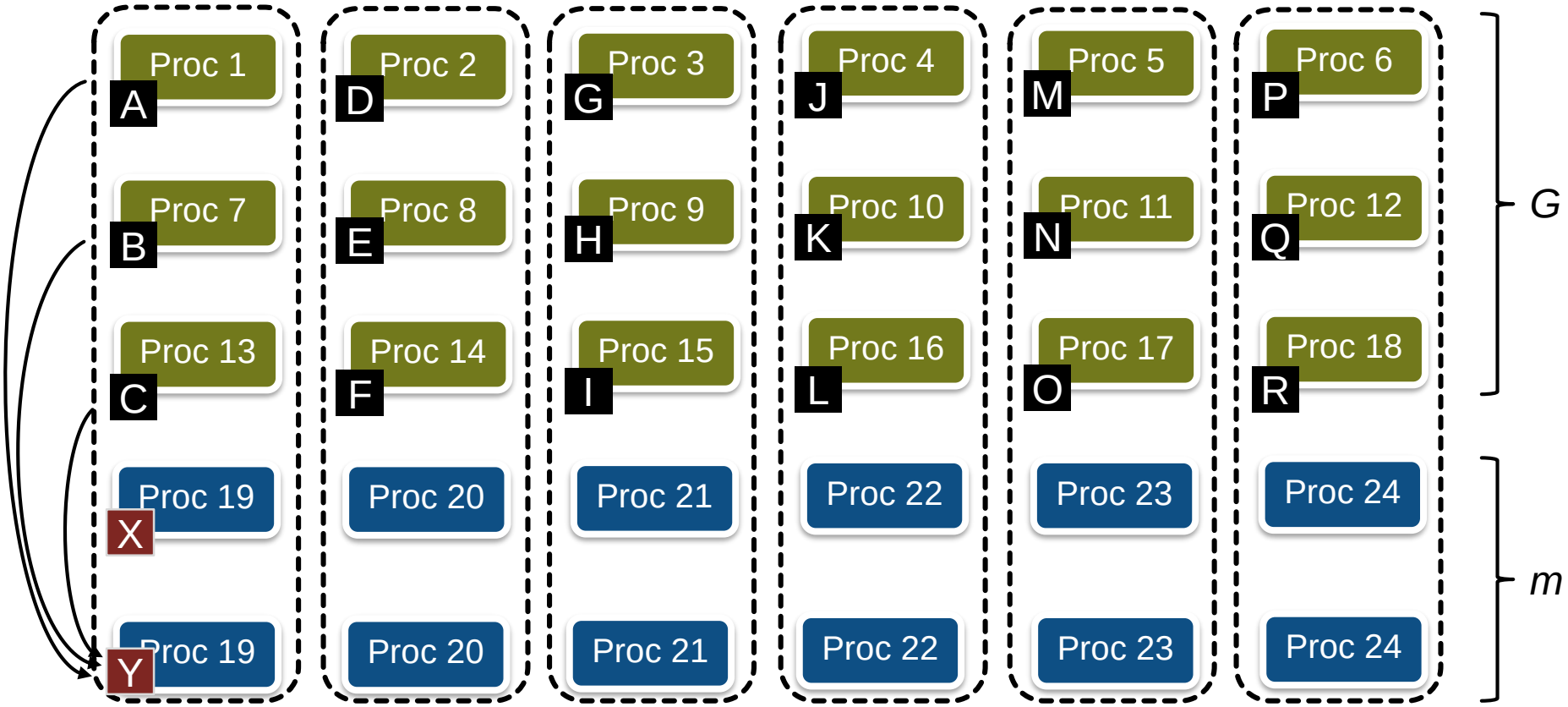
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



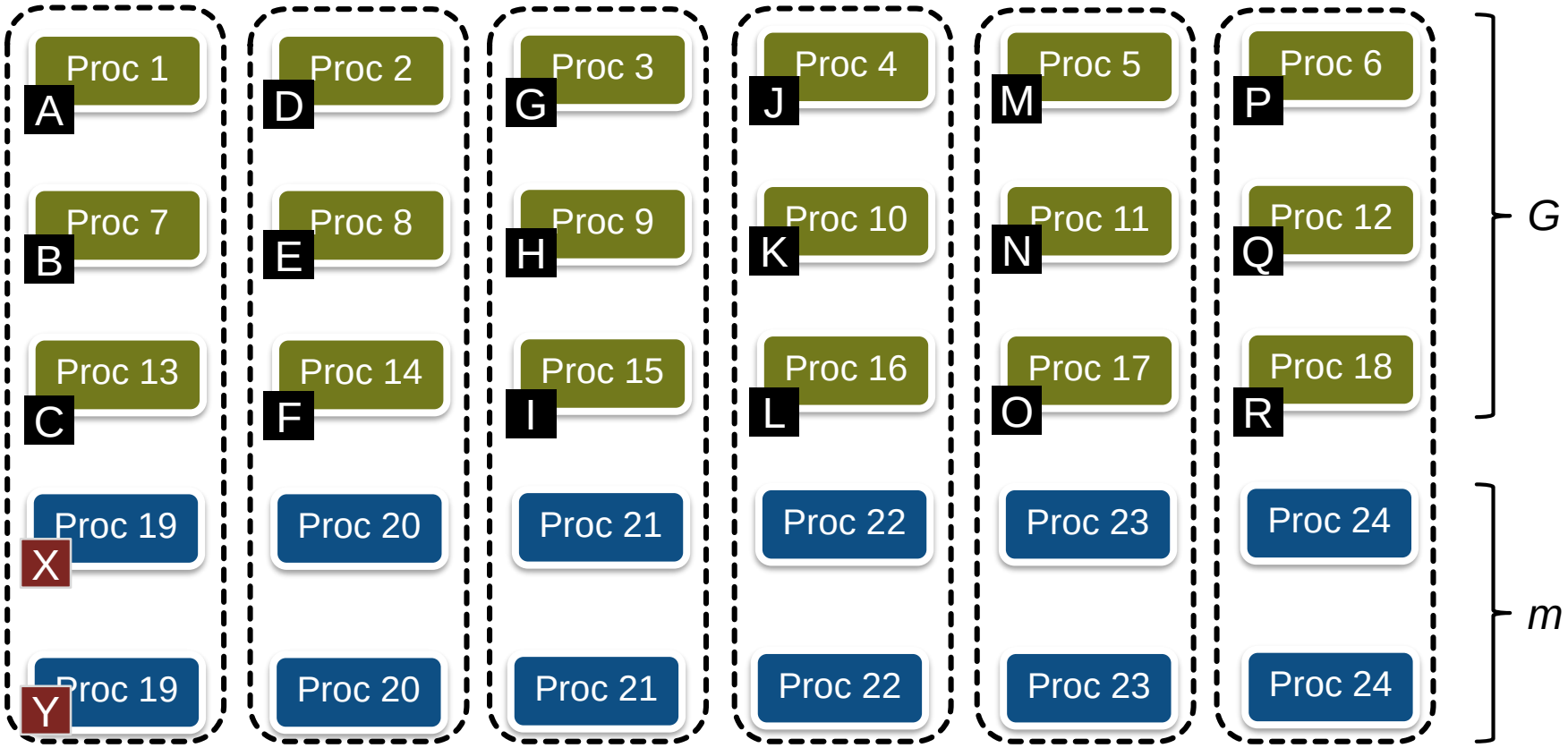
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



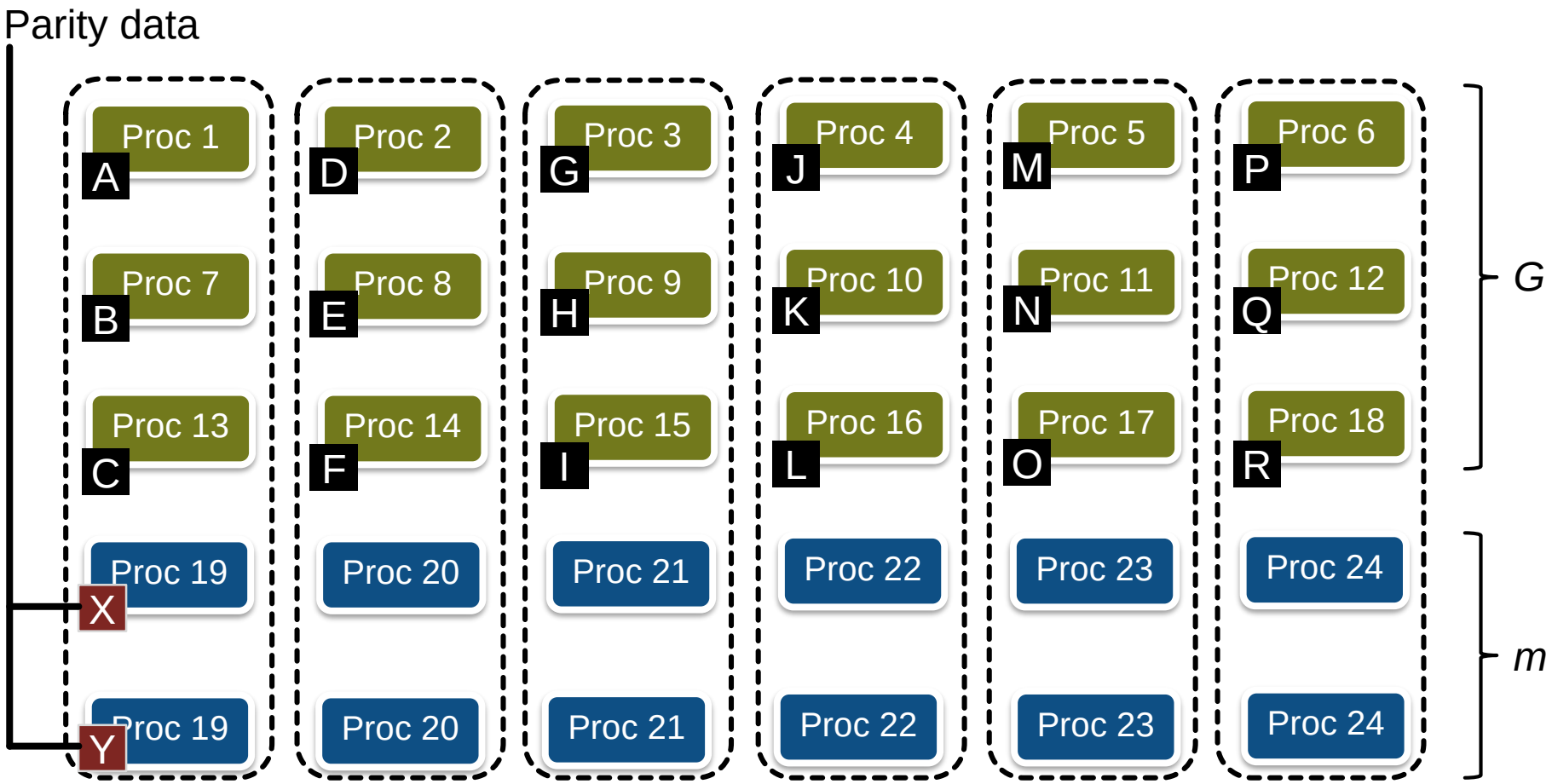
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



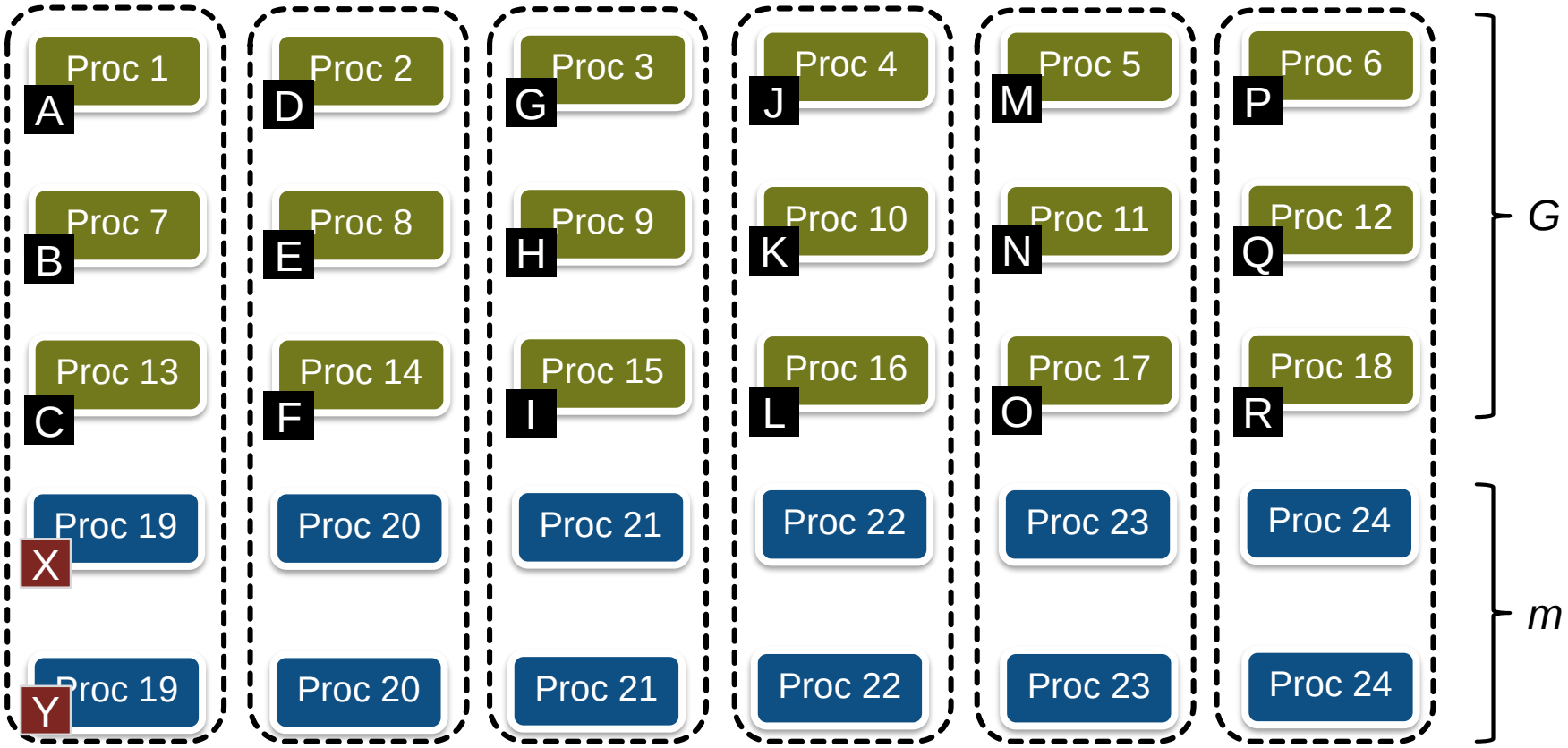
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



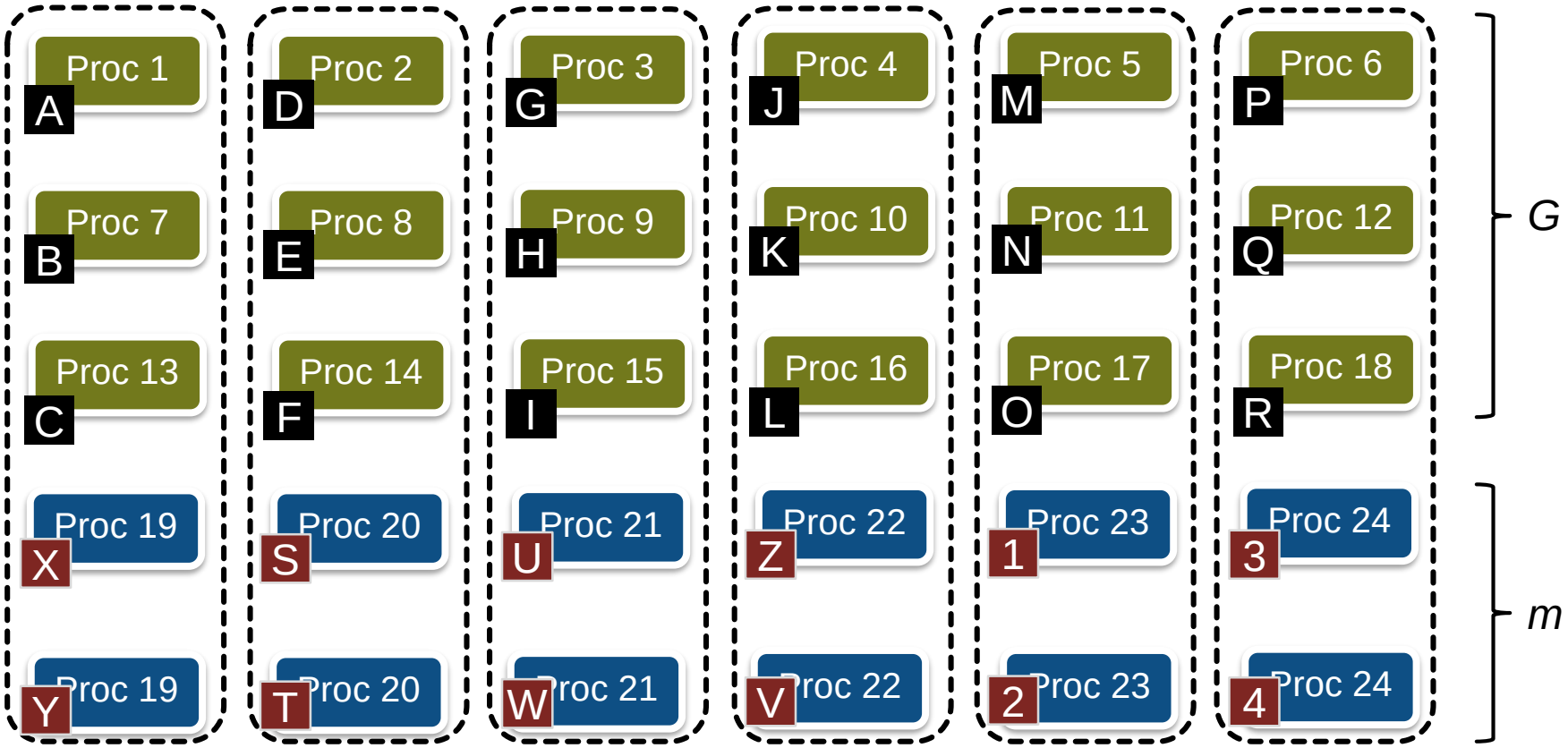
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



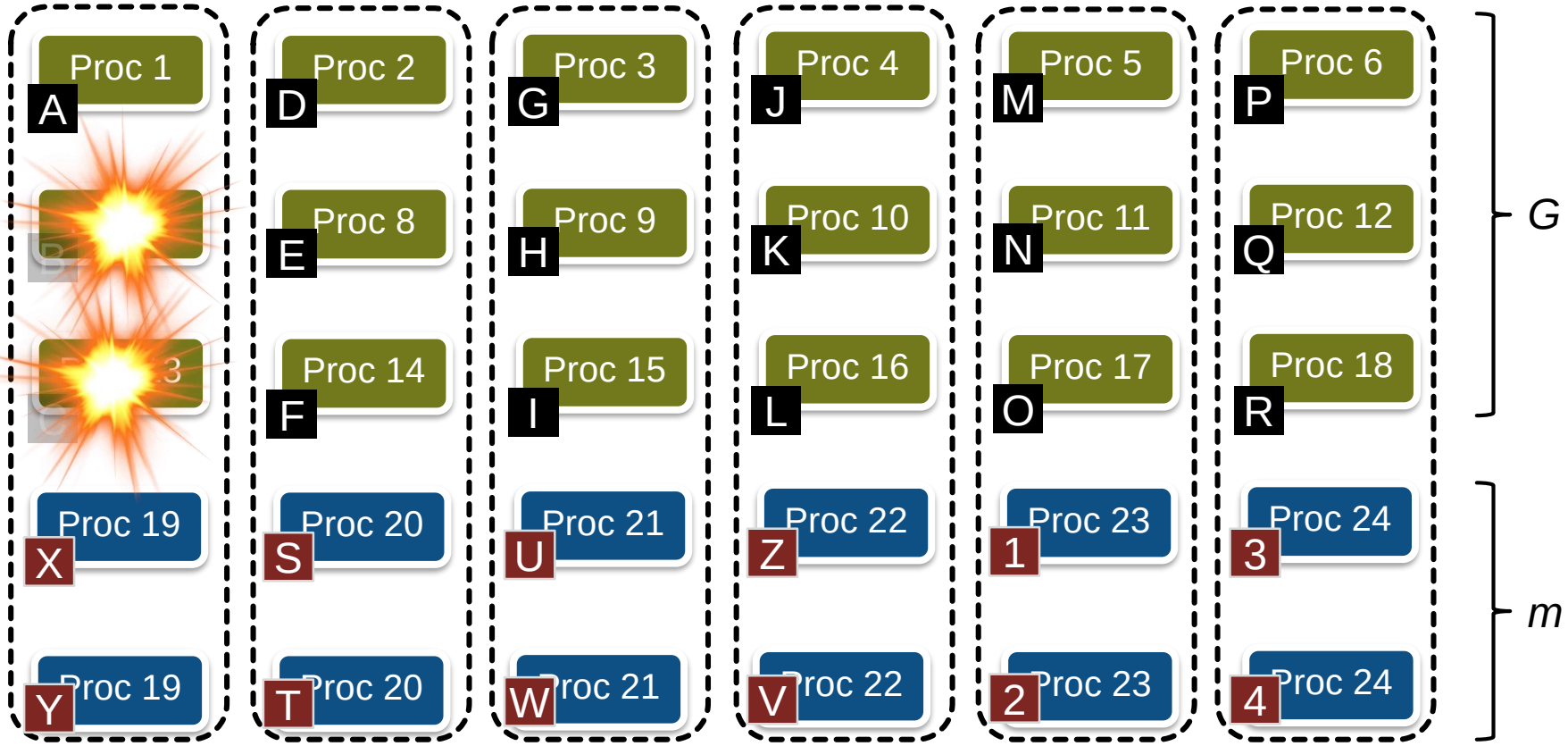
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



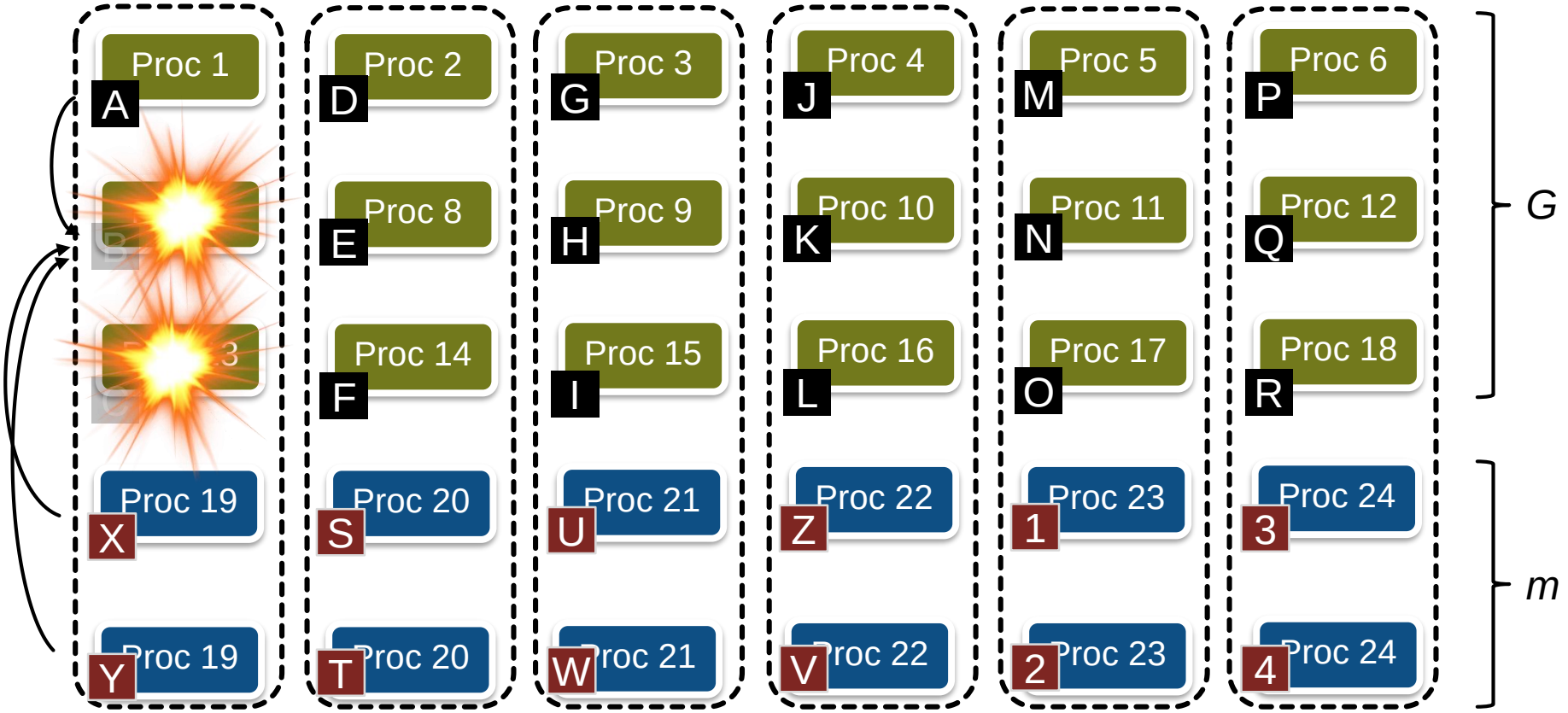
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



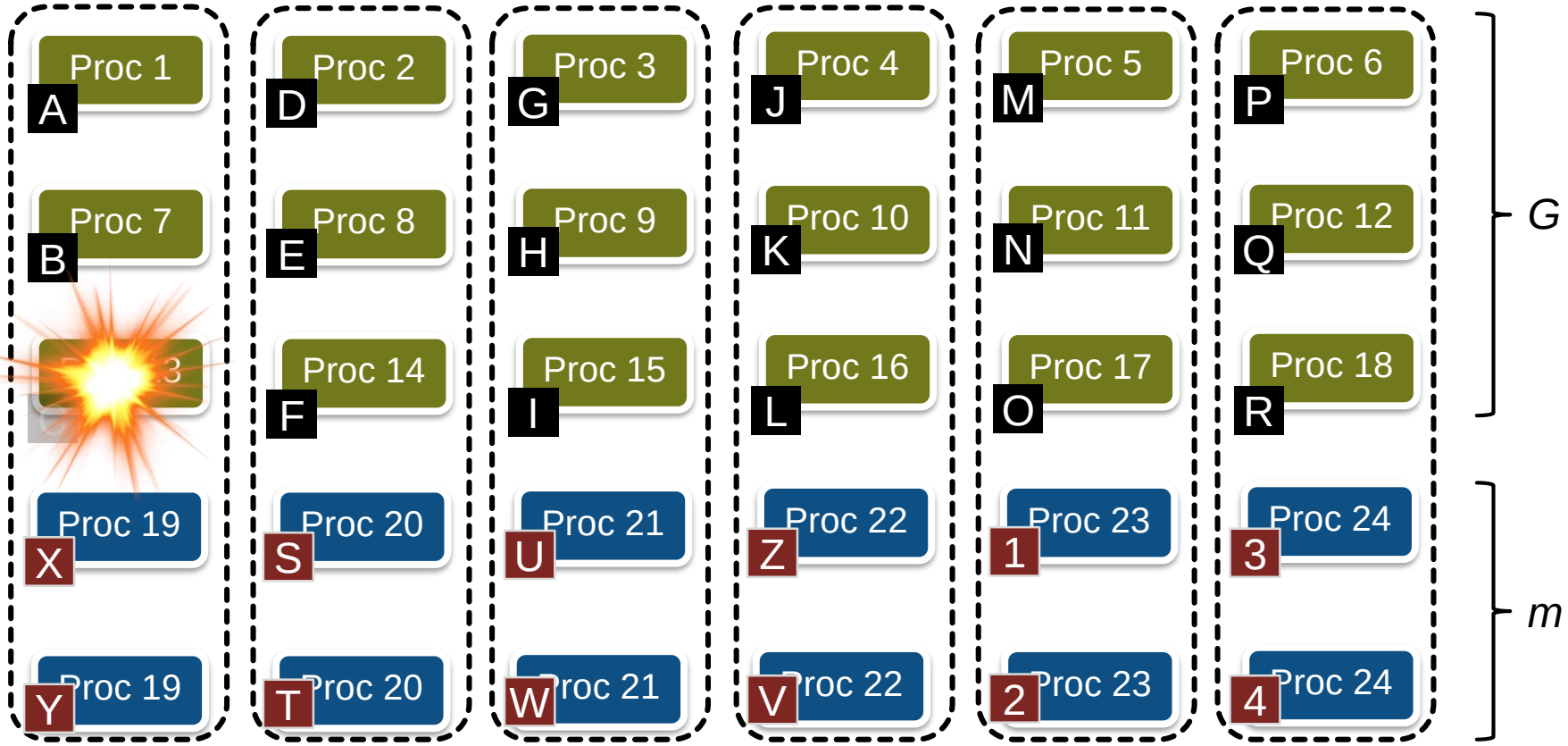
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



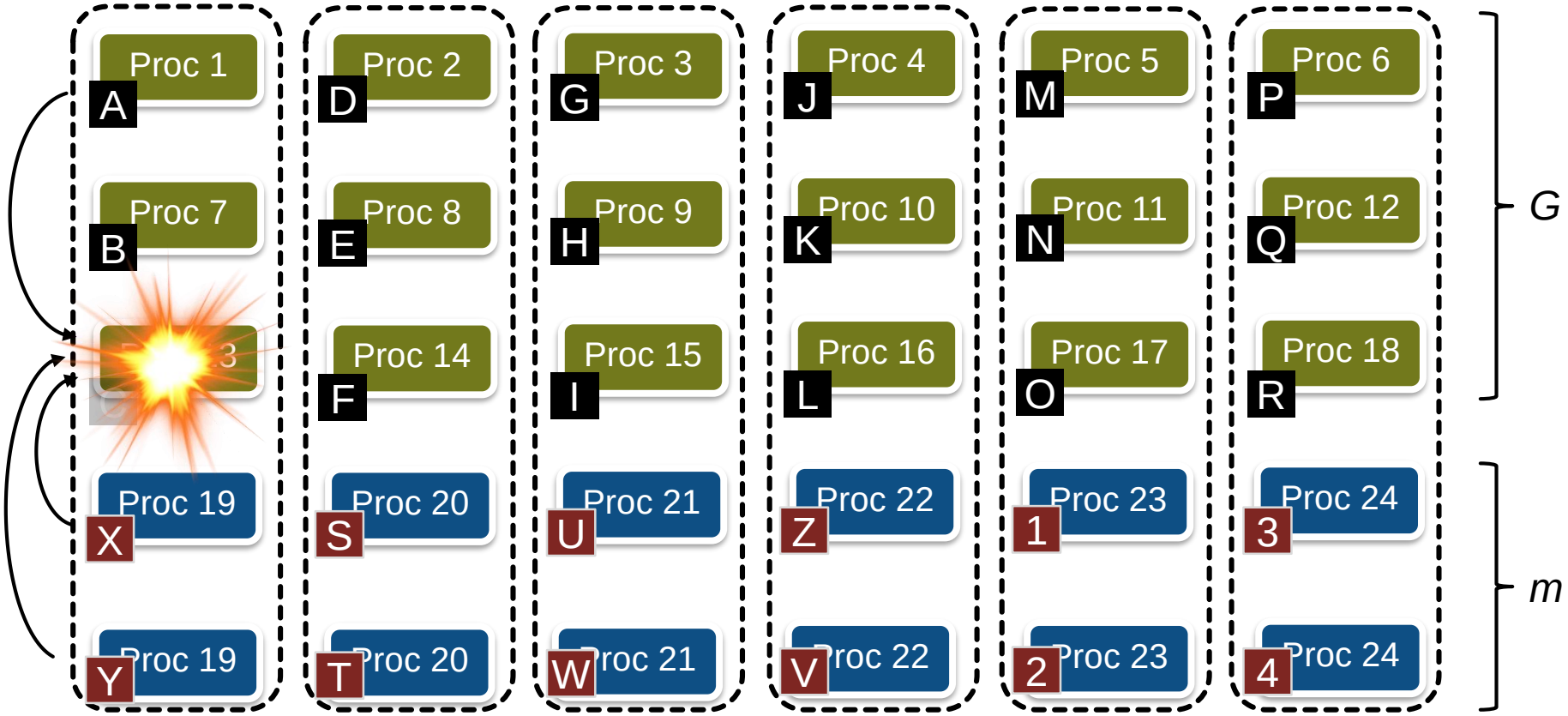
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



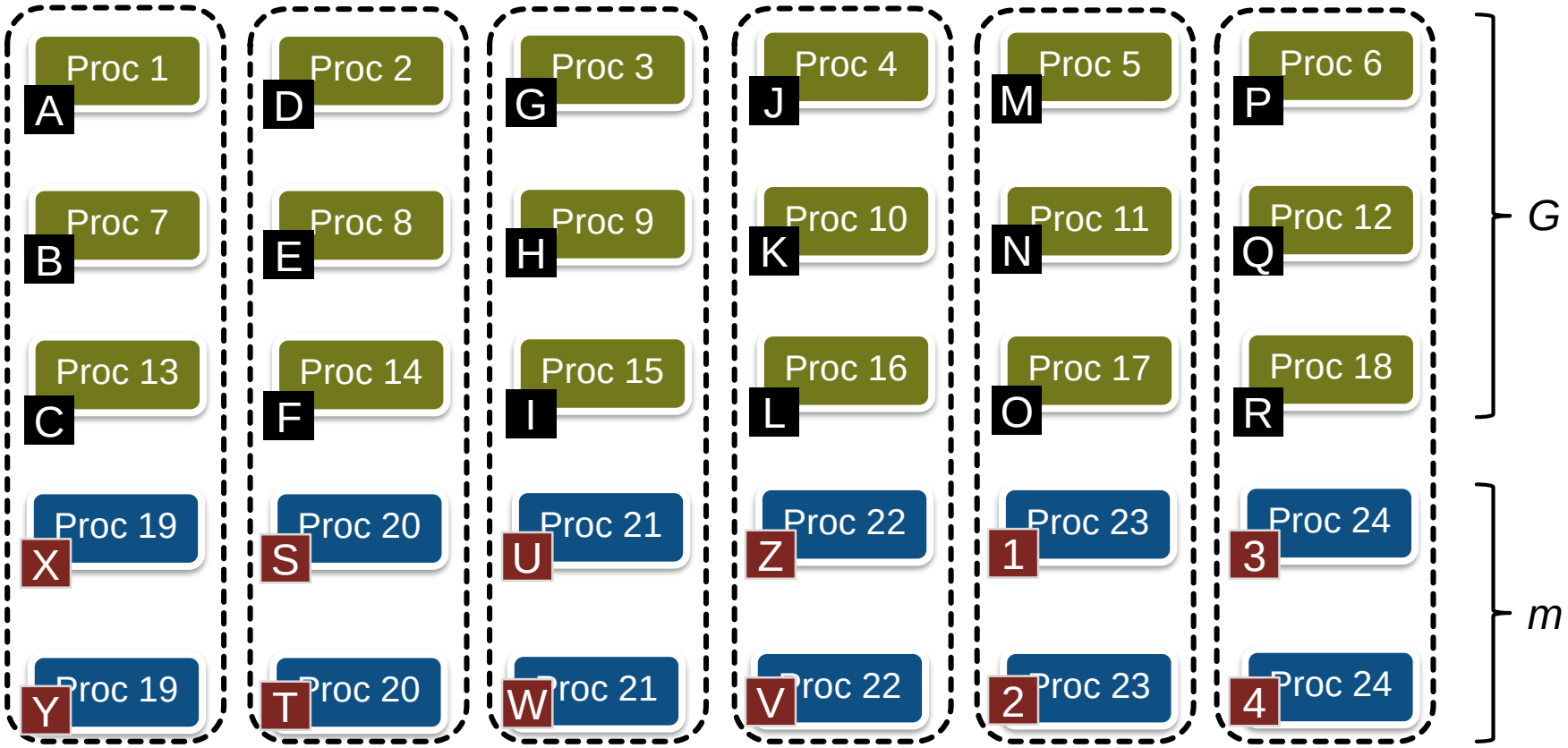
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



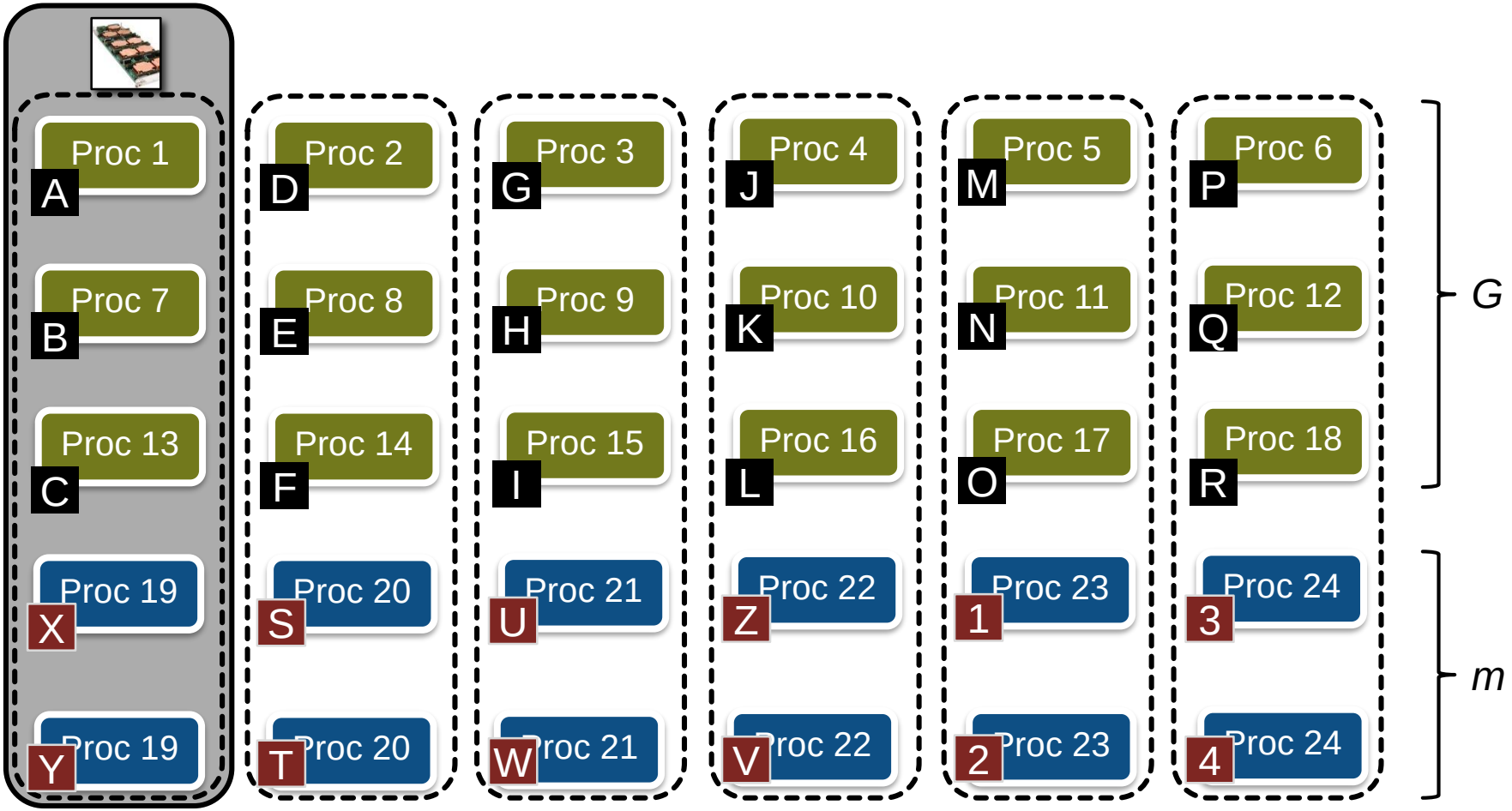
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



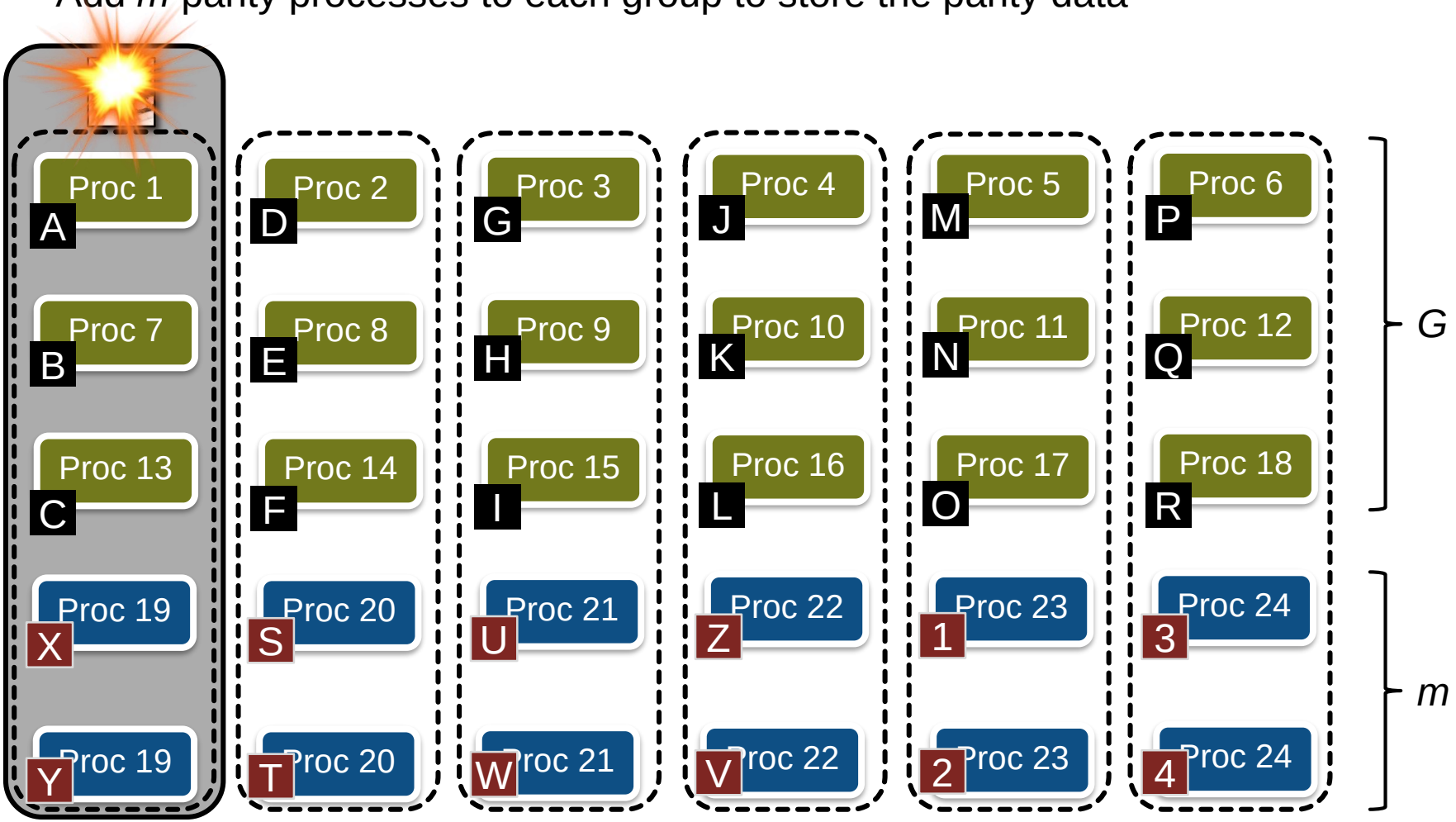
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



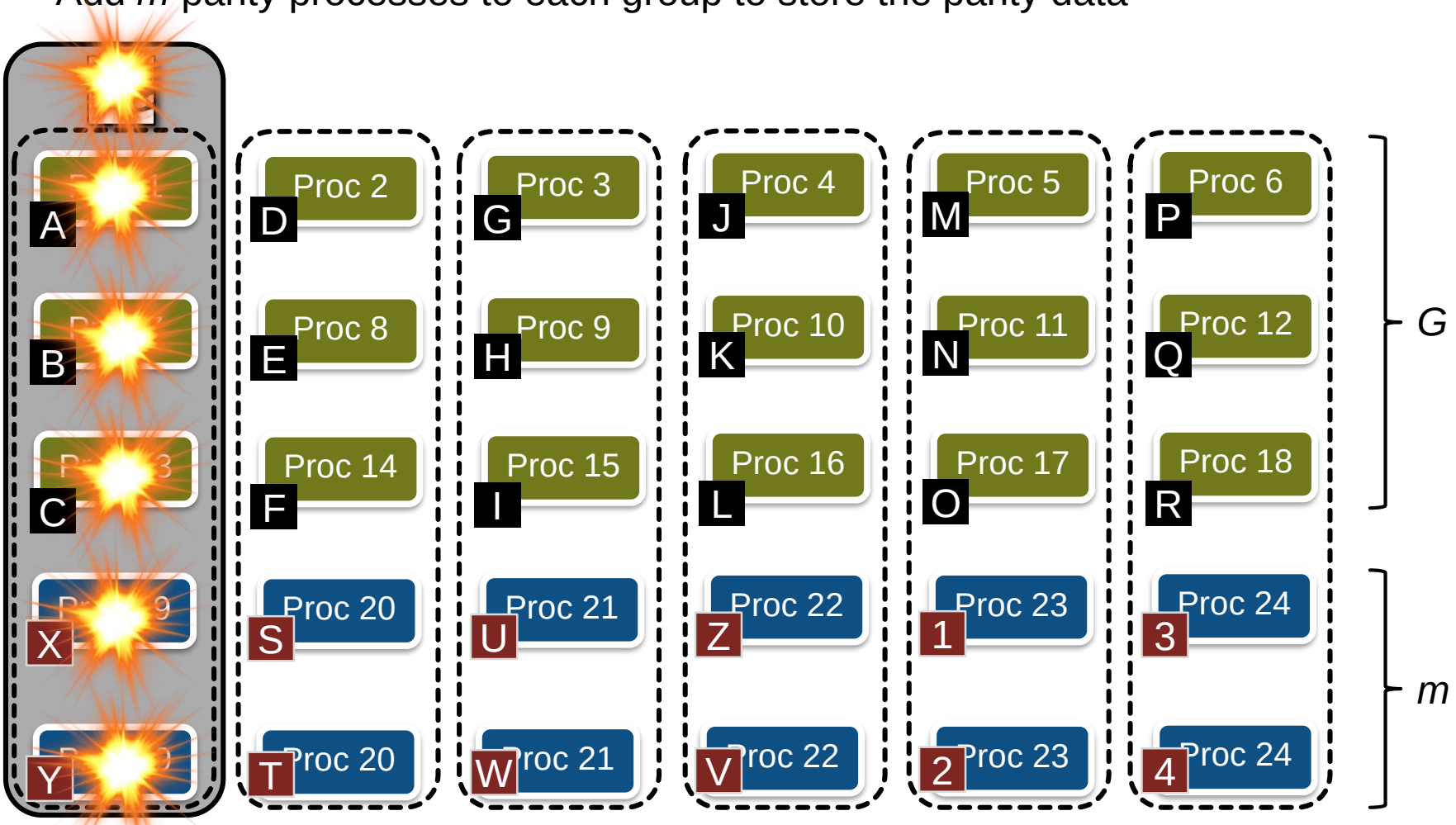
Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data

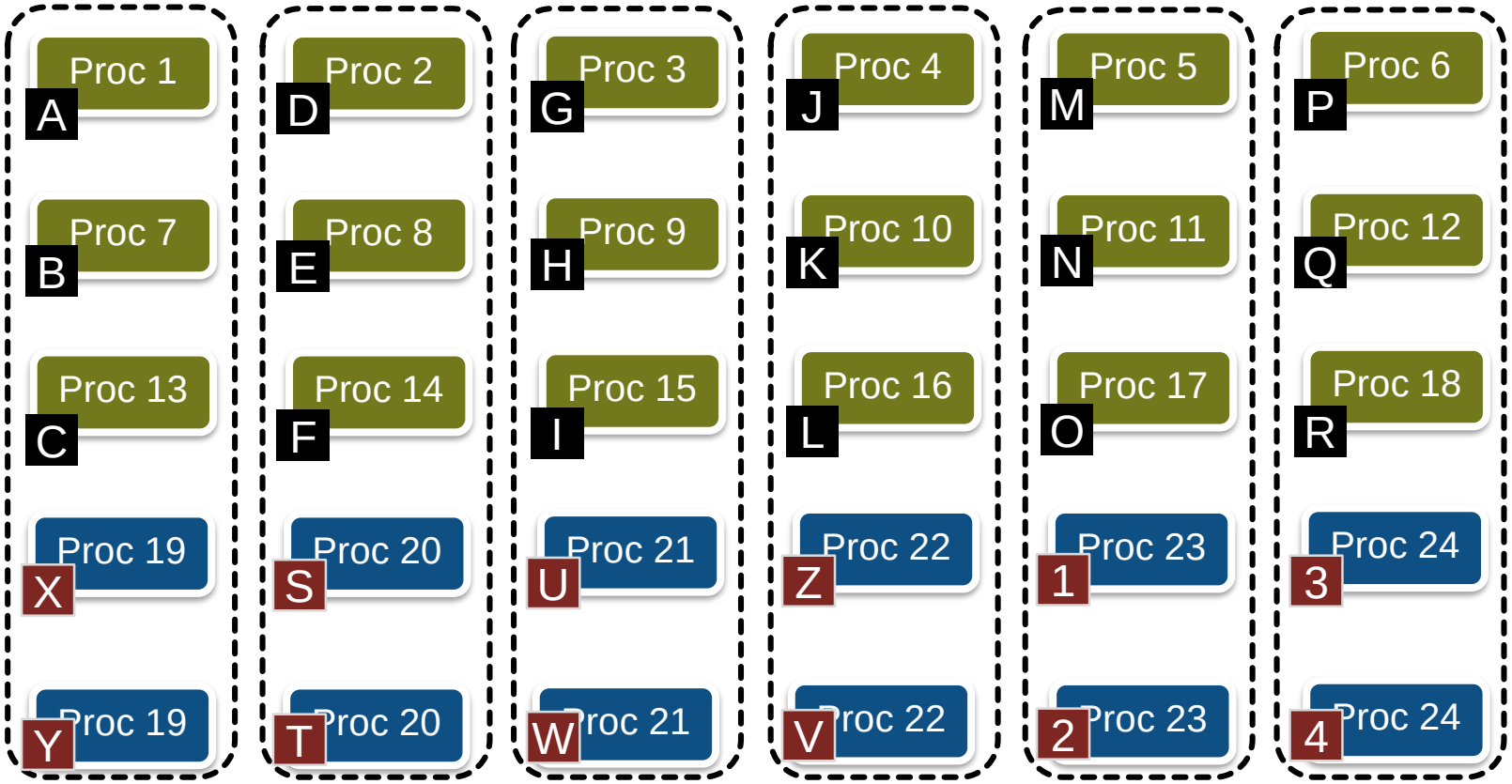


Extending the Protocols for more Resilience

- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data

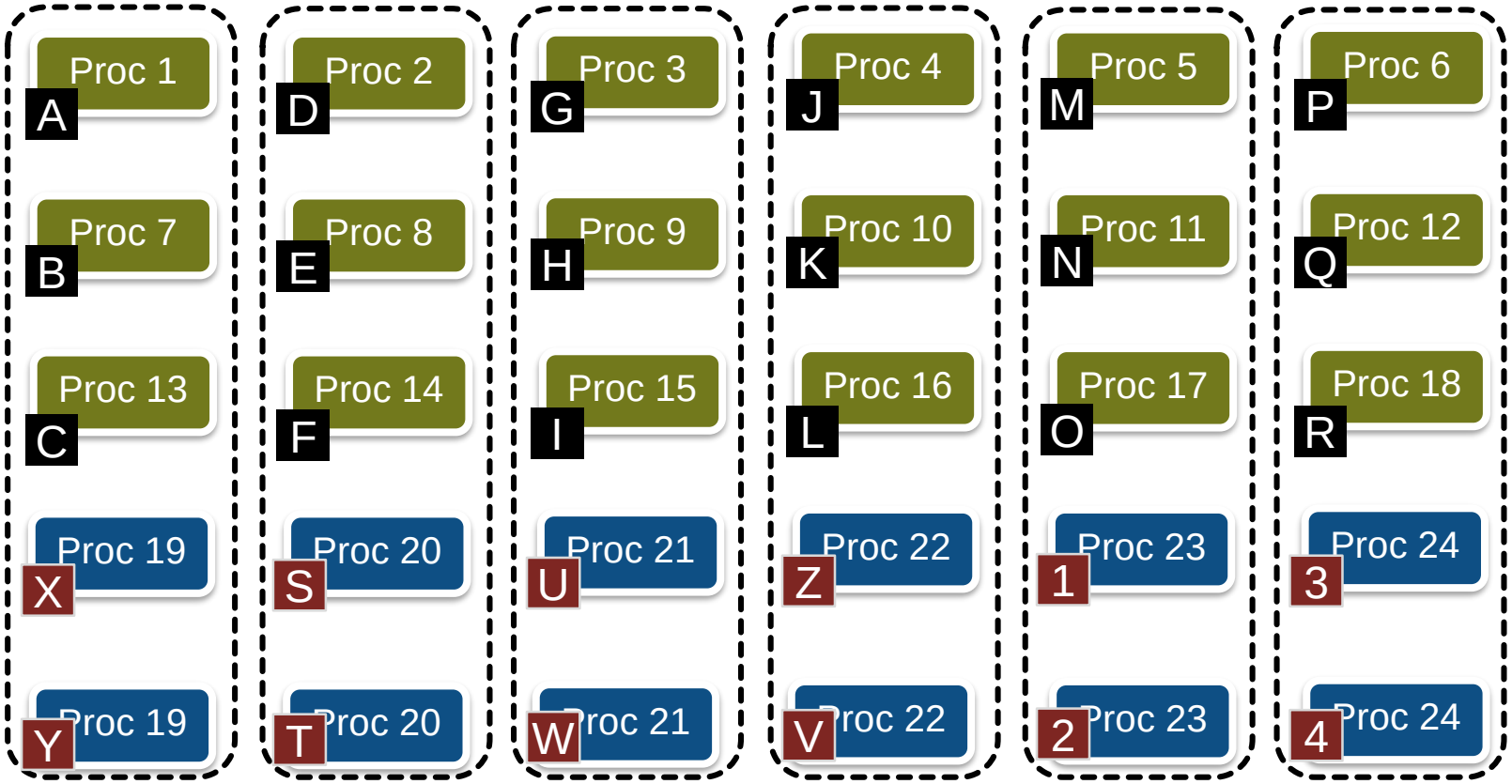


Extending the Protocols for more Resilience



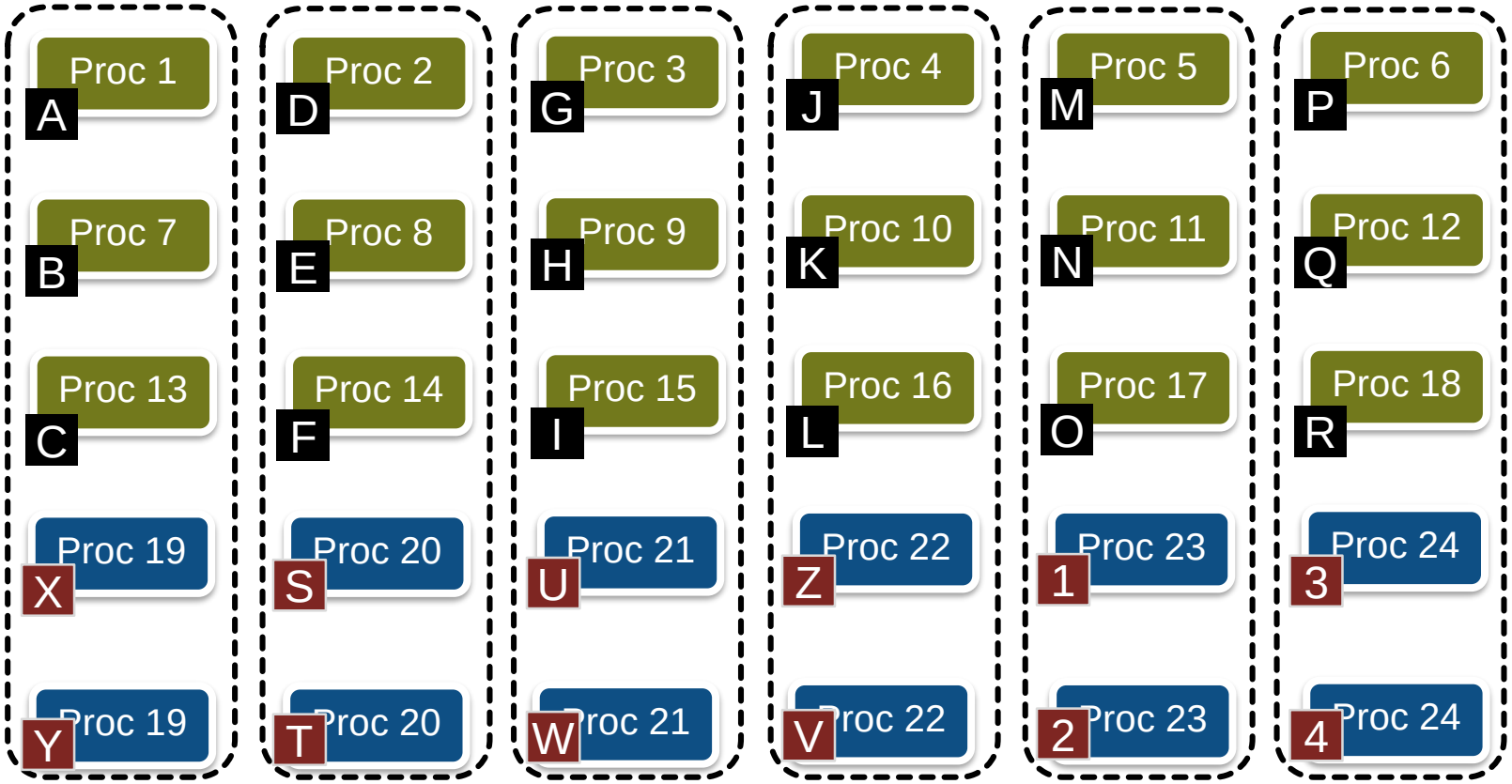
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:



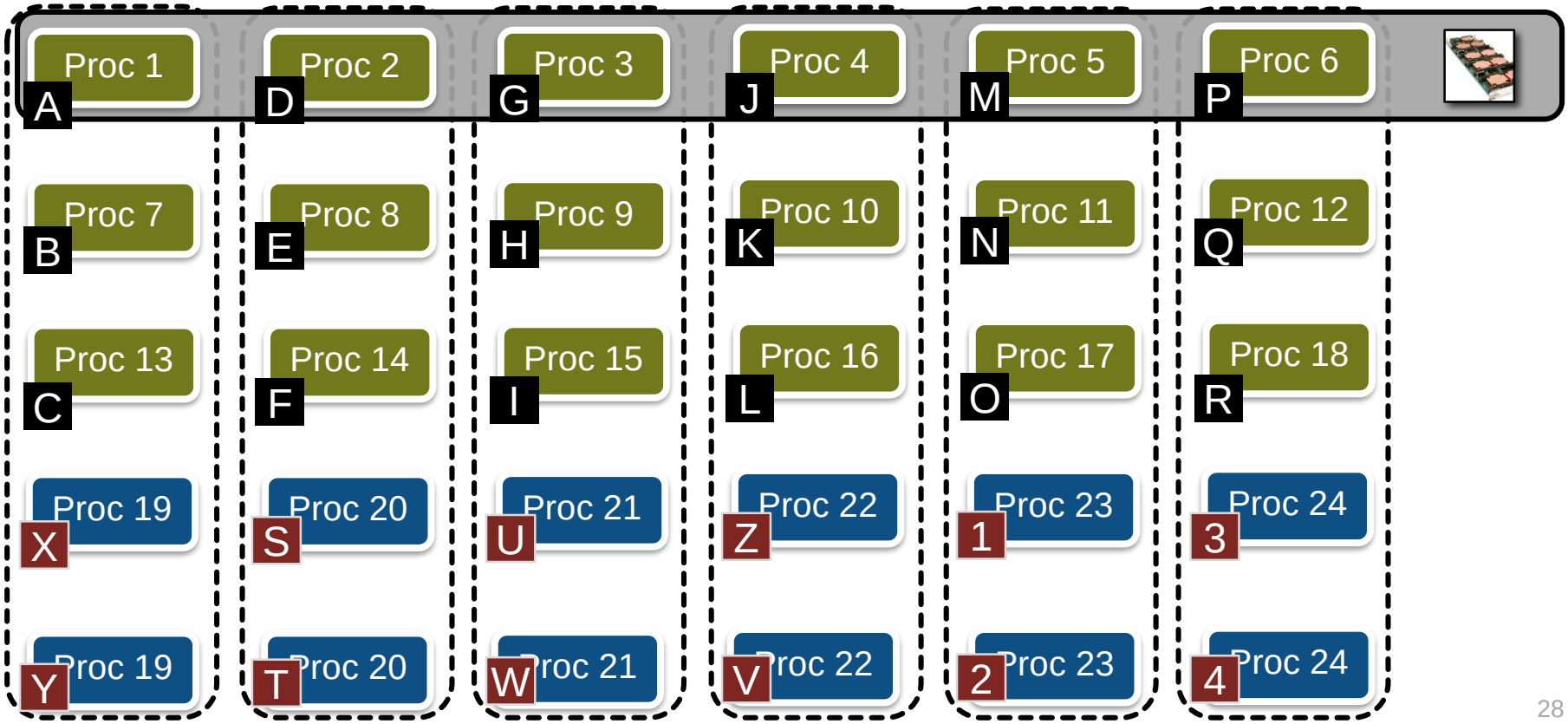
Extending the Protocols for more Resilience

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...



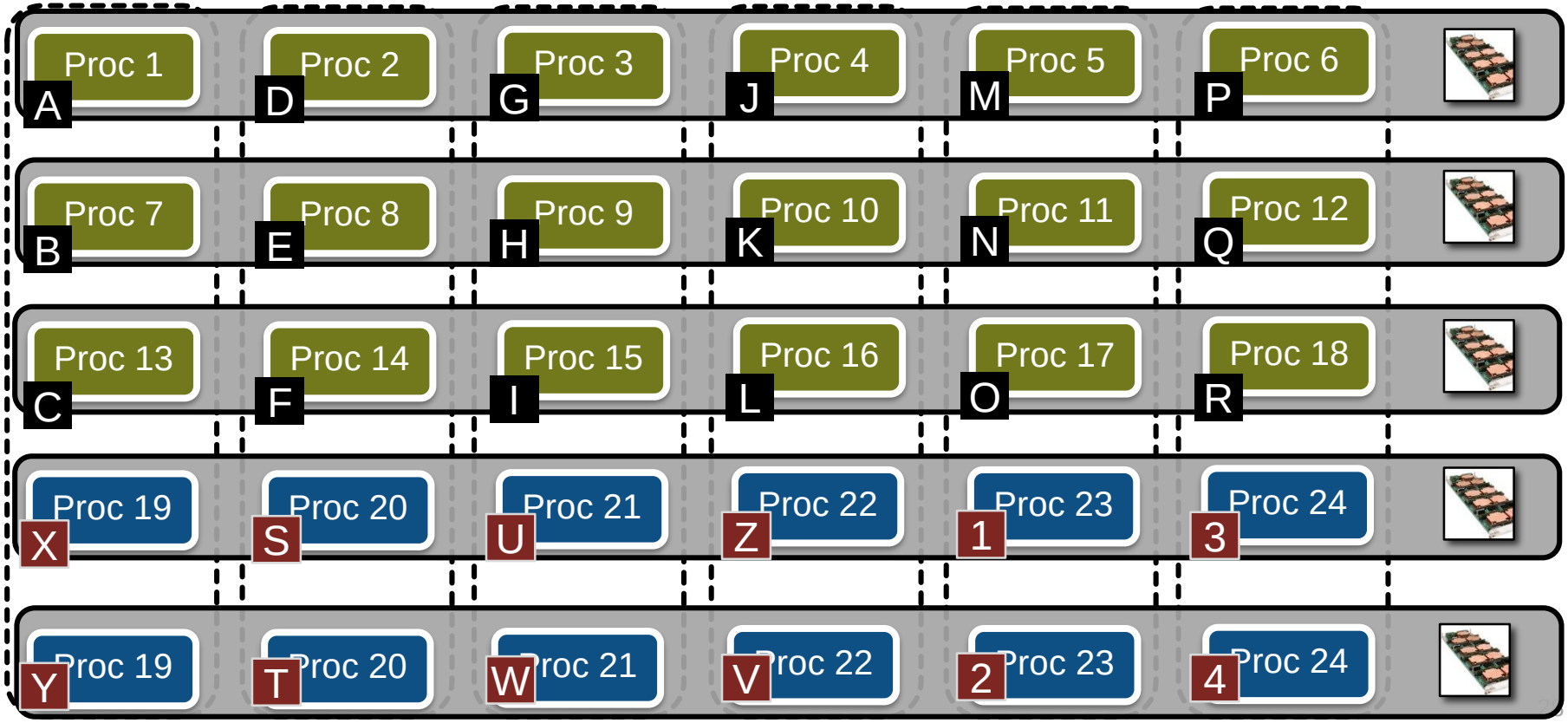
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...



Extending the Protocols for more Resilience

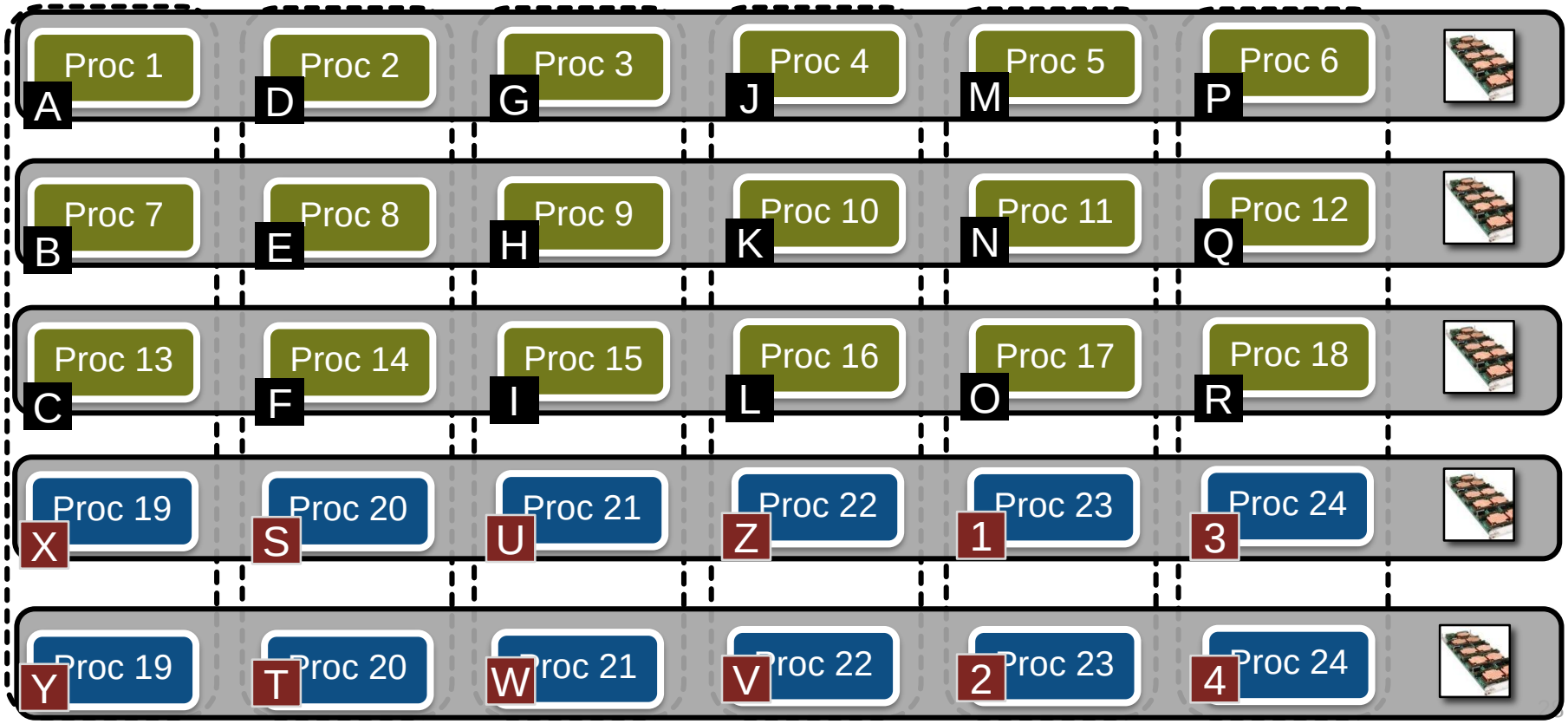
- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...



Extending the Protocols for more Resilience

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

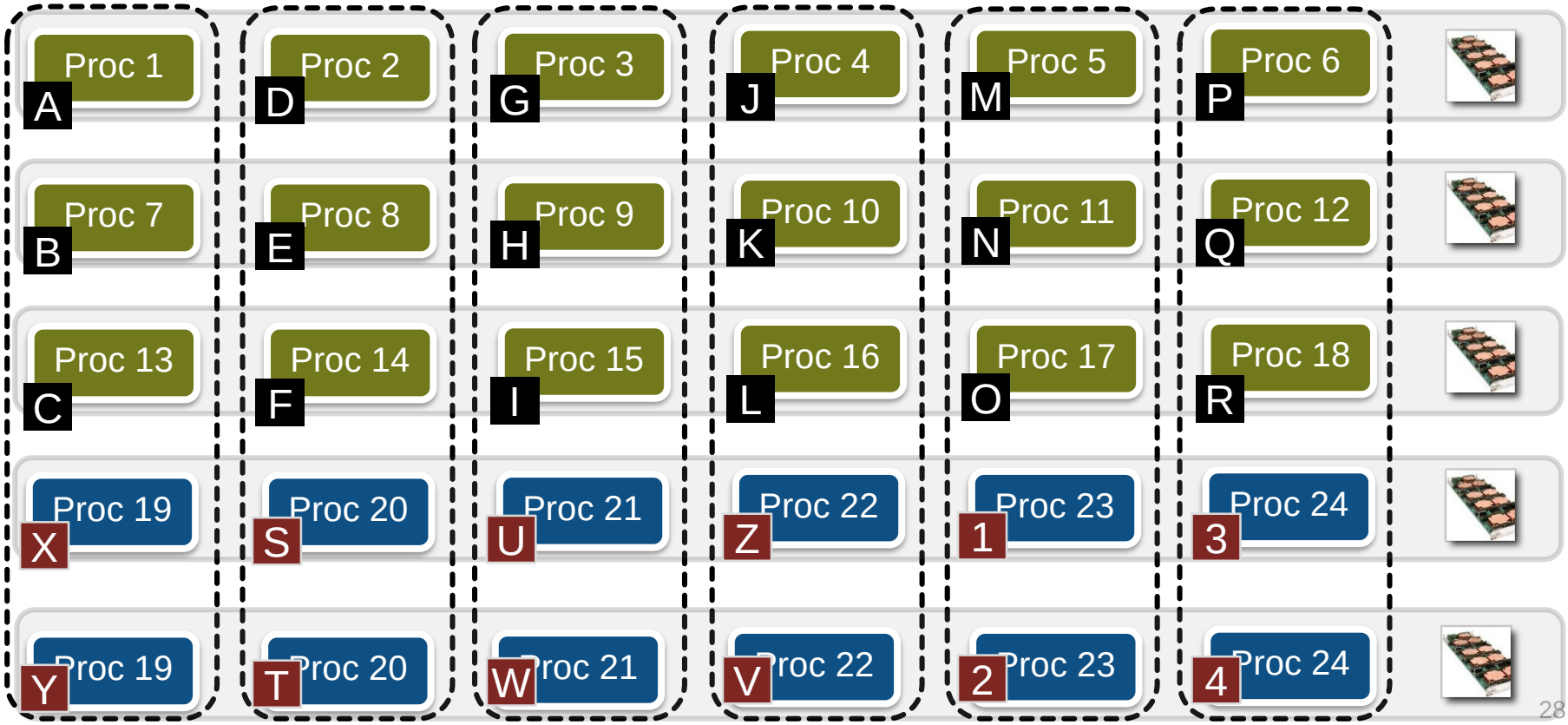
... and nodes



Extending the Protocols for more Resilience

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

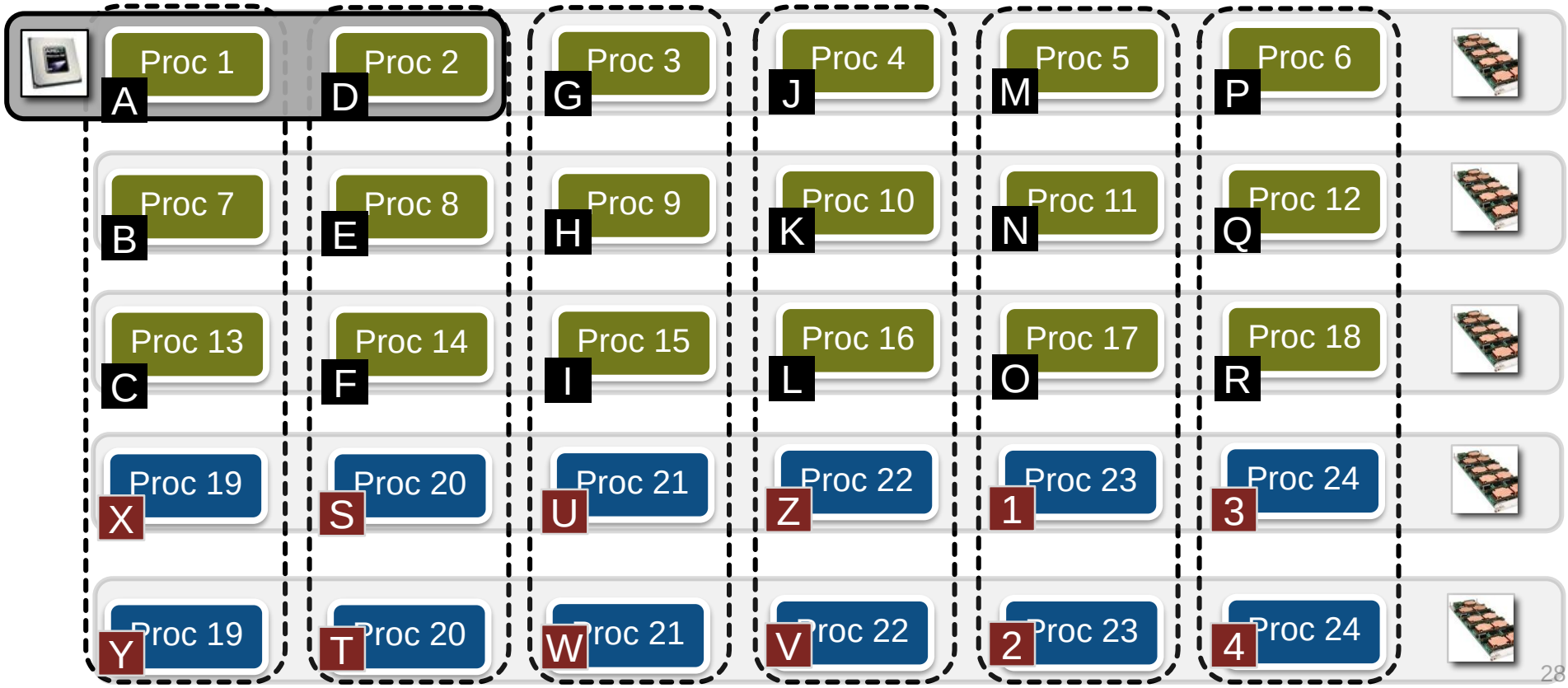
... and nodes



Extending the Protocols for more Resilience

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

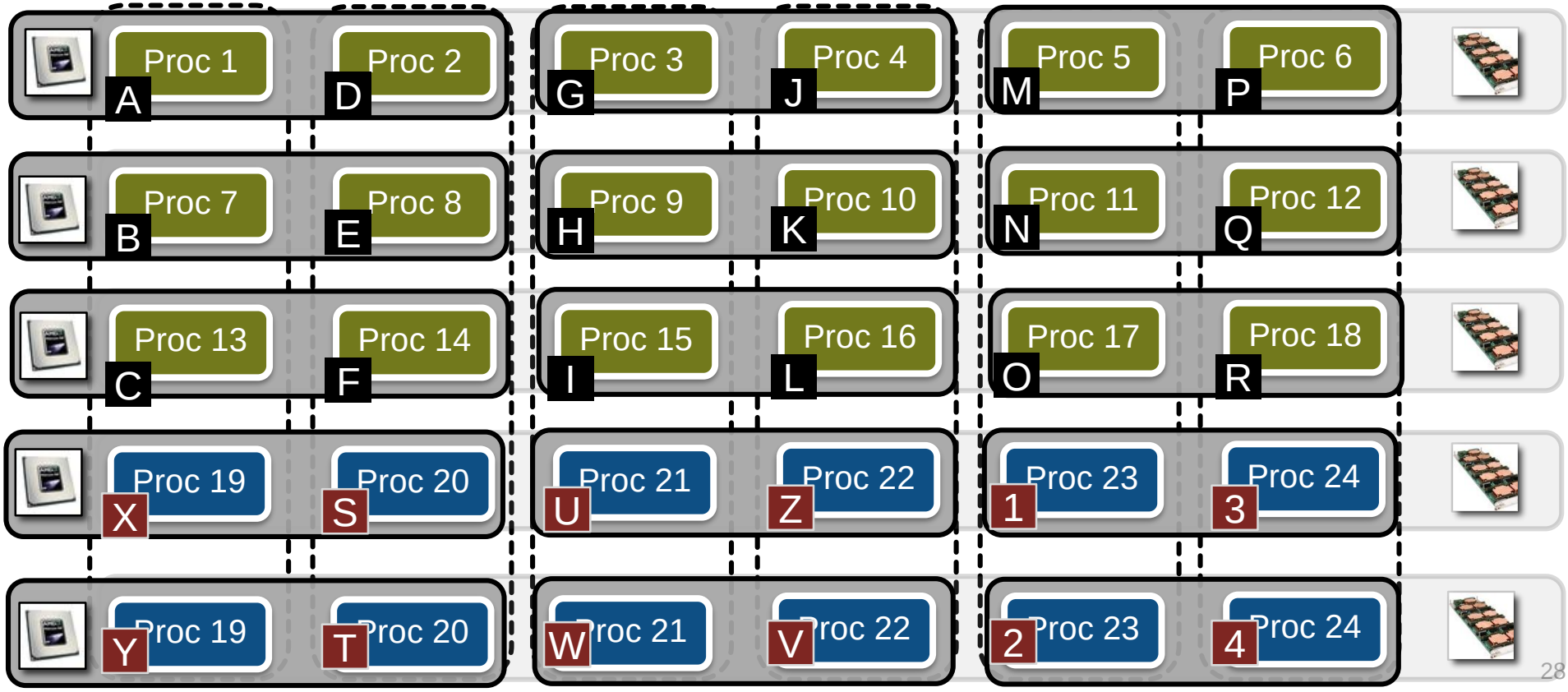
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

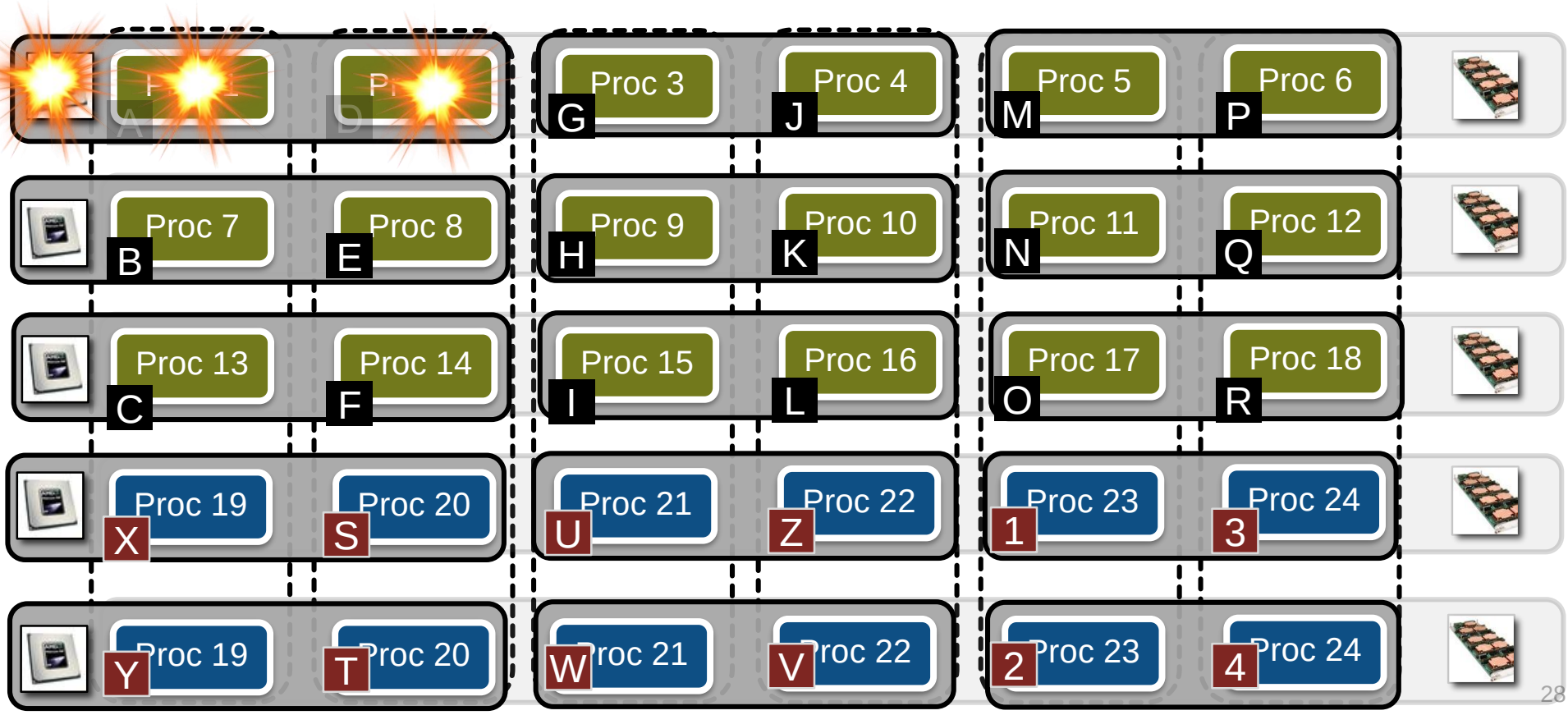
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

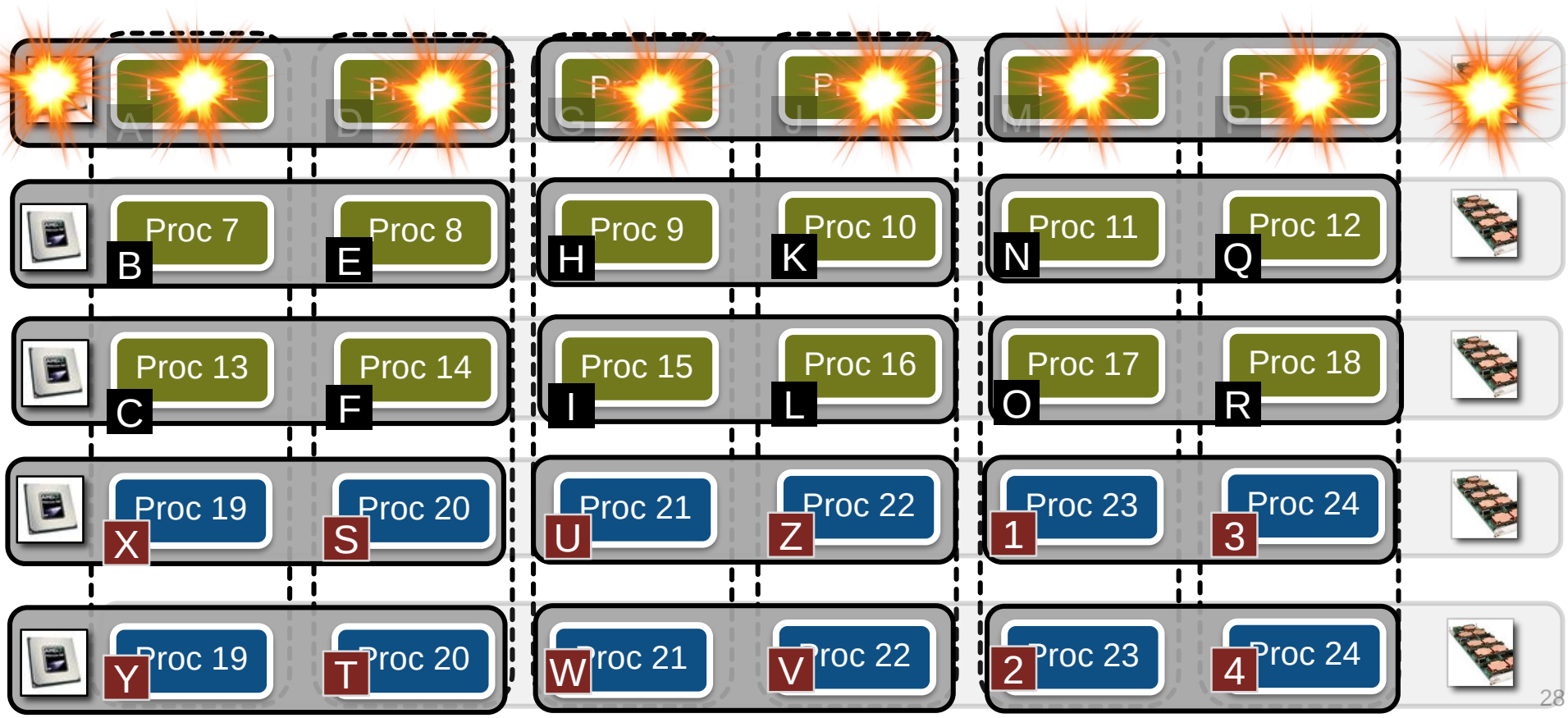
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

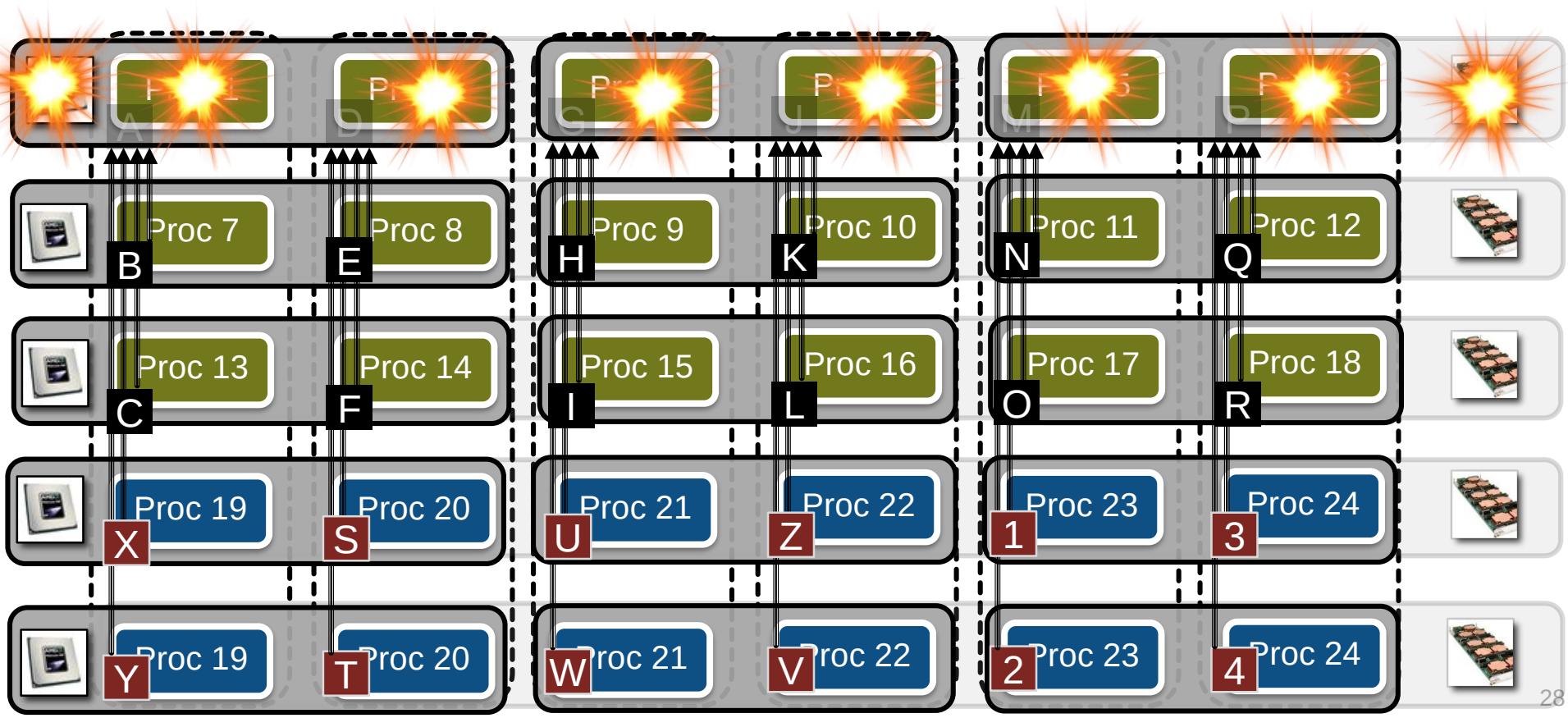
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

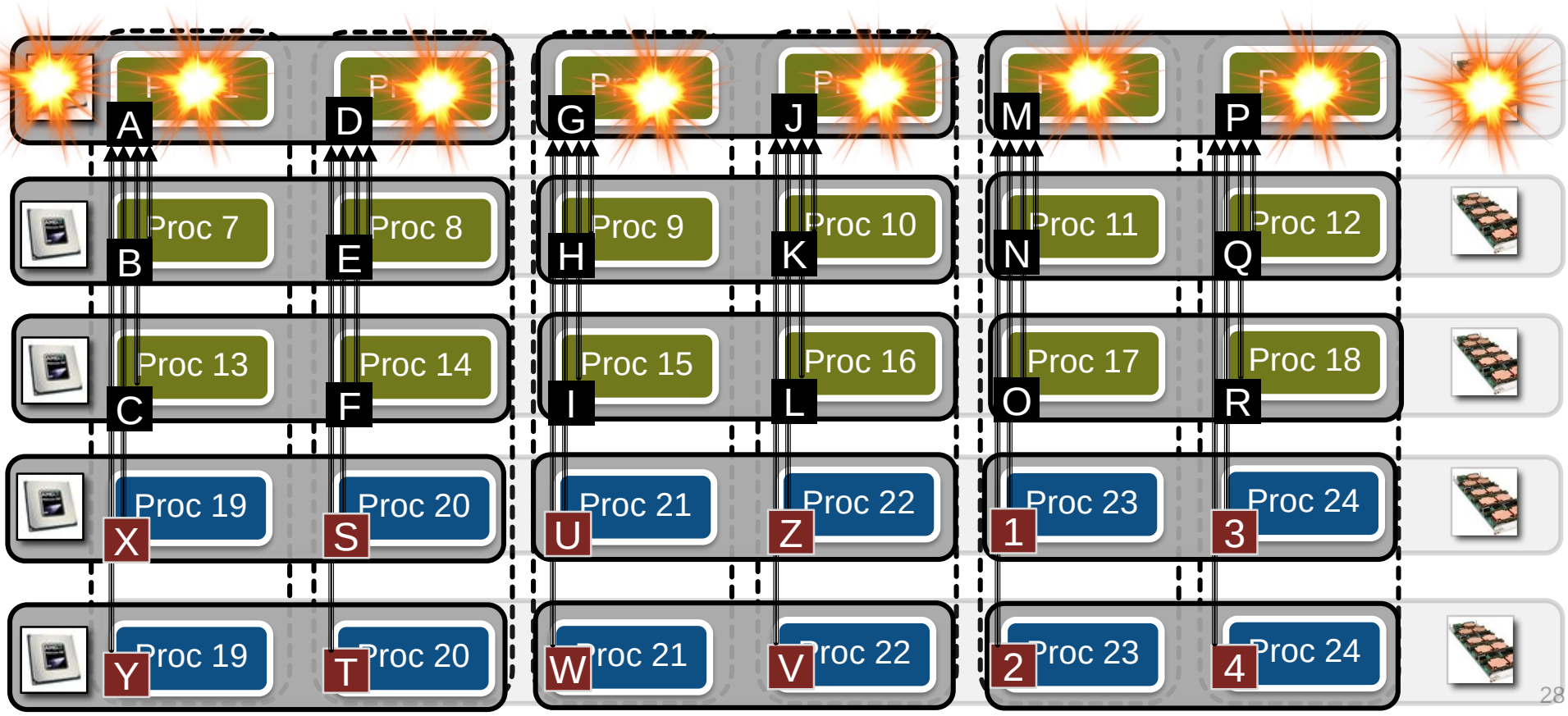
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

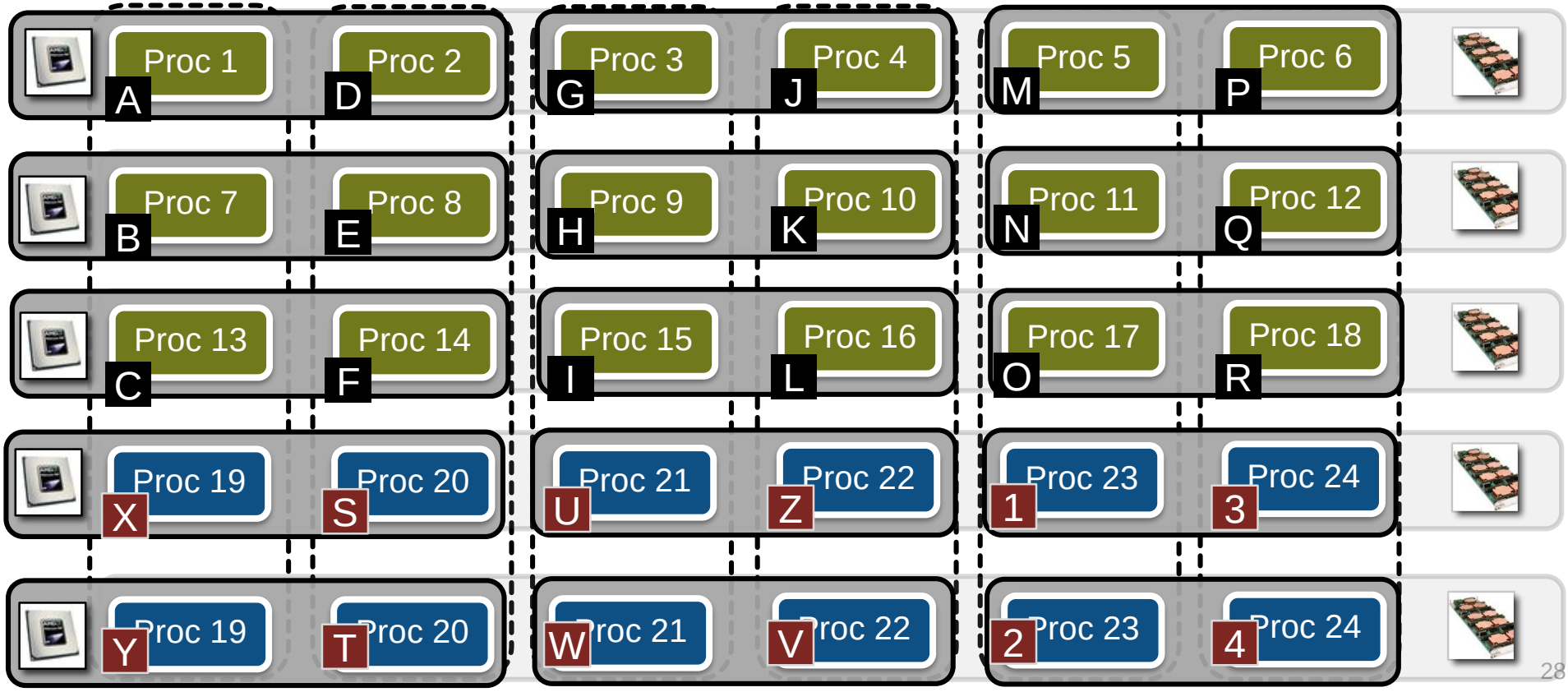
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

... and nodes



Extending the Protocols for more Resilience

Extending the Protocols for more Resilience

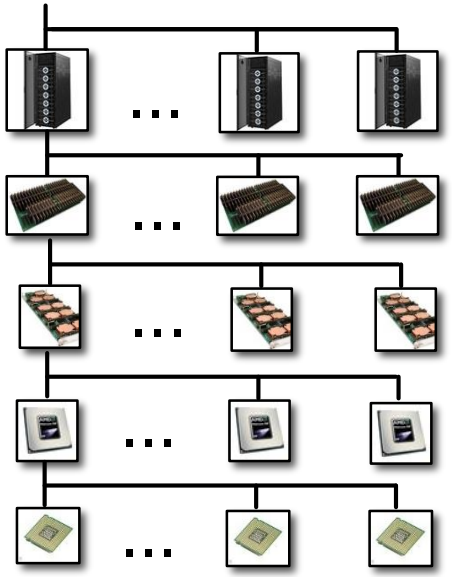
- The probability of a catastrophic failure in a multi-level computing machine

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

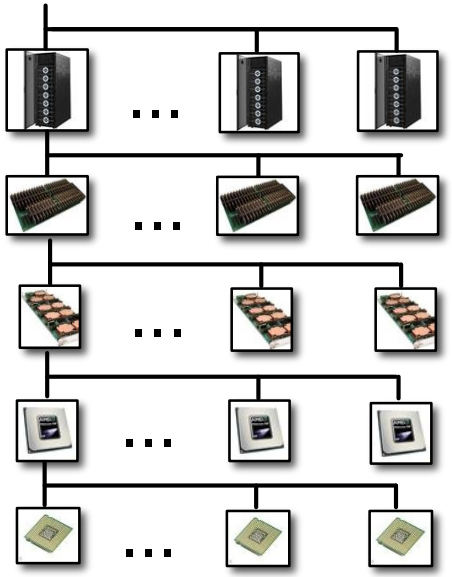
Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



Extending the Protocols for more Resilience

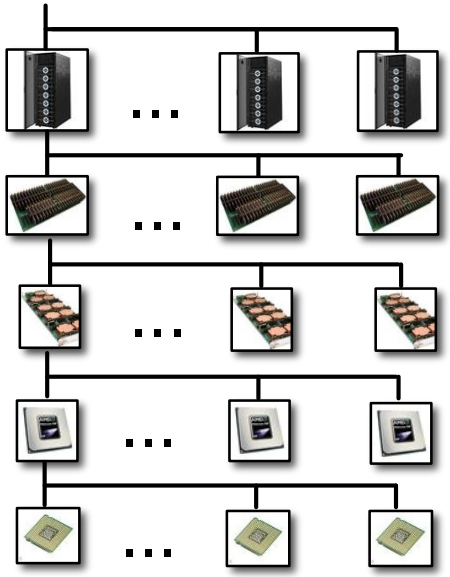
- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$P_{cf} =$ $P_j(x_j \cap x_{j,cf}) =$

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

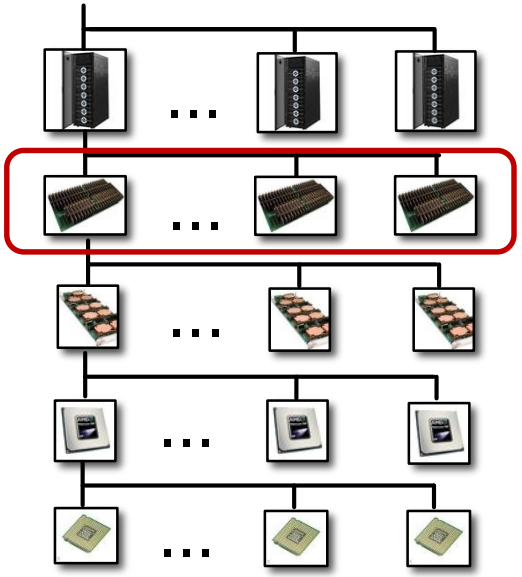


$$P_{cf} = P_j(x_j \cap x_{j,cf}) =$$

Probability that x_j elements of level j will fail and cause a catastrophic failure

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

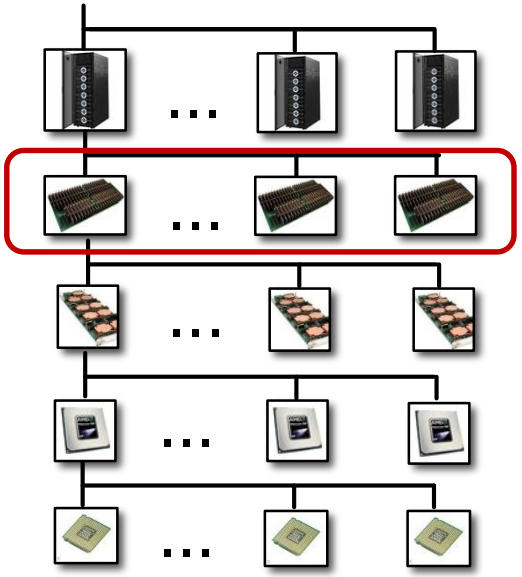


$$P_{cf} = P_j(x_j \cap x_{j,cf}) =$$

Probability that x_j elements of level j will fail and cause a catastrophic failure

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

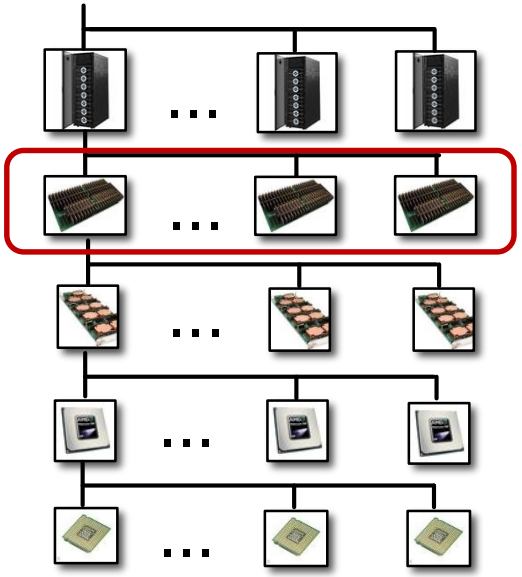


$$P_{cf} = P_j(x_j \cap x_{j,cf}) = P_j(x_j)\ P_j(x_{j,cf}|x_j)$$

Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



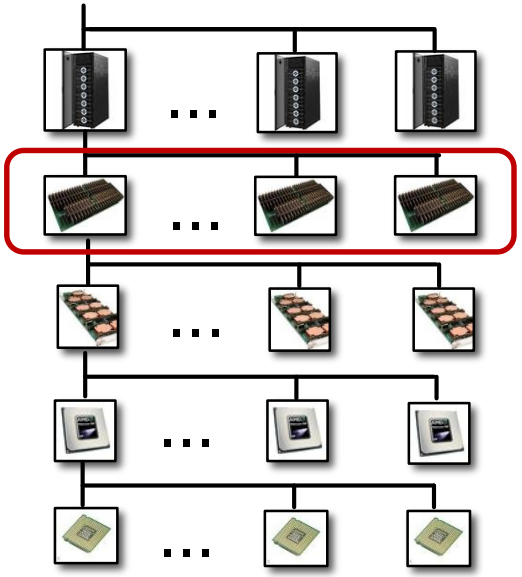
$$P_{cf} = P_j(x_j \cap x_{j,cf}) = P_j(x_j)\, P_j(x_{j,cf}|x_j)$$

Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = P_j(x_j \cap x_{j,cf}) = P_j(x_j)\,P_j(x_{j,cf}|x_j)$$

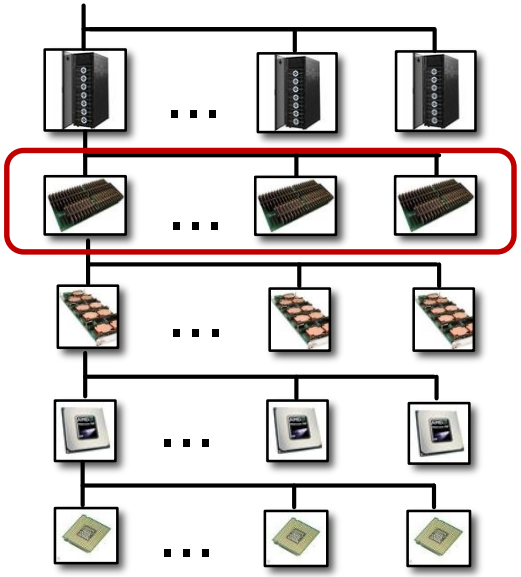
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

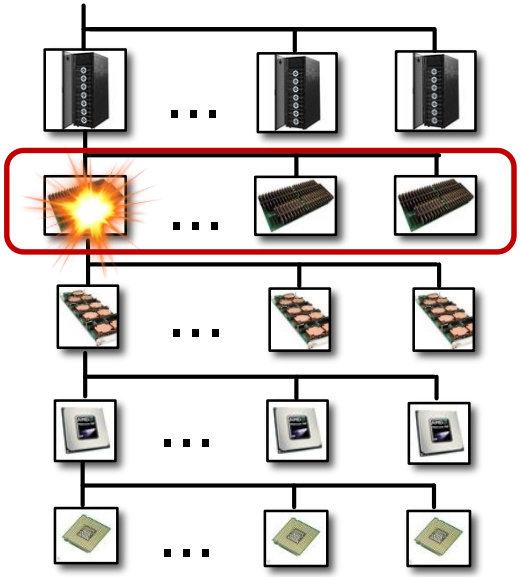
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

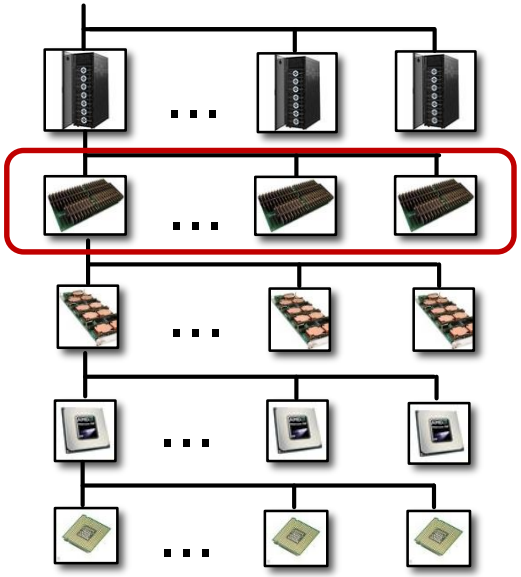
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

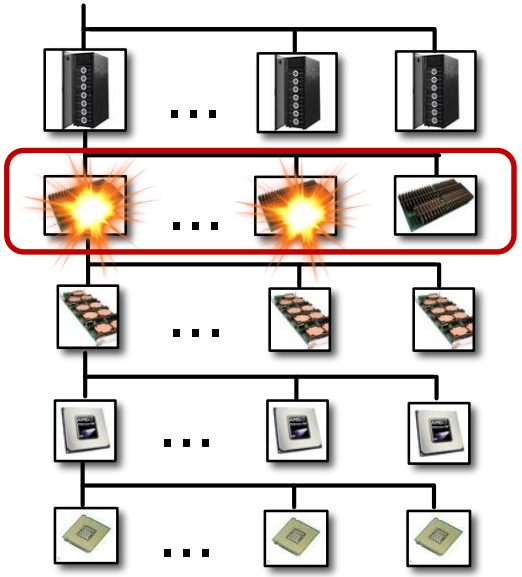
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

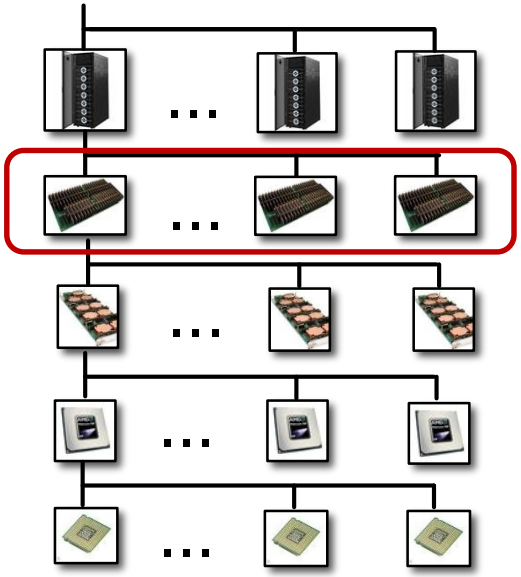
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

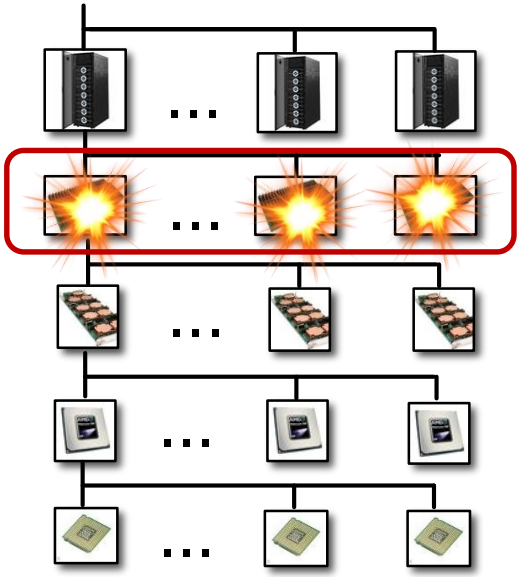
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

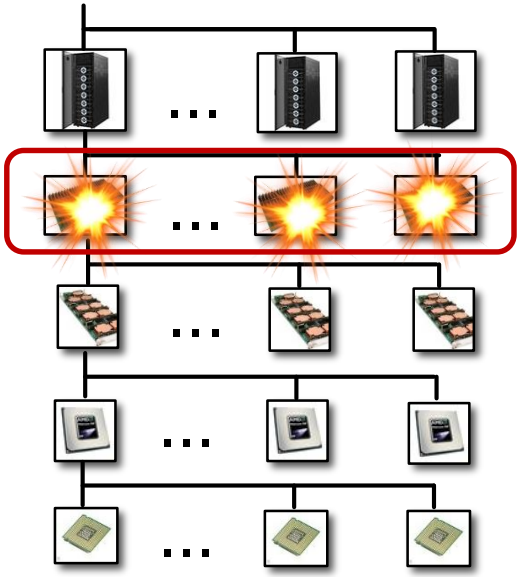
Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

...at every hierarchy level

Every number of x_j elements is considered...

$$P_{cf} = \sum_{j=1}^{h} \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{j=1}^{h} \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$



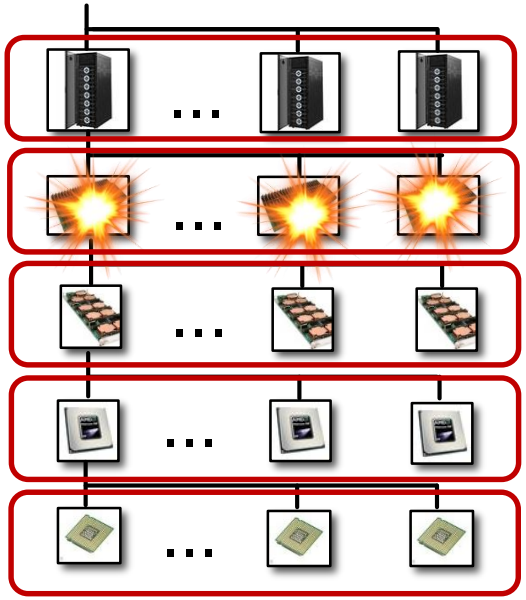
Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die



...at every hierarchy level

Every number of x_j elements is considered...

$$P_{cf} = \sum_{j=1}^{h} \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{j=1}^{h} \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf}|x_j)$$

Probability that x_j elements of level *j* will fail and cause a catastrophic failure

Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

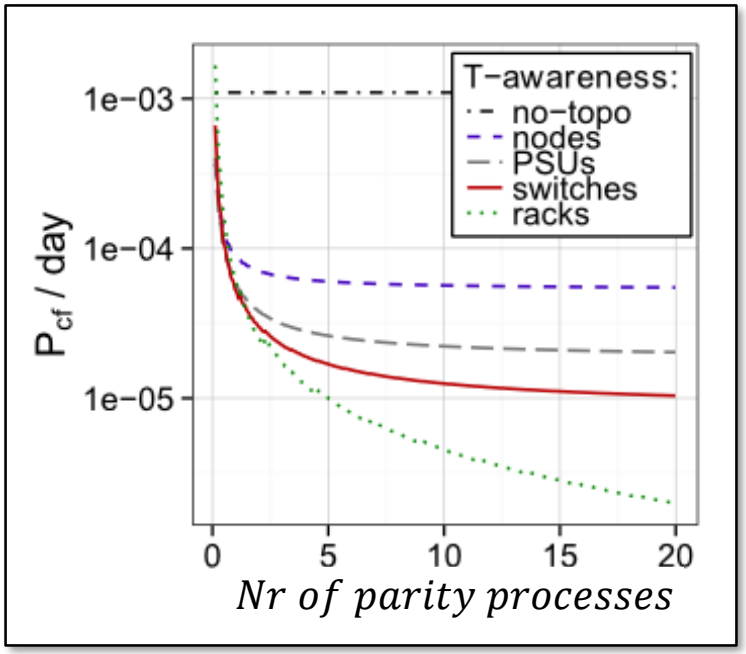
Extending the Protocols for more Resilience

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
  - A catastrophic failure: a failure that takes place when more than *m* processes in the same group die

P_{cf}/day:



Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

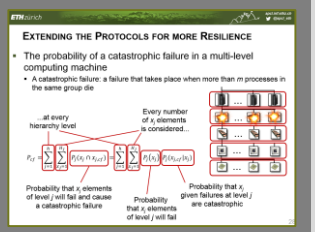
Performance

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes



Decreasing failure prob.

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

MP vs. RMA

Schemes

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Deadlock freedom
Correct recovery

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

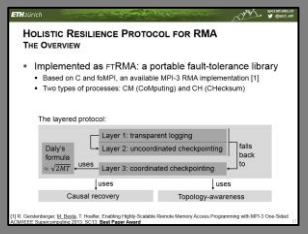
$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

Performance

Holistic Resilience Protocol for RMA

The Overview

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

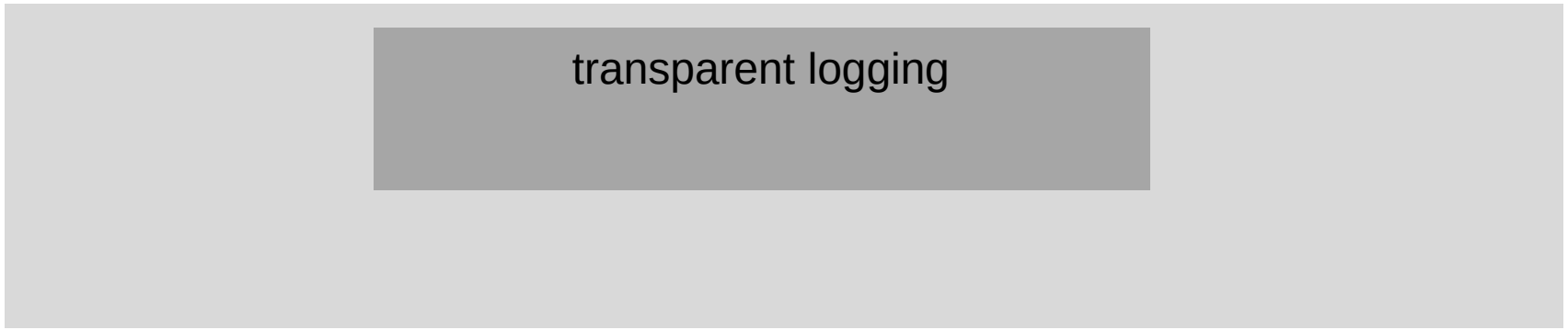
The layered protocol:

[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

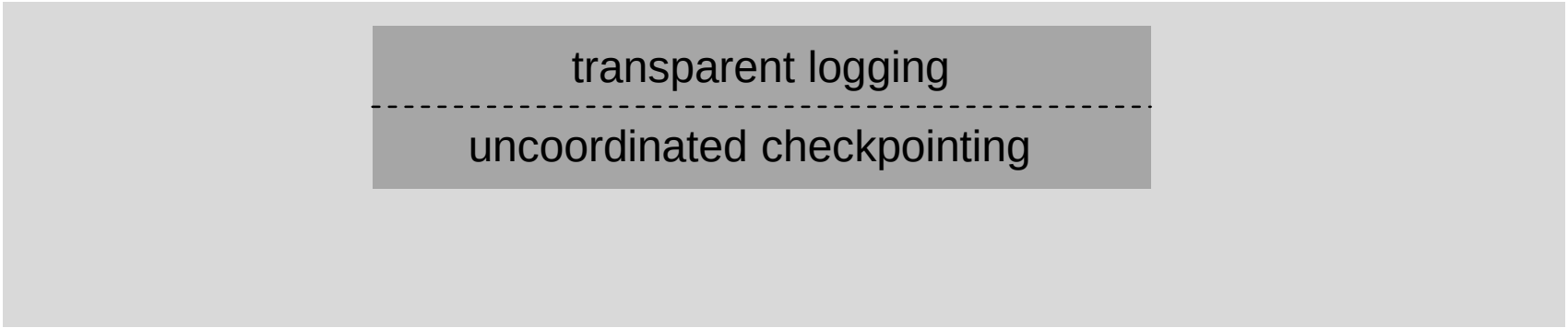


[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

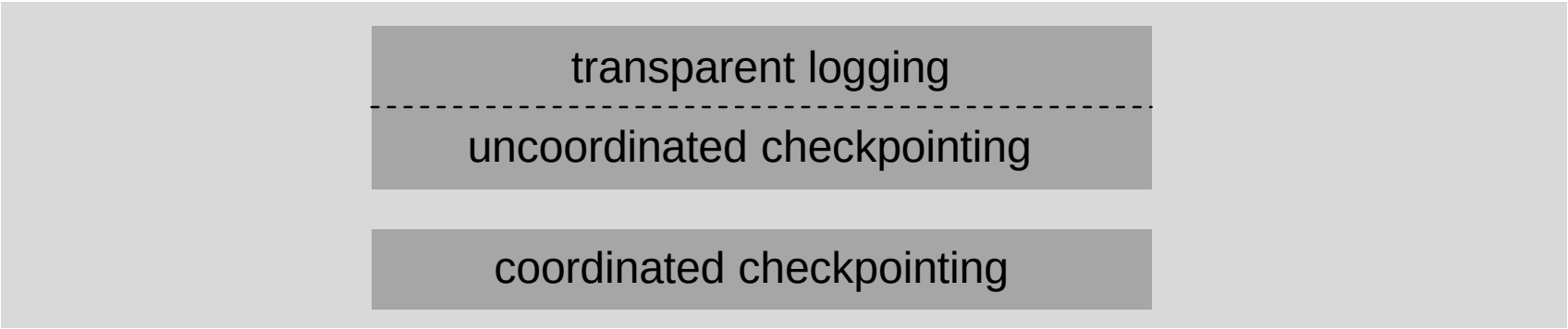


[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

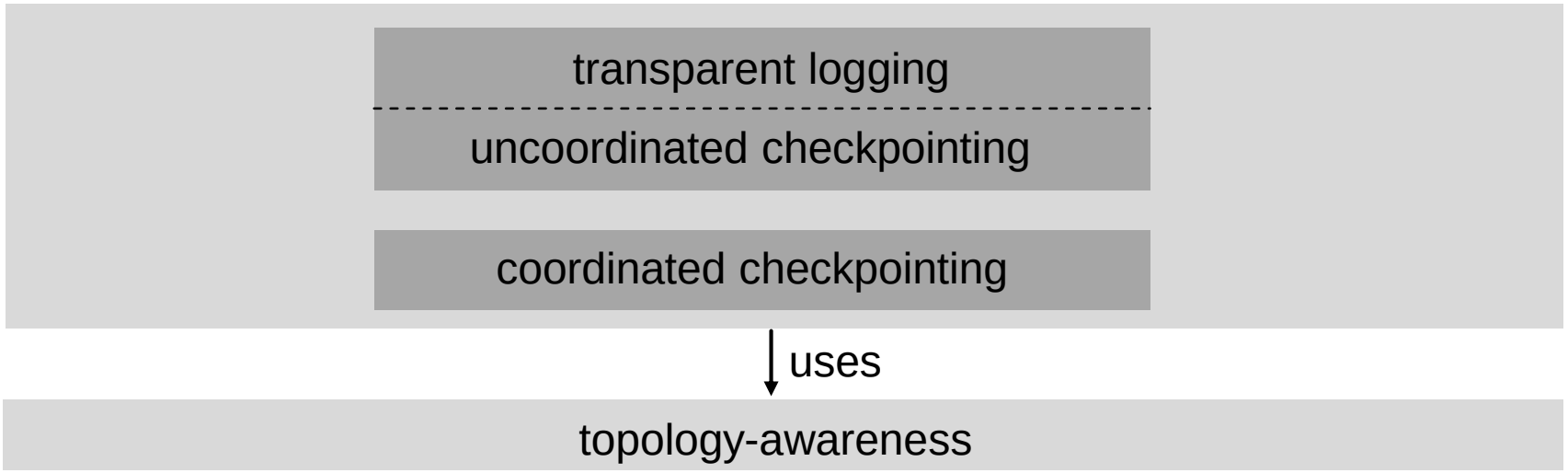


[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

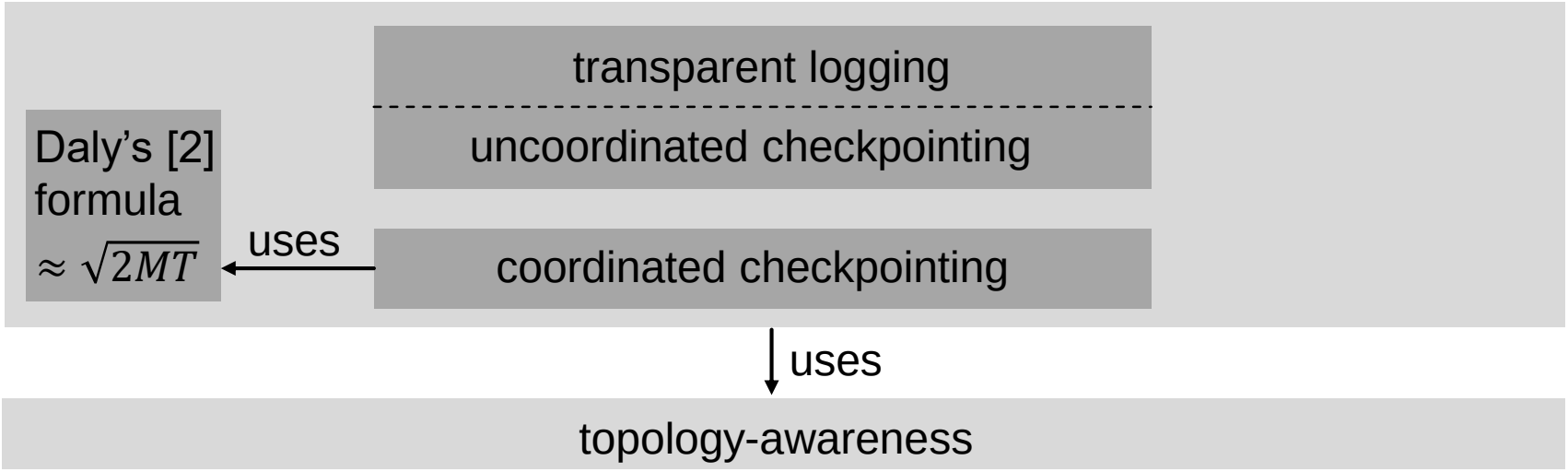


[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:



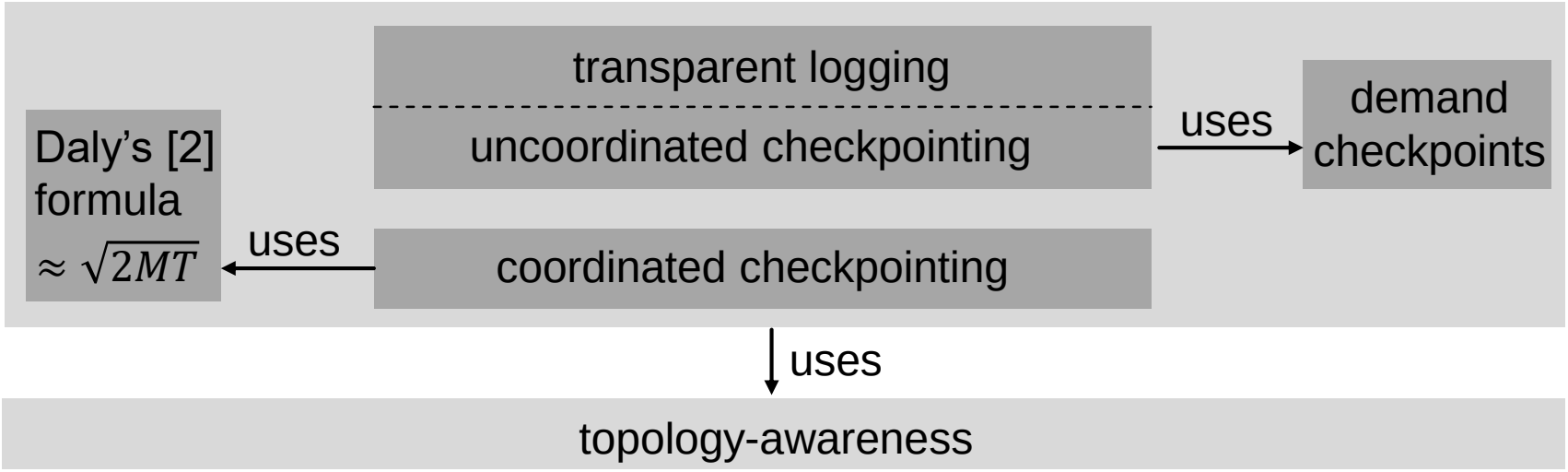
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

[2] J.T.Daly, A higher order estimate of the optimum checkpoint interval for restart dumps. Future Generation Computer Systems, Volume 22 Issue 3, February 2006, Pages 303-312

HOLISTIC RESILIENCE PROTOCOL FOR RMA
THE OVERVIEW

- Implemented as FTRMA: a portable fault-tolerance library
  - Based on C and foMPI, an available MPI-3 RMA implementation [1]

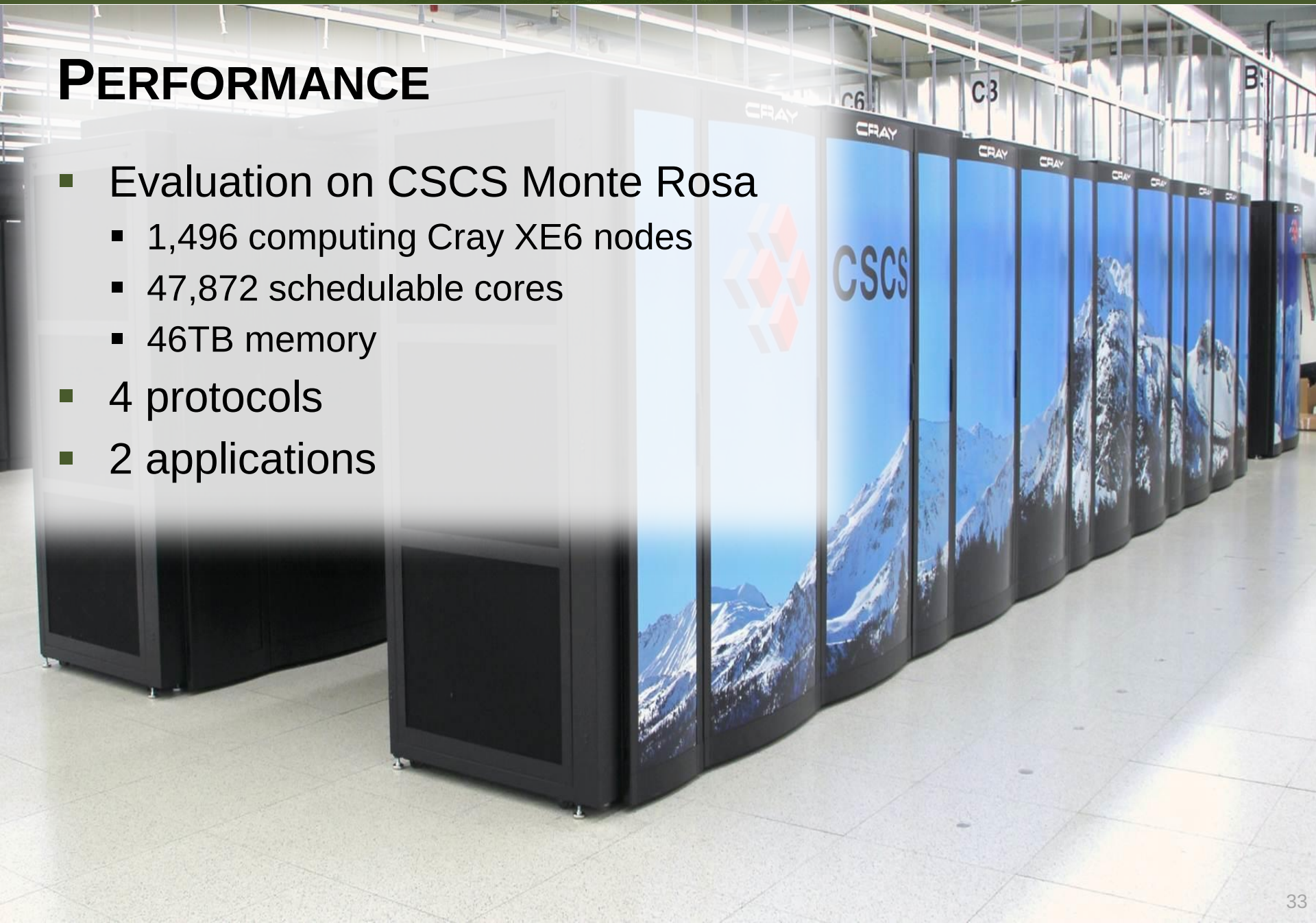
The layered protocol:



[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

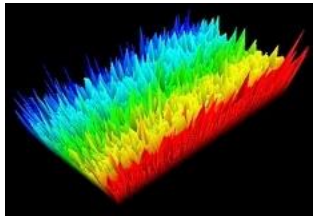
[2] J.T.Daly, A higher order estimate of the optimum checkpoint interval for restart dumps. Future Generation Computer Systems, Volume 22 Issue 3, February 2006, Pages 303-312

PERFORMANCE

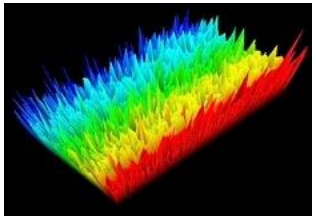


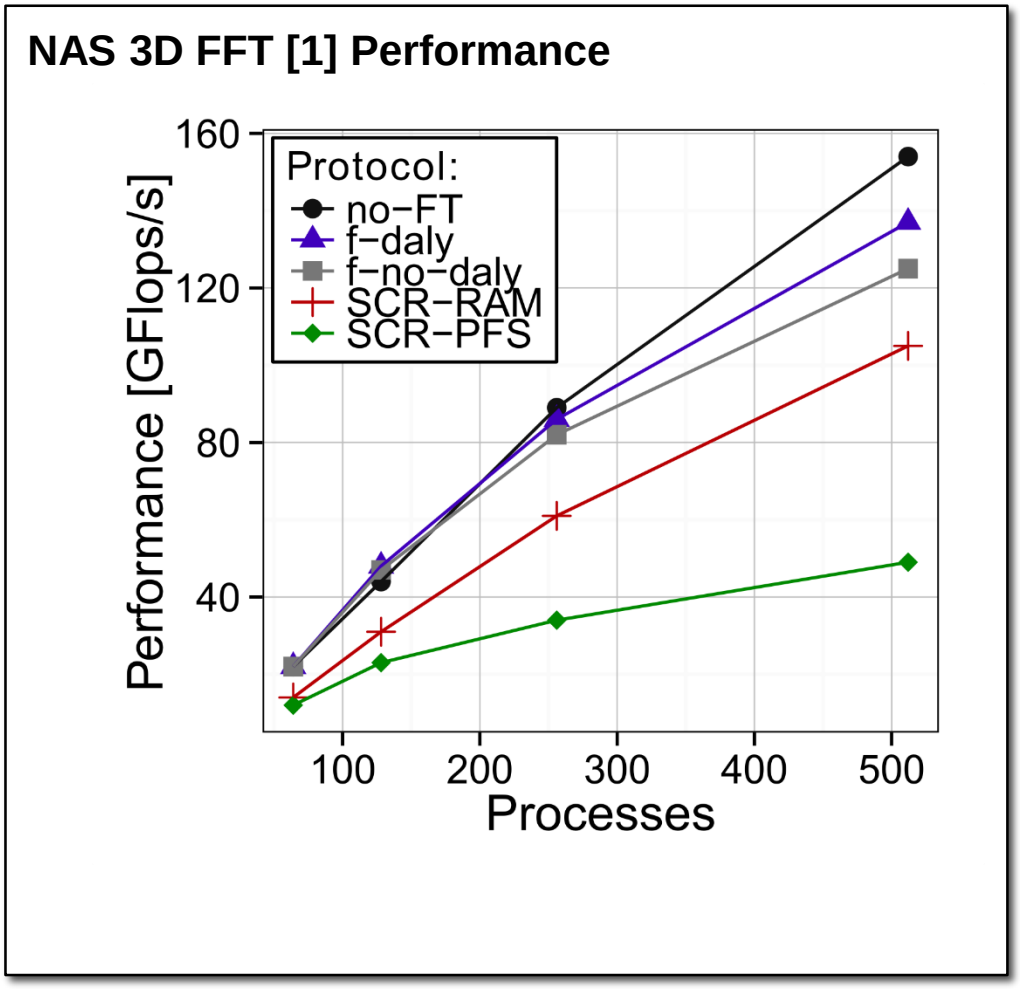
- Evaluation on CSCS Monte Rosa
  - 1,496 computing Cray XE6 nodes
  - 47,872 schedulable cores
  - 46TB memory
- 4 protocols
- 2 applications

Performance: Coordinated Checkpointing NAS 3D FFT



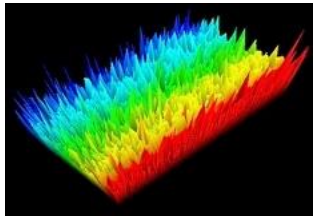
PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT

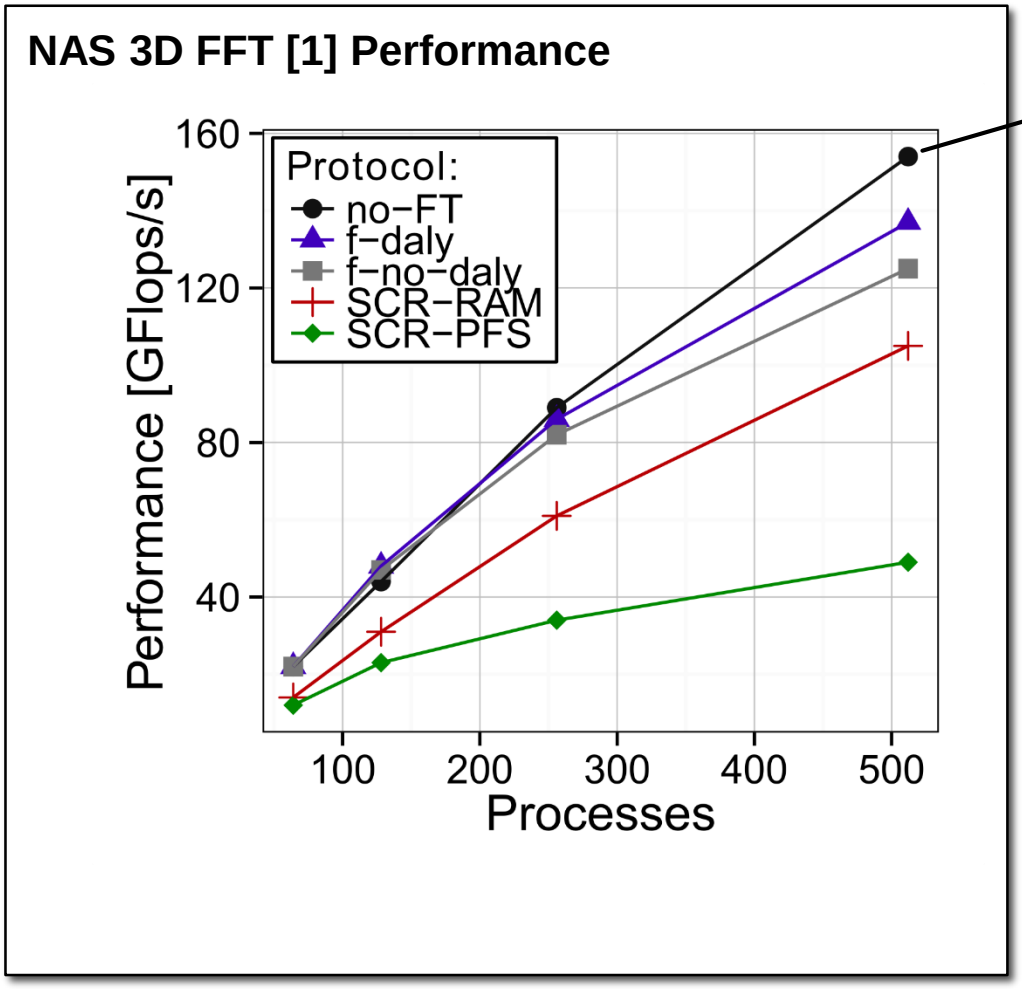




[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT

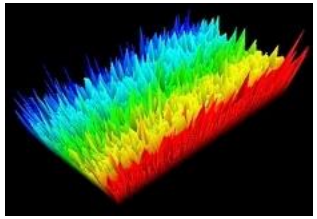


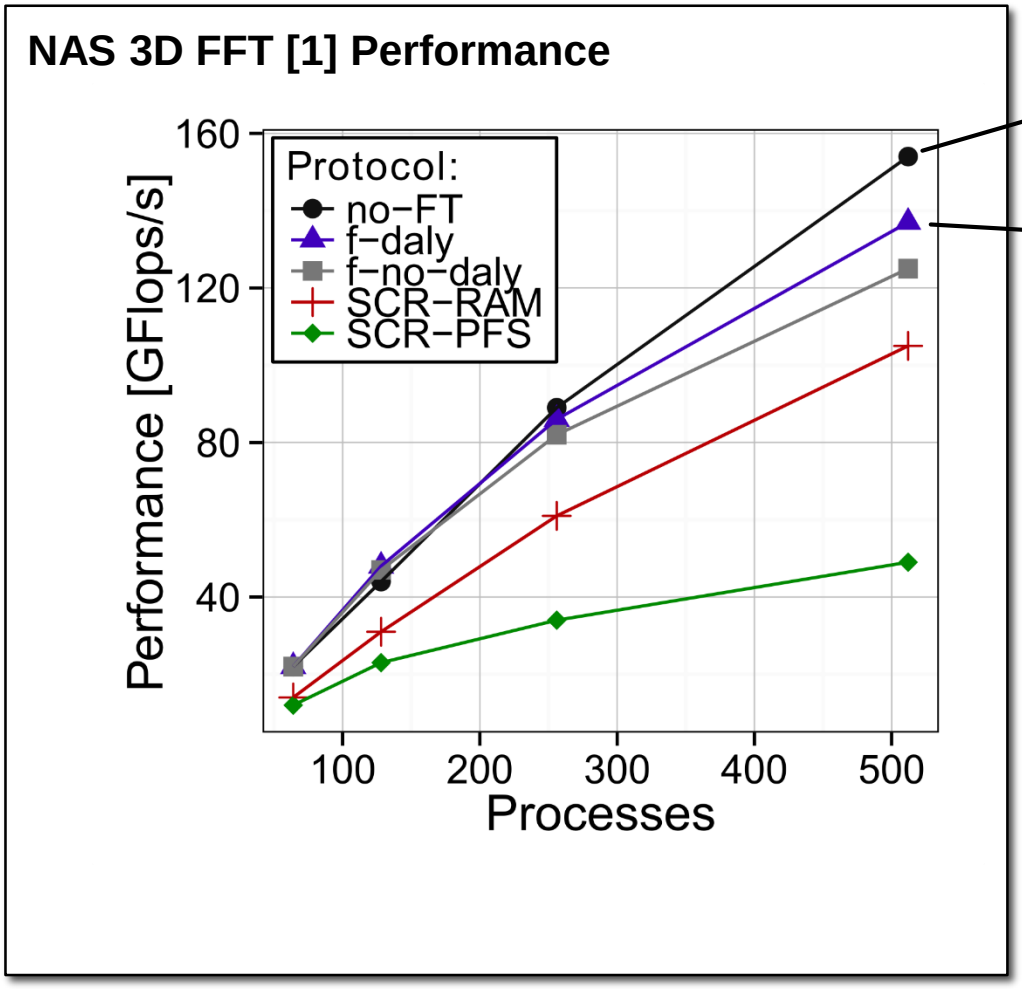


no-FT: no fault tolerance

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT



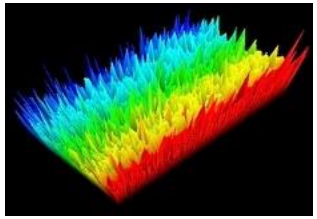


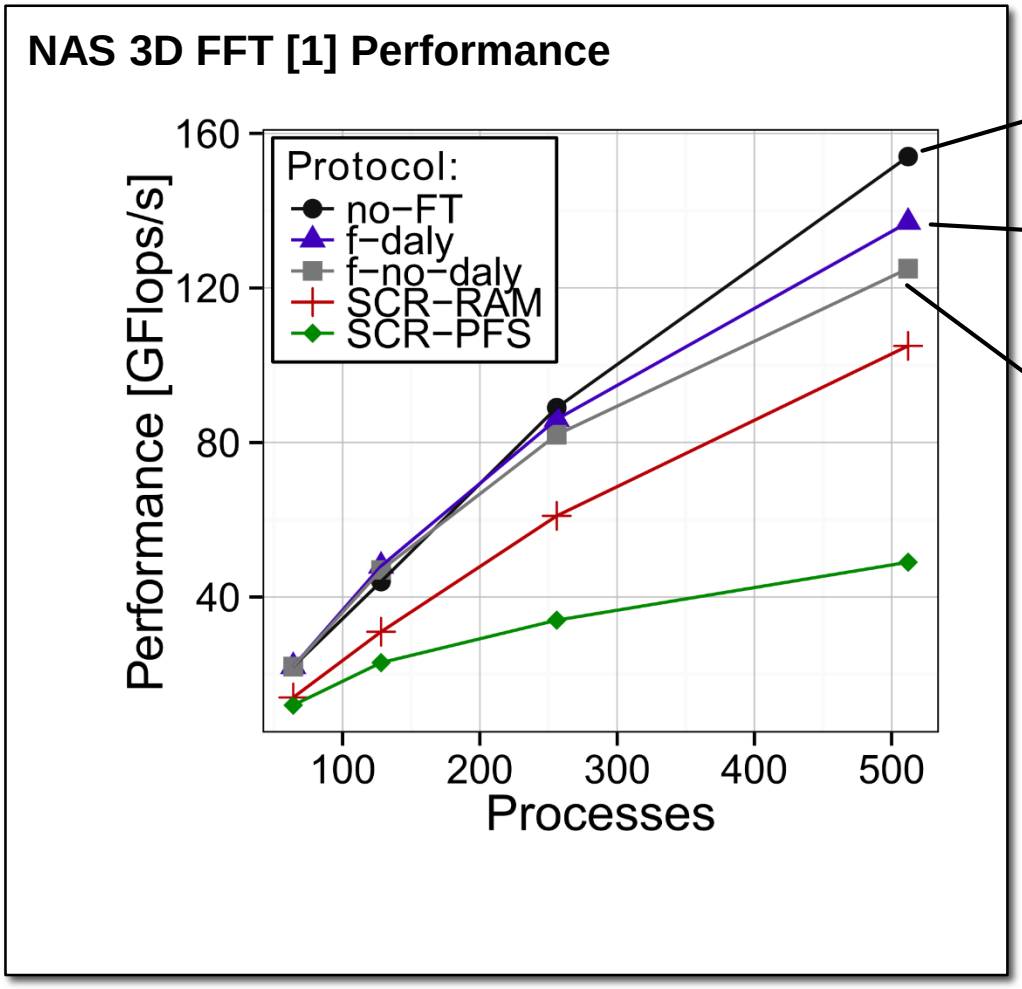
no-FT: no fault tolerance

f-daly: using Daly's interval
1-5% slower than no-FT

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT





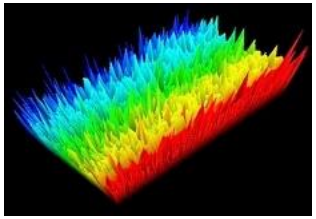
no-FT: no fault tolerance

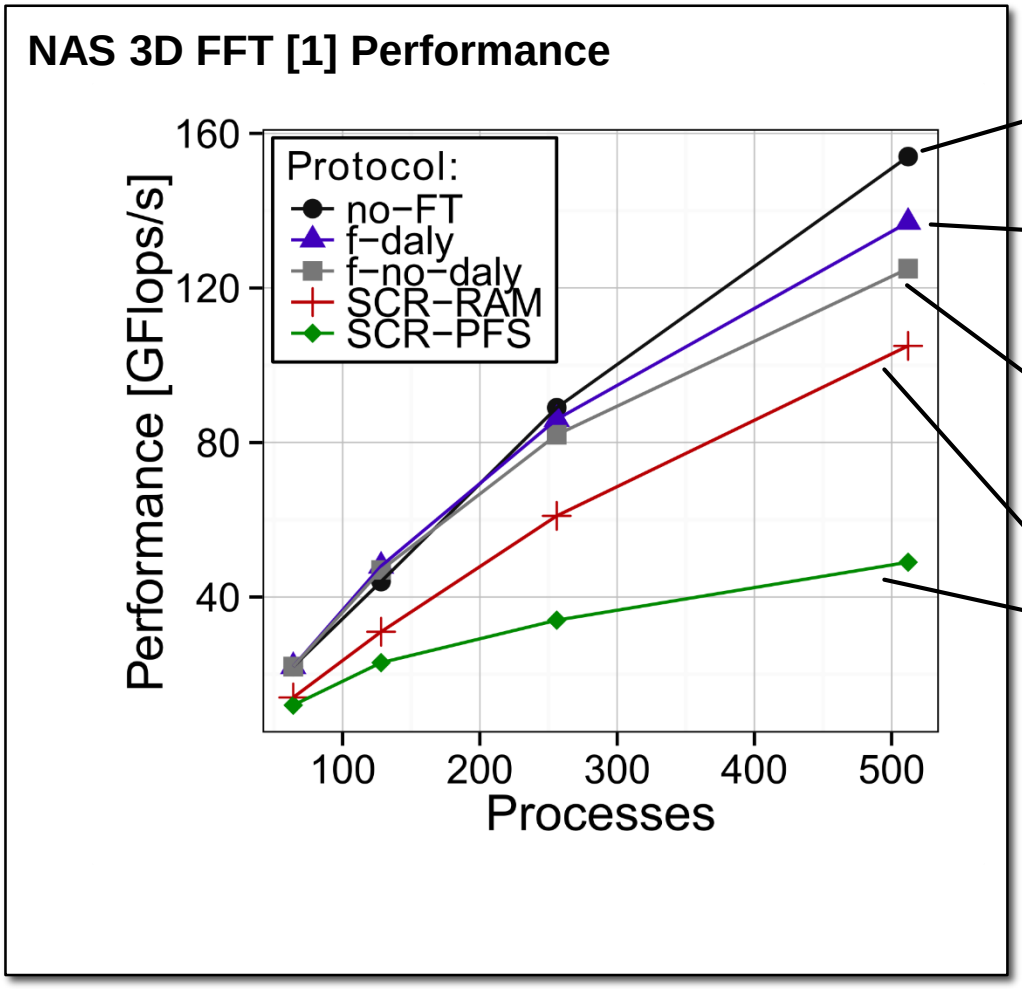
f-daly: using Daly's interval
1-5% slower than no-FT

f-no-daly: no Daly's interval
1-15% slower than no-FT

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT





no-FT: no fault tolerance

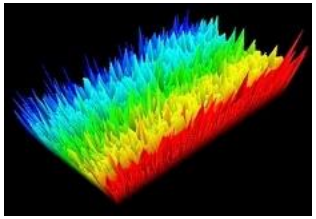
f-daly: using Daly's interval
1-5% slower than no-FT

f-no-daly: no Daly's interval
1-15% slower than no-FT

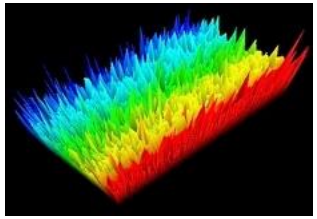
SCR: a popular checkpoint / restart library
21-67% slower than no-FT

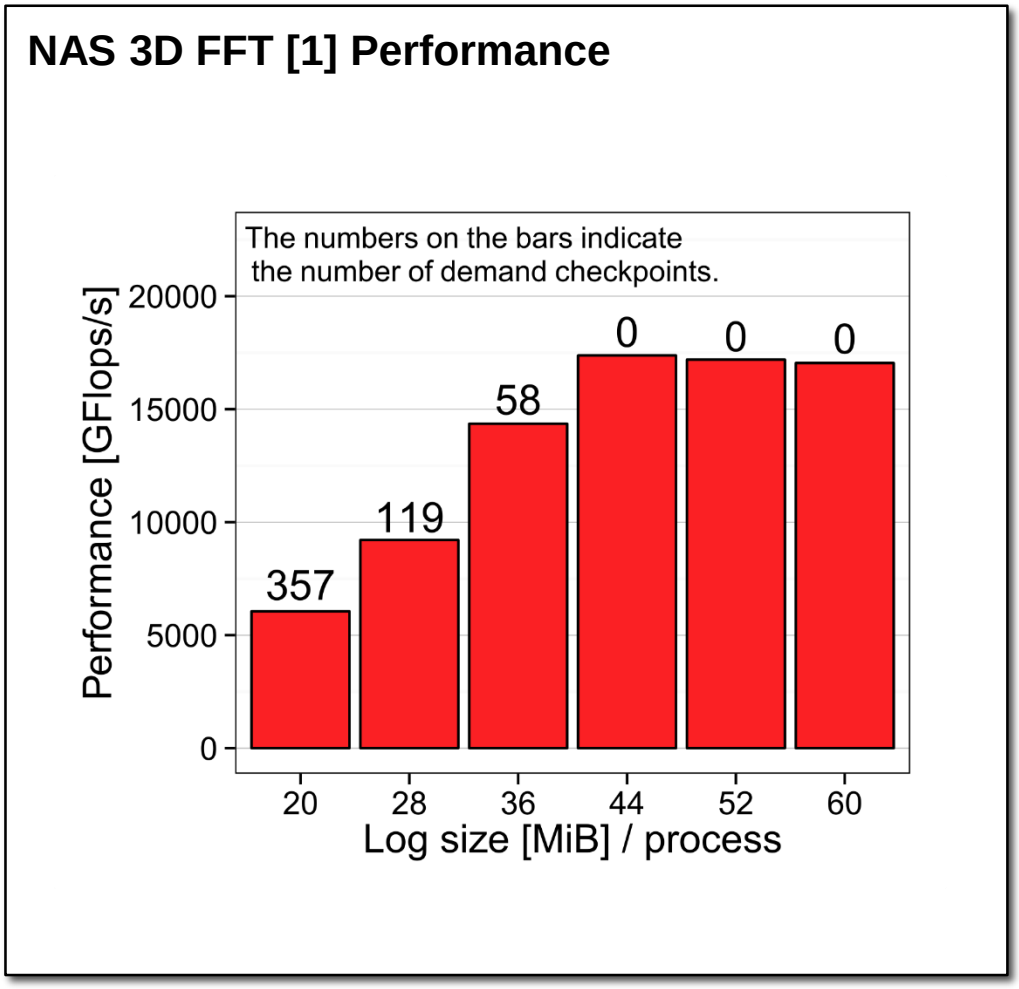
[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

Performance: Uncoordinated Checkpointing NAS 3D FFT



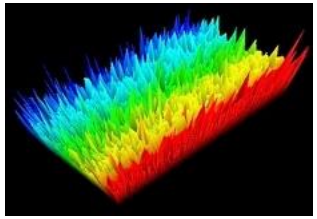
PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT

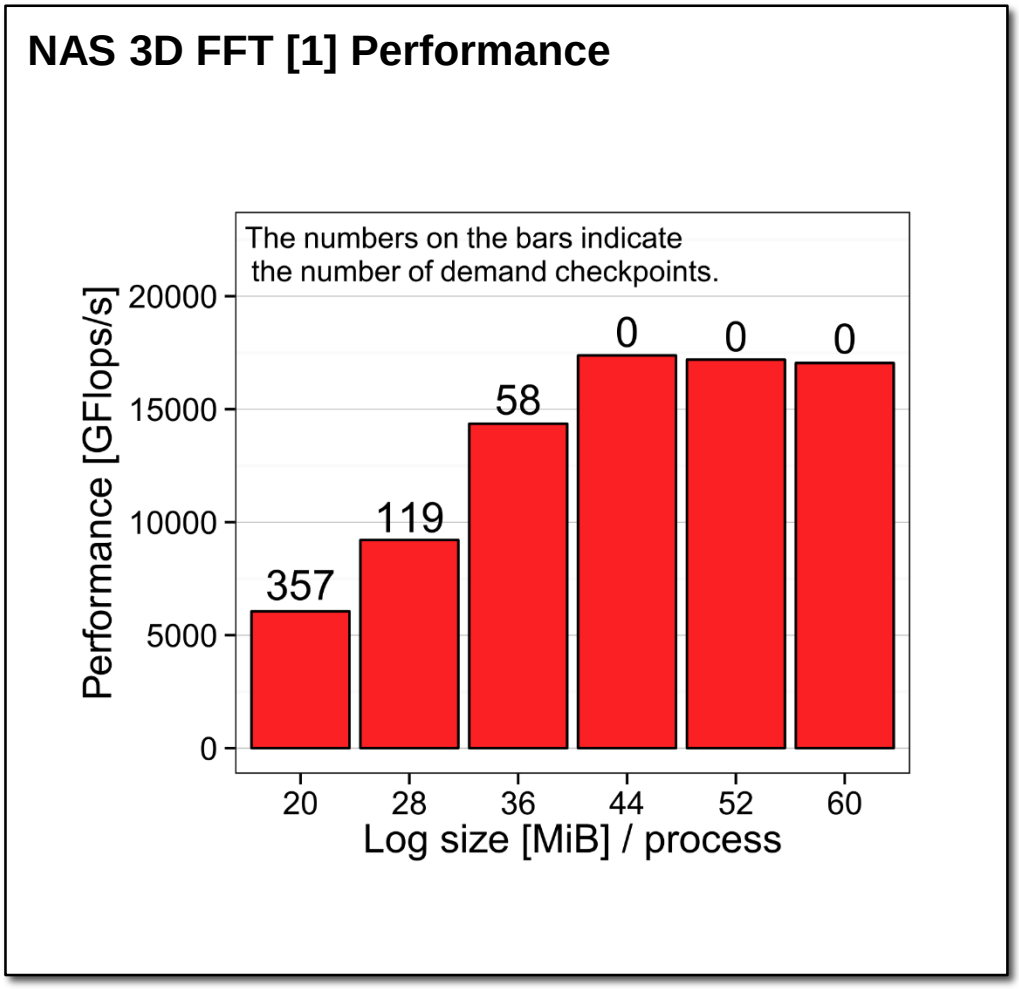




[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT

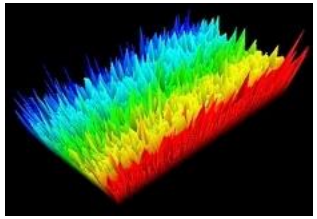


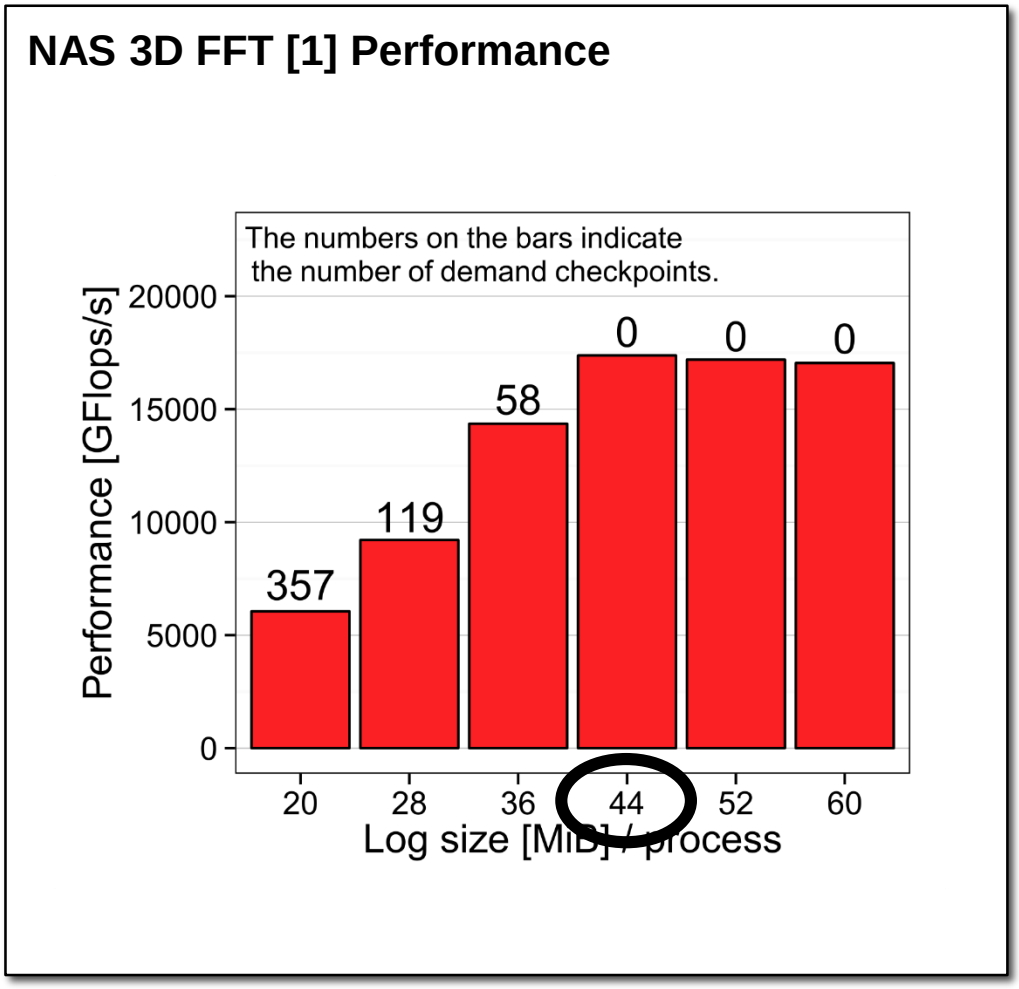


How the log size impacts the number of uncoordinated checkpoints and the performance of the code?

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT

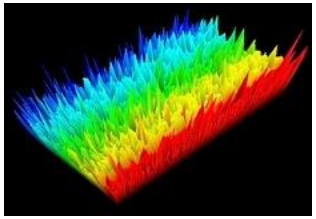




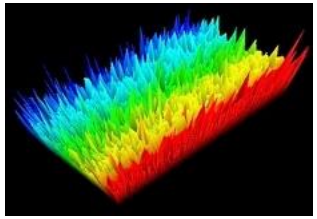
How the log size impacts the number of uncoordinated checkpoints and the performance of the code?

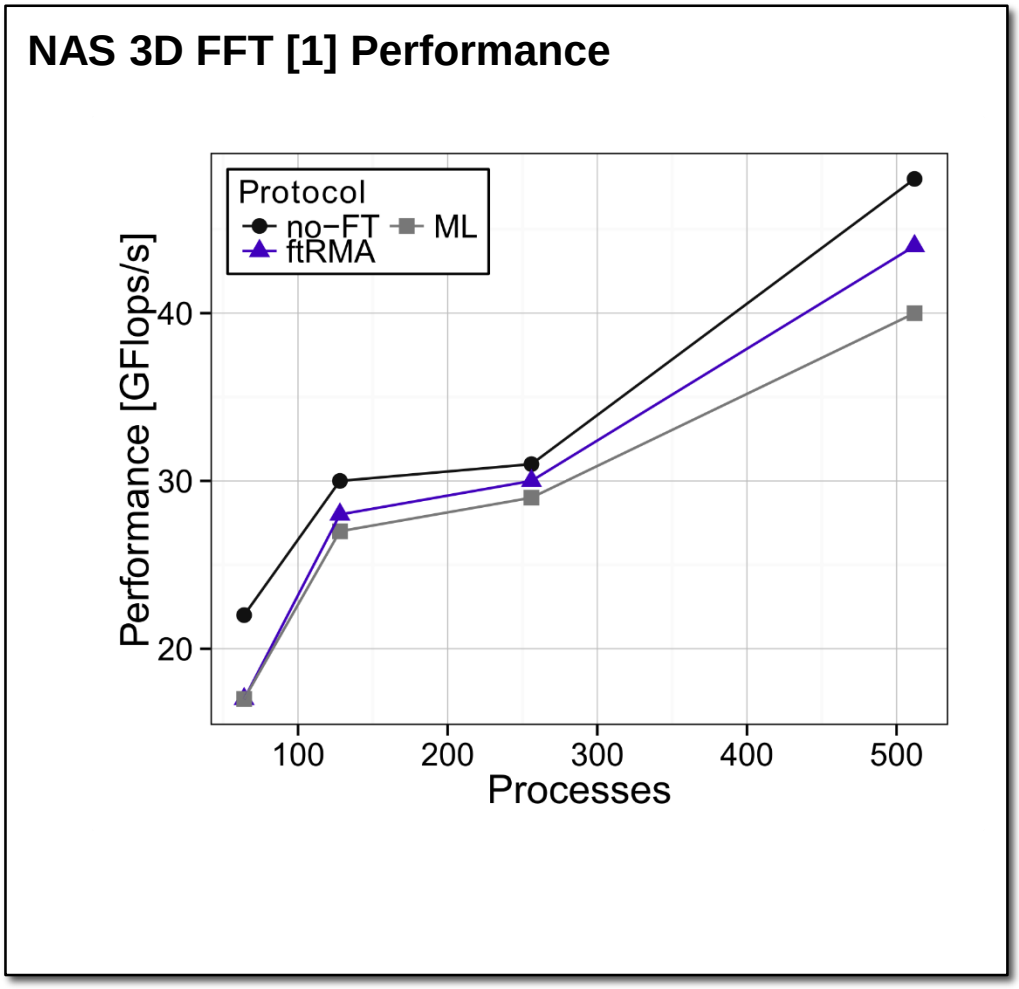
[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: ACCESS LOGGING
NAS 3D FFT



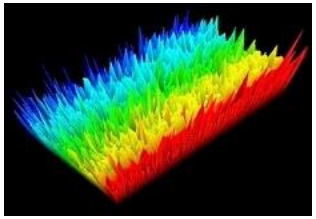
PERFORMANCE: ACCESS LOGGING NAS 3D FFT

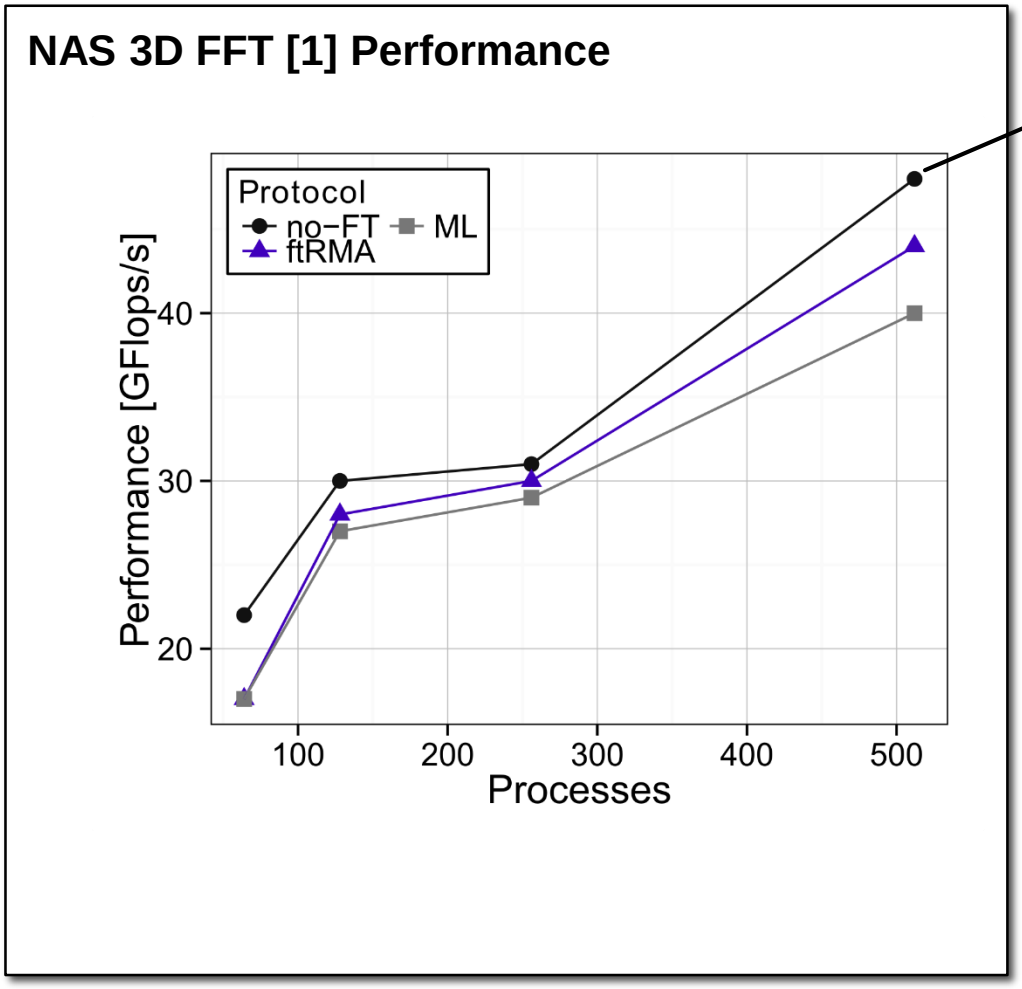




[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: ACCESS LOGGING
NAS 3D FFT

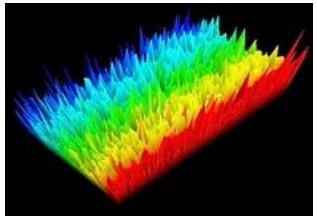


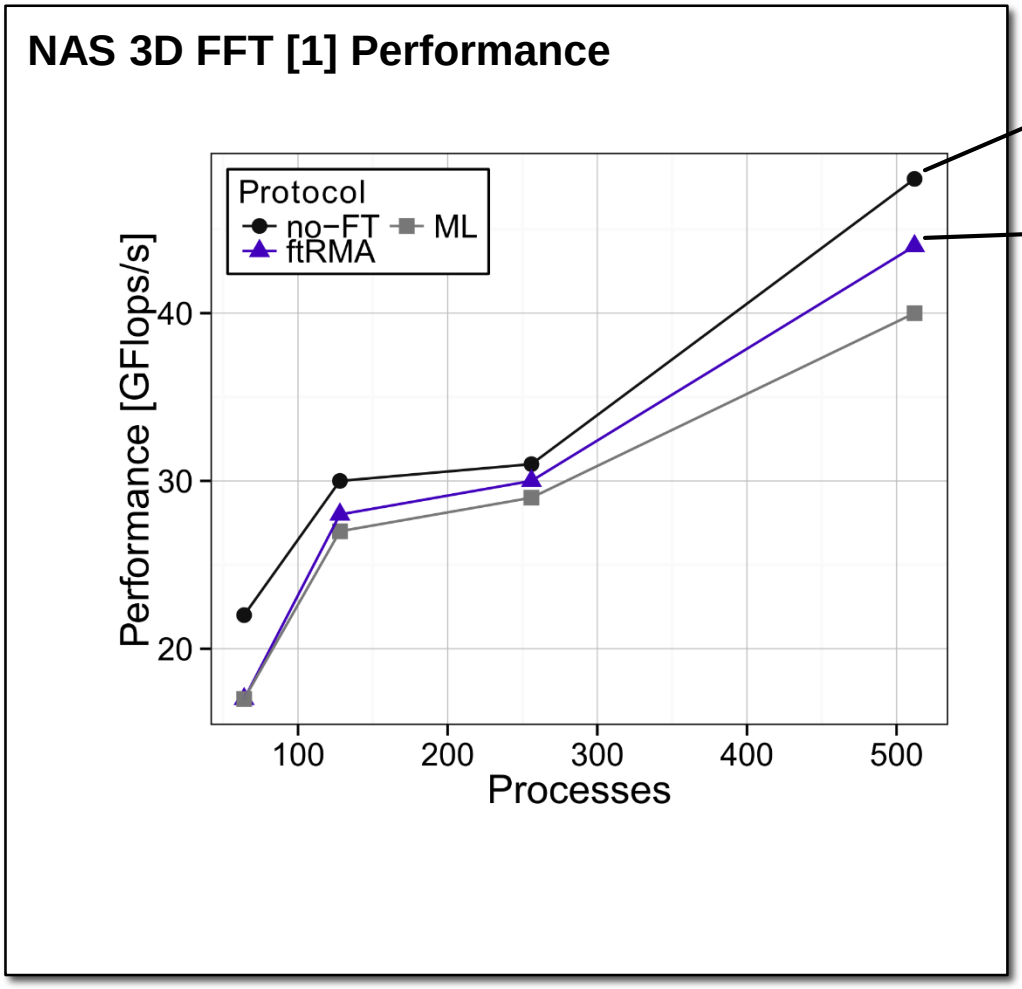


no-FT: no fault tolerance

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: ACCESS LOGGING
NAS 3D FFT



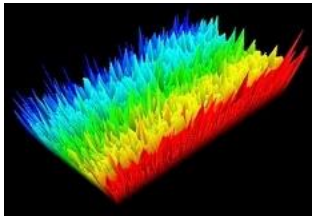


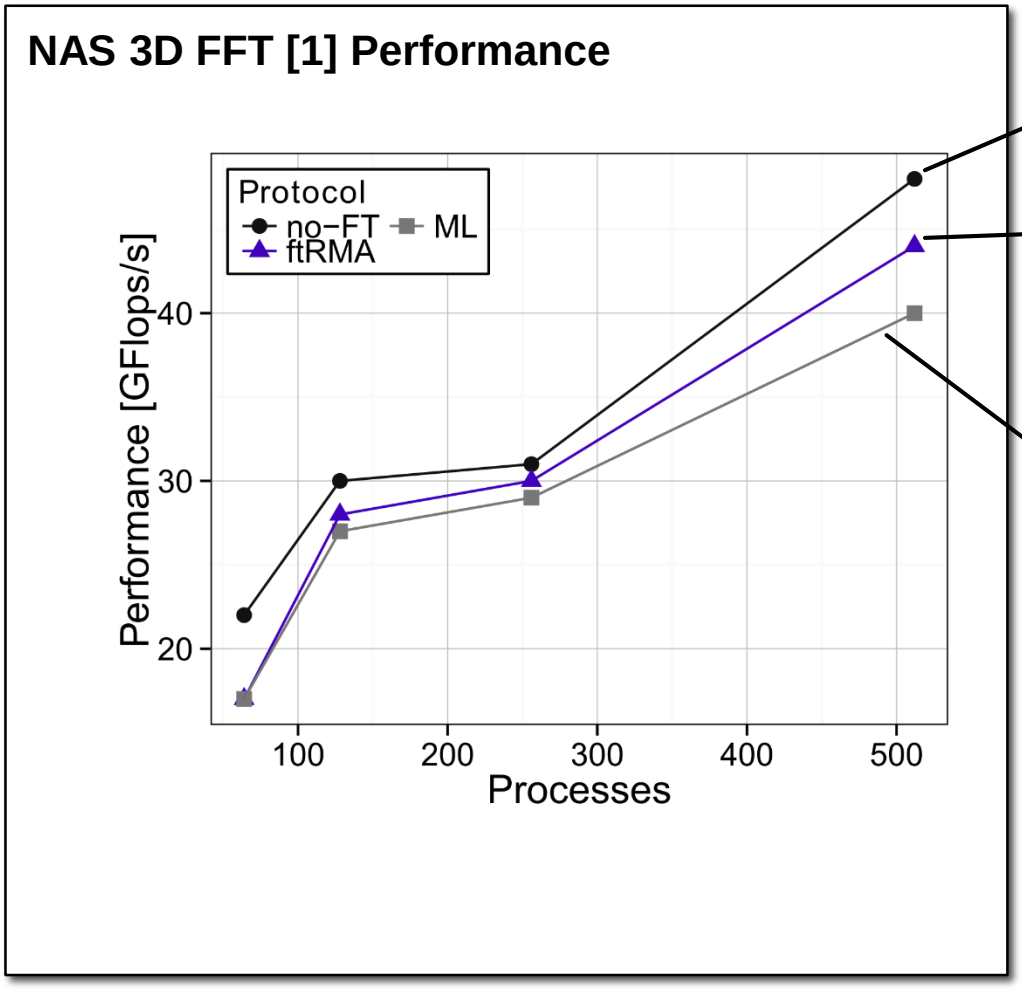
no-FT: no fault tolerance

FTRMA: logging puts (FFT code does not use gets)
8-9% slower than no-FT

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

PERFORMANCE: ACCESS LOGGING
NAS 3D FFT





no-FT: no fault tolerance

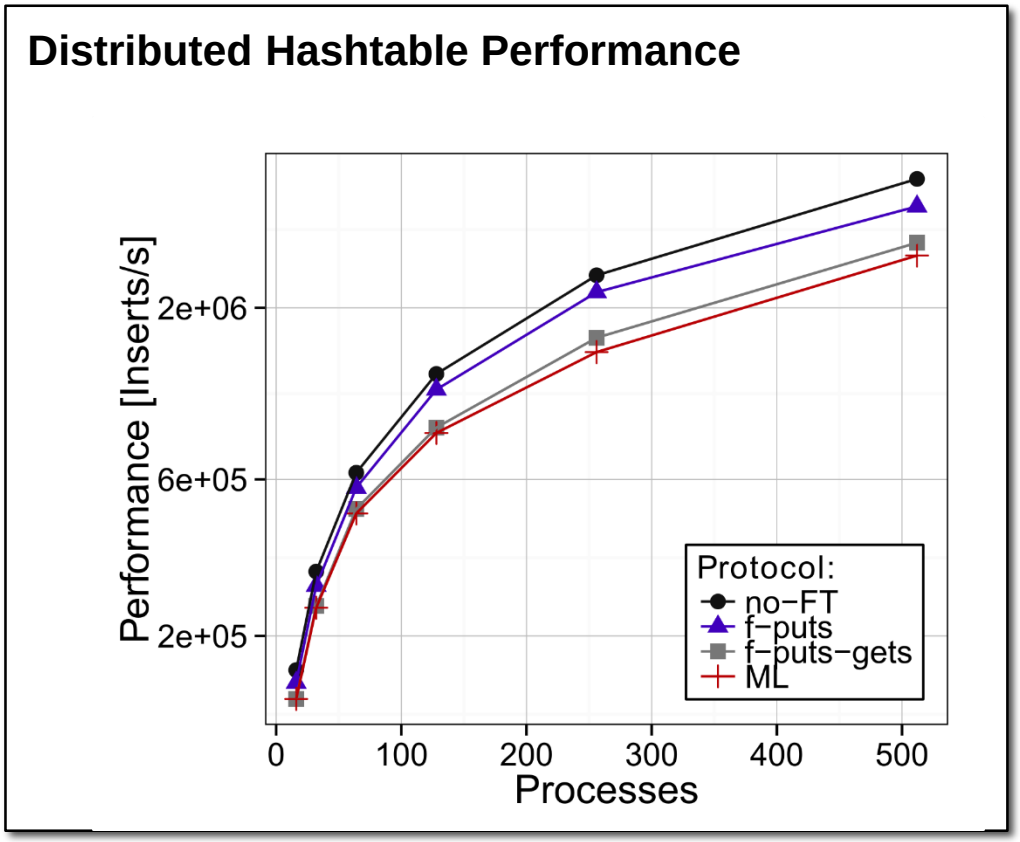
FTRMA: logging puts (FFT code does not use gets) 8-9% slower than no-FT

ML: a simple protocol based on message logging 18% slower than no-FT

[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap.IPDPS'09

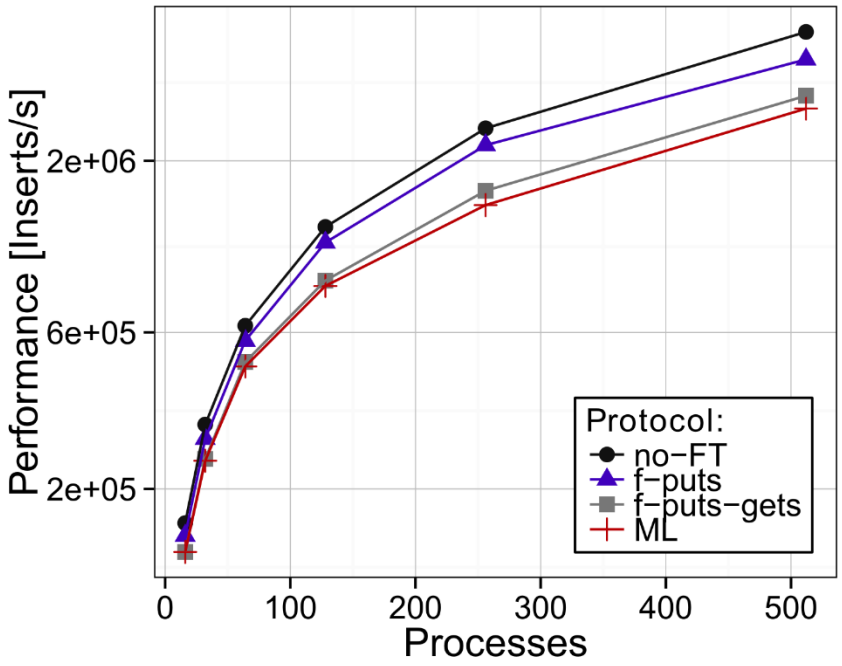
Performance: Access Logging
Distributed Hashtable

PERFORMANCE: ACCESS LOGGING
DISTRIBUTED HASHTABLE



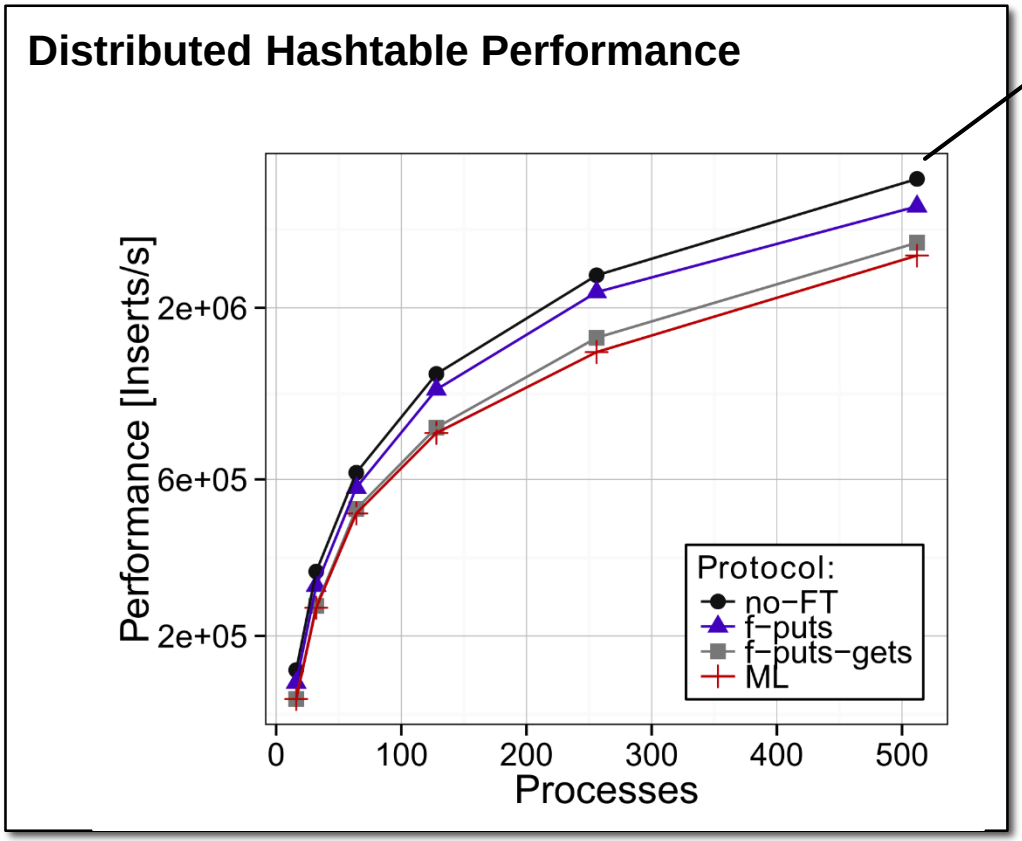
Performance: Access Logging
Distributed Hashtable

Distributed Hashtable Performance



An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

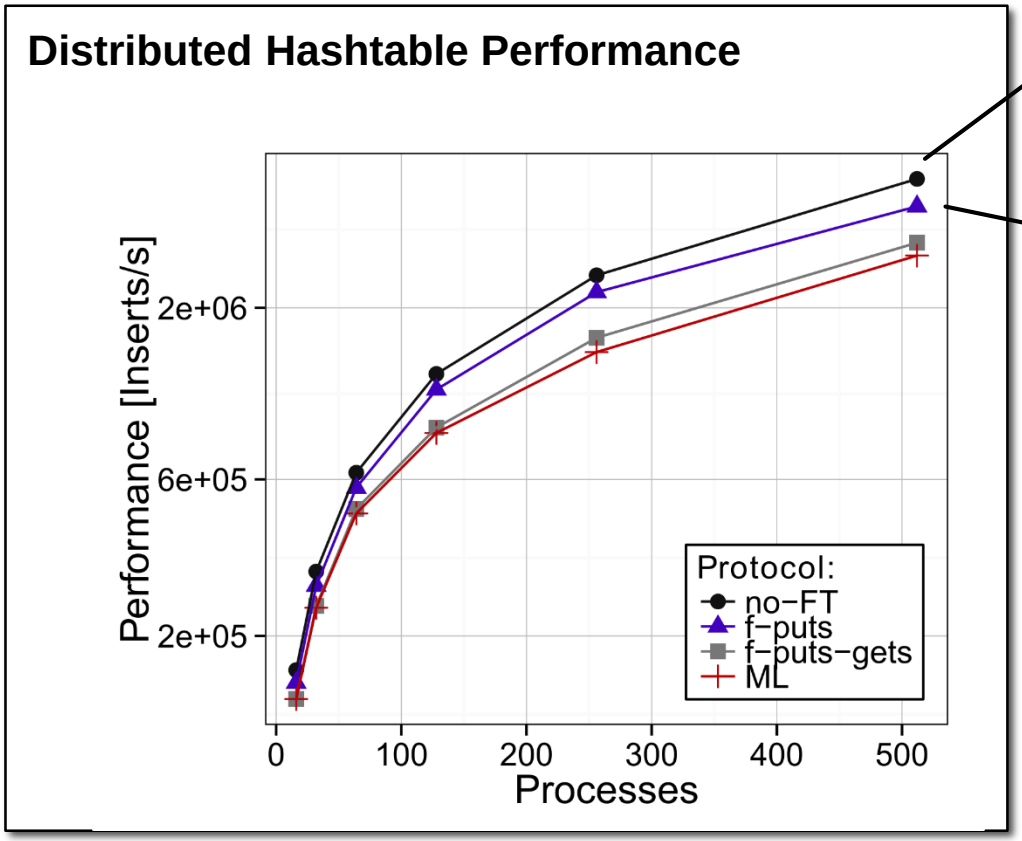
Performance: Access Logging
Distributed Hashtable



no-FT: no fault tolerance

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

Performance: Access Logging
Distributed Hashtable

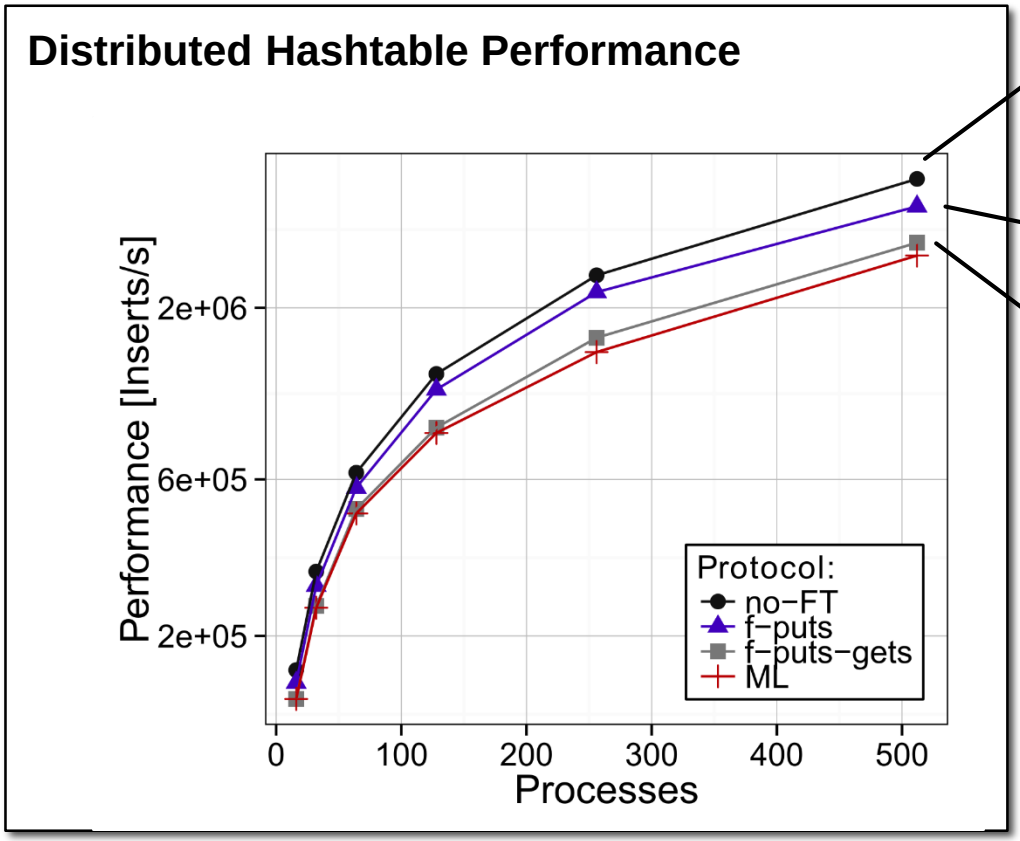


no-FT: no fault tolerance

f-puts: logging puts
12% slower than no-FT

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

Performance: Access Logging
Distributed Hashtable



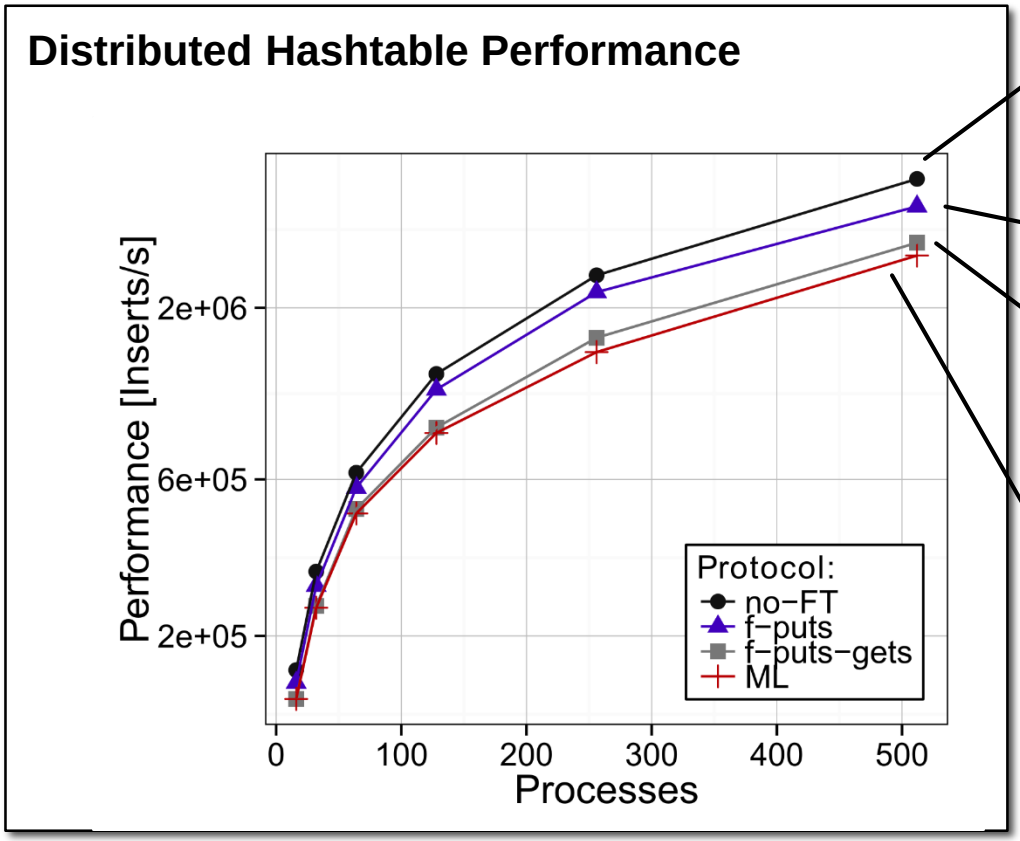
no-FT: no fault tolerance

f-puts: logging puts
12% slower than no-FT

f-puts-gets: logging puts and gets
33% slower than no-FT

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

Performance: Access Logging
Distributed Hashtable



no-FT: no fault tolerance

f-puts: logging puts
12% slower than no-FT

f-puts-gets: logging puts and gets
33% slower than no-FT

ML: logging puts and gets
40% slower than no-FT

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

MP vs. RMA

Schemes

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Overview of our Research

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

$\text{PUT}(p \rightrightarrows q)$
$\text{GET}(p \leftrightarrows q)$

MPI-2
PGAS

CC in RMA

MP vs. RMA

Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

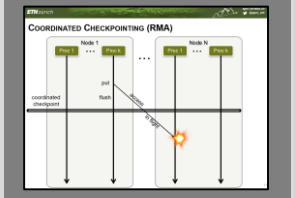
Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

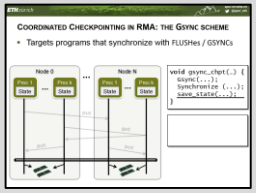
$\text{PUT}(p \rightrightarrows q)$
$\text{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA



MP vs. RMA



Schemes

UC in RMA

MP vs. RMA

Checkpointing schemes

Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. The recove rithm 2 preserves the $\xrightarrow{cohb}$ or ferred to as the gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

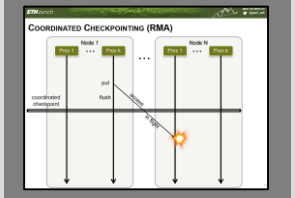
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

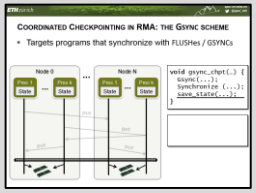
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

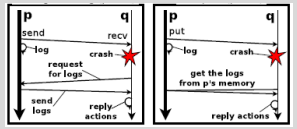


MP vs. RMA

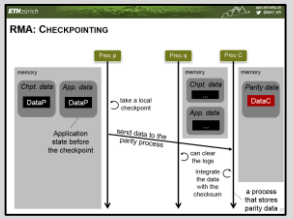


Schemes

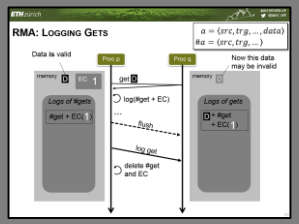
UC in RMA



MP vs. RMA



Checkpointing schemes



Logging accesses

Recovery in RMA

Basic scheme

Extended RMA semantics

Proofs

Theorem 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

Distribution of processes

Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

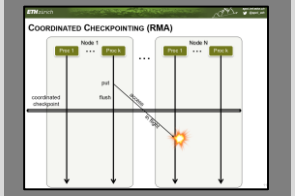
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

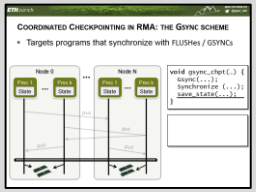
$$\mathrm{PUT}(p \rightrightarrows q)$$
$$\mathrm{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA

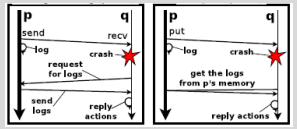


MP vs. RMA

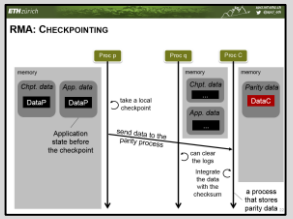


Schemes

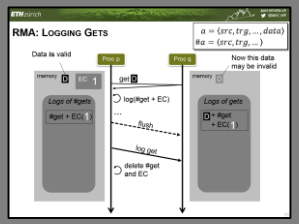
UC in RMA



MP vs. RMA

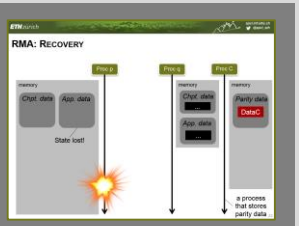


Checkpointing schemes



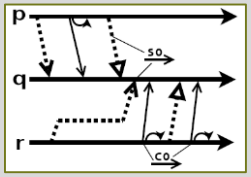
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

Theorem 4.2. *The recovery algorithm 2 preserves the* $\xrightarrow{cohb}$ *order (referred to as the gsync order).*

Deadlock freedom
Correct recovery

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Distribution of processes

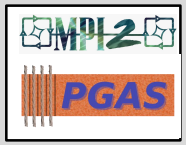
Decreasing failure prob.

Overview of our Research

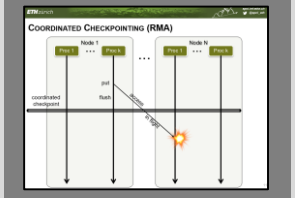
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

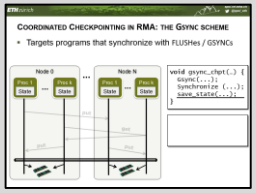
$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$



CC in RMA

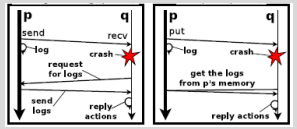


MP vs. RMA

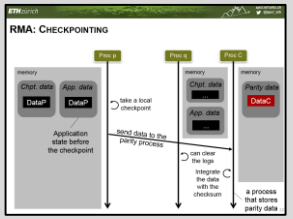


Schemes

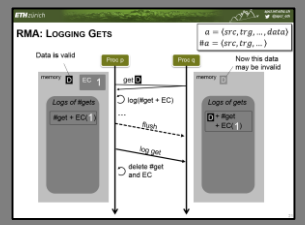
UC in RMA



MP vs. RMA

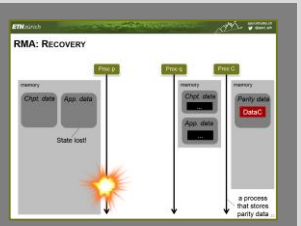


Checkpointing schemes



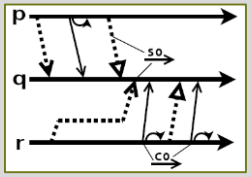
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

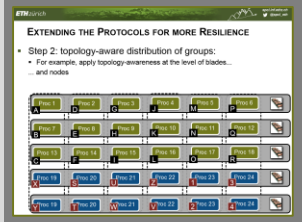
Theorem 4.2. *The recovery algorithm 2 preserves the* $\xrightarrow{cohb}$ *order referred to as the gsync order).*

Deadlock freedom
Correct recovery

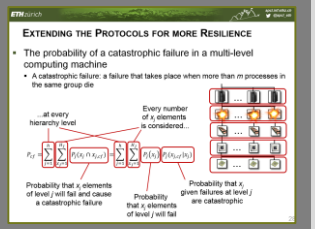
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library

Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

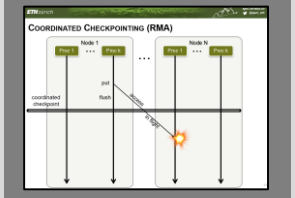
Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

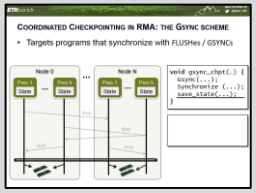
$$\mathrm{PUT}(p \rightrightarrows q)$$
$$\mathrm{GET}(p \leftrightarrows q)$$

MPI-2 PGAS

CC in RMA

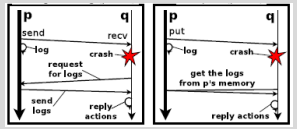


MP vs. RMA

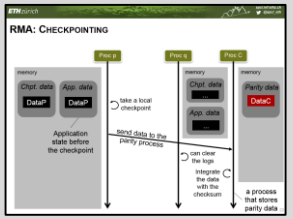


Schemes

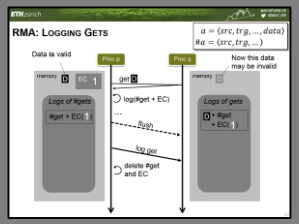
UC in RMA



MP vs. RMA

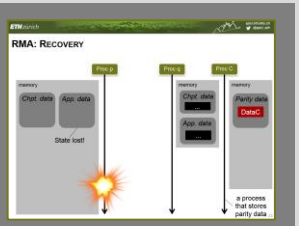


Checkpointing schemes



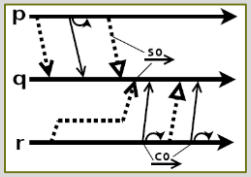
Logging accesses

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

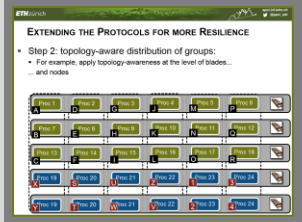
Theorem 4.2. *The recovery algorithm 2 preserves the* $\xrightarrow{cohb}$ *order referred to as the gsync order).*

Deadlock freedom
Correct recovery

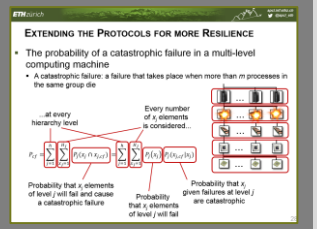
Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

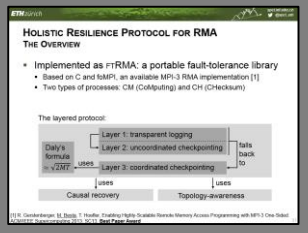


Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

Performance

Overview of our Research

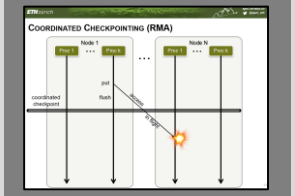
Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$

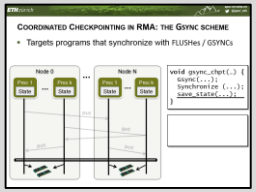
$\text{PUT}(p \rightrightarrows q)$
$\text{GET}(p \leftrightarrows q)$

MPI-2 PGAS

CC in RMA

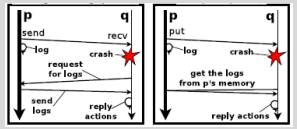


MP vs. RMA

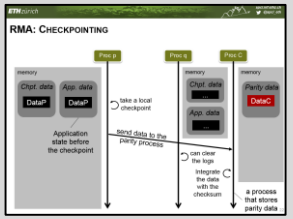


Schemes

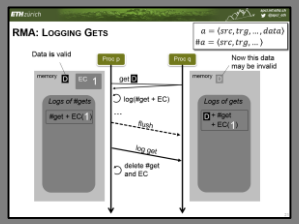
UC in RMA



MP vs. RMA

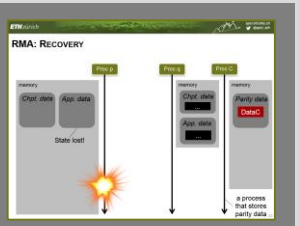


Checkpointing schemes



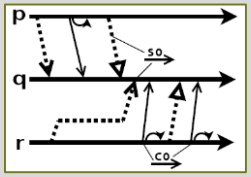
Logging accesses

Recovery in RMA

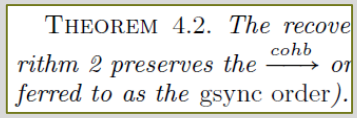


Basic scheme

Extended RMA semantics



Proofs



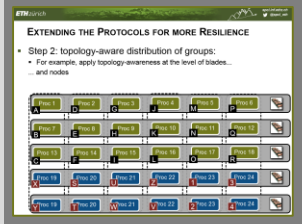
THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

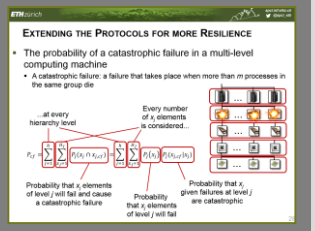
Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

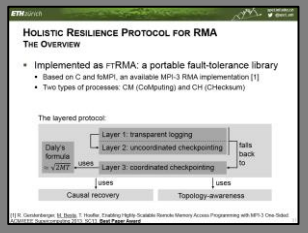


Distribution of processes



Decreasing failure prob.

Holistic fault-tolerance library



Design

Checkpoints on demand

Optimizations

Performance

ACKNOWLEDGMENTS

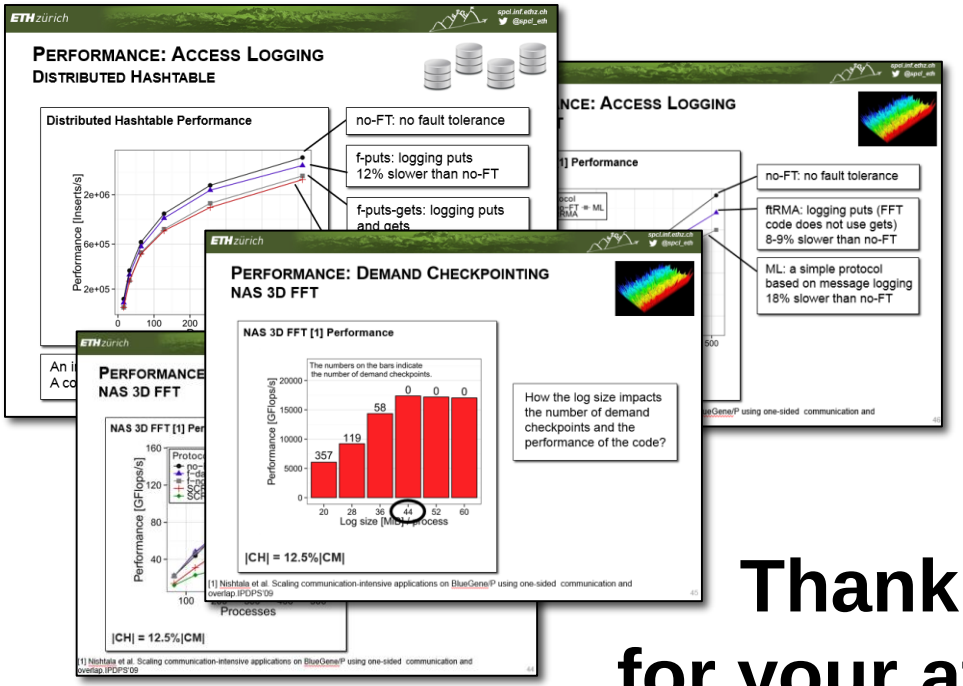
Thanks to:

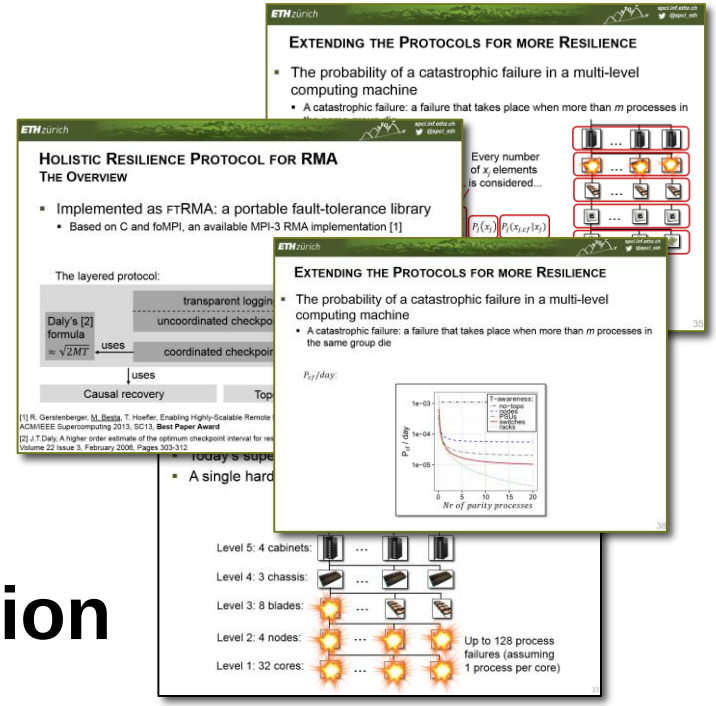
Paul Hargrove (and the whole UPC team)
and the MPI Forum RMA WG …

… and the institutions:

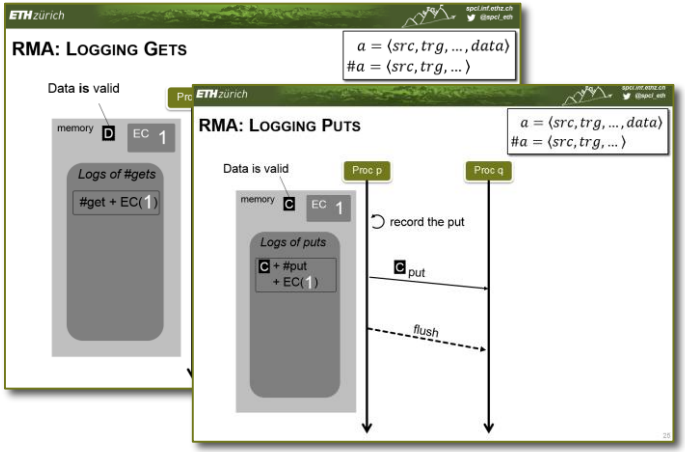
CSCS

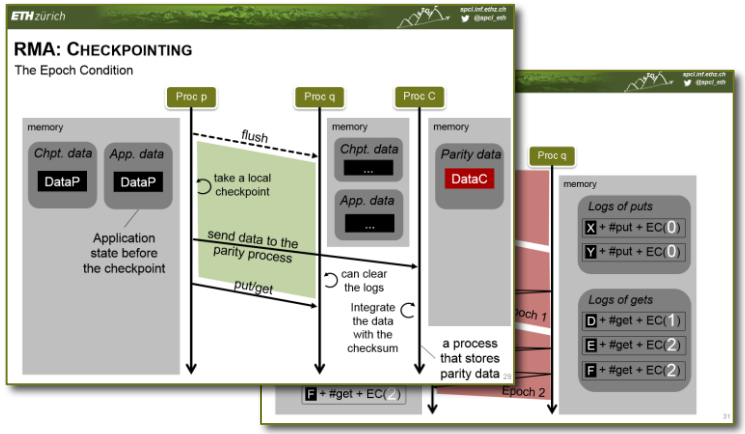
Google™



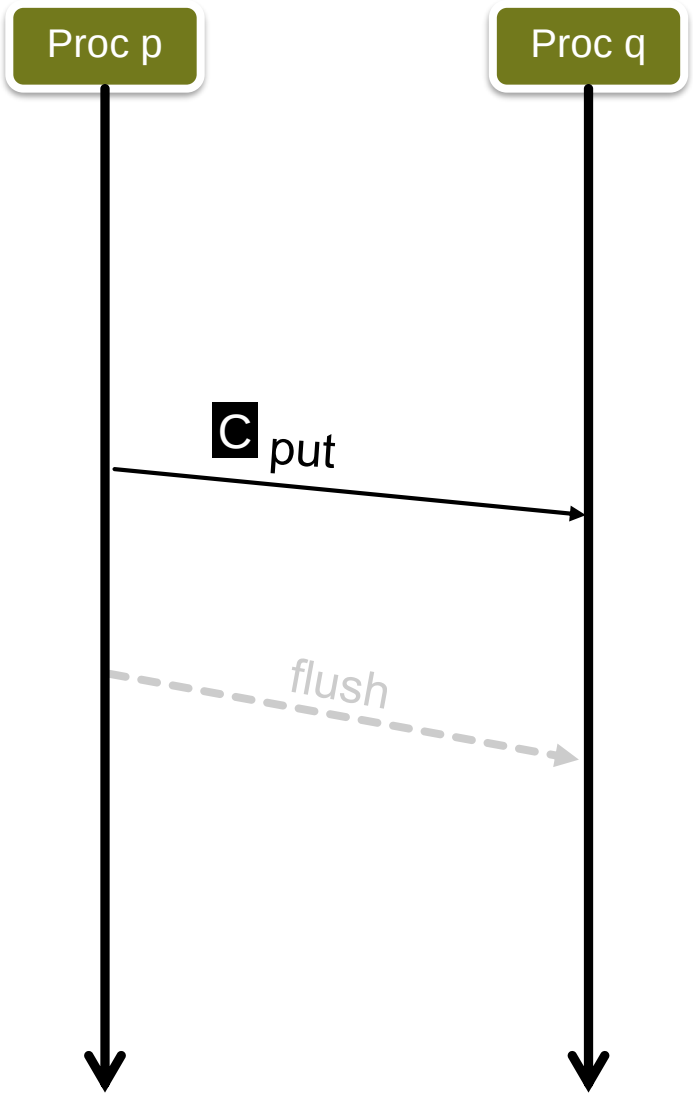


Thank you for your attention

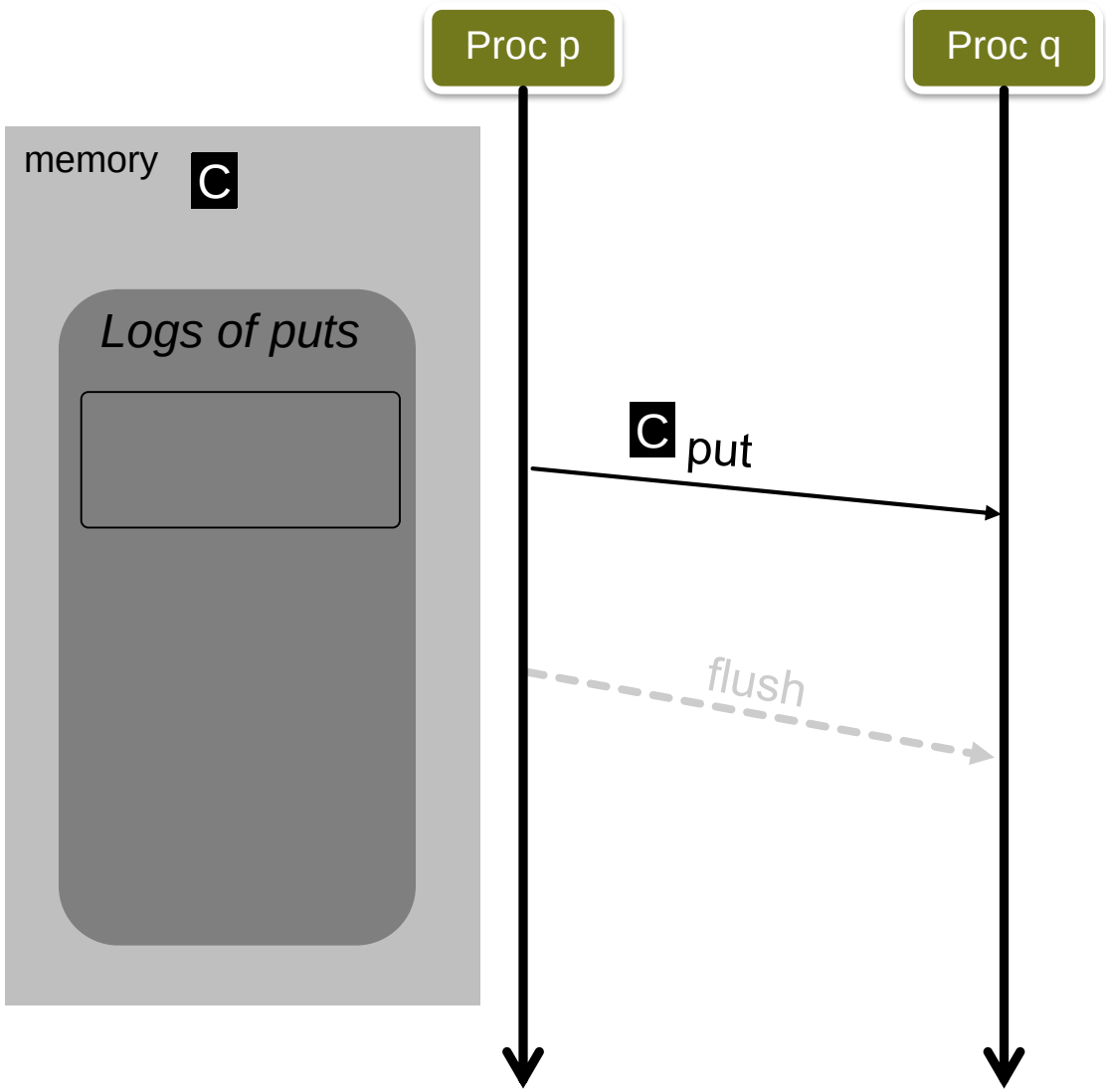




RMA: Logging Puts

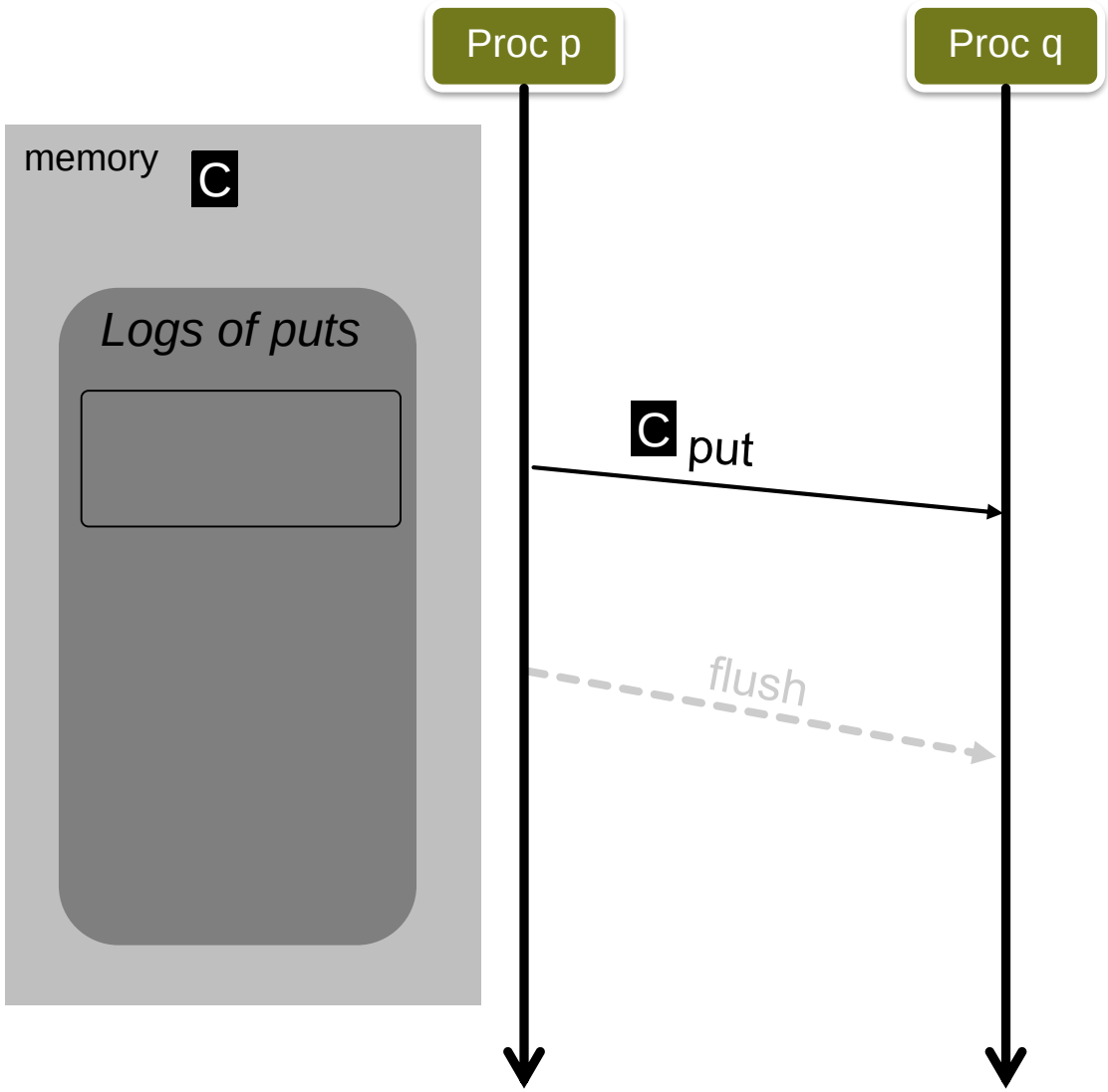


RMA: Logging Puts



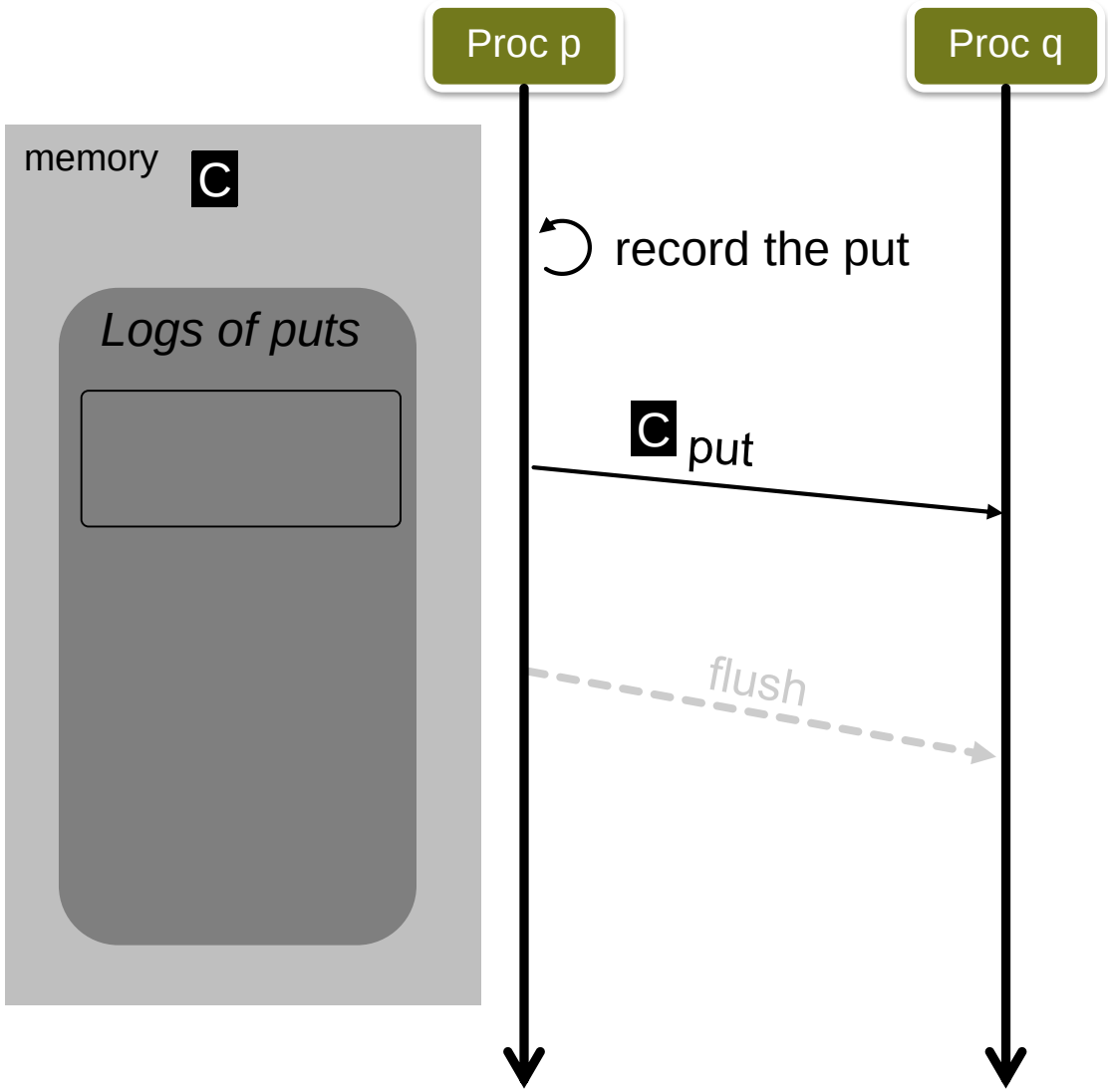
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



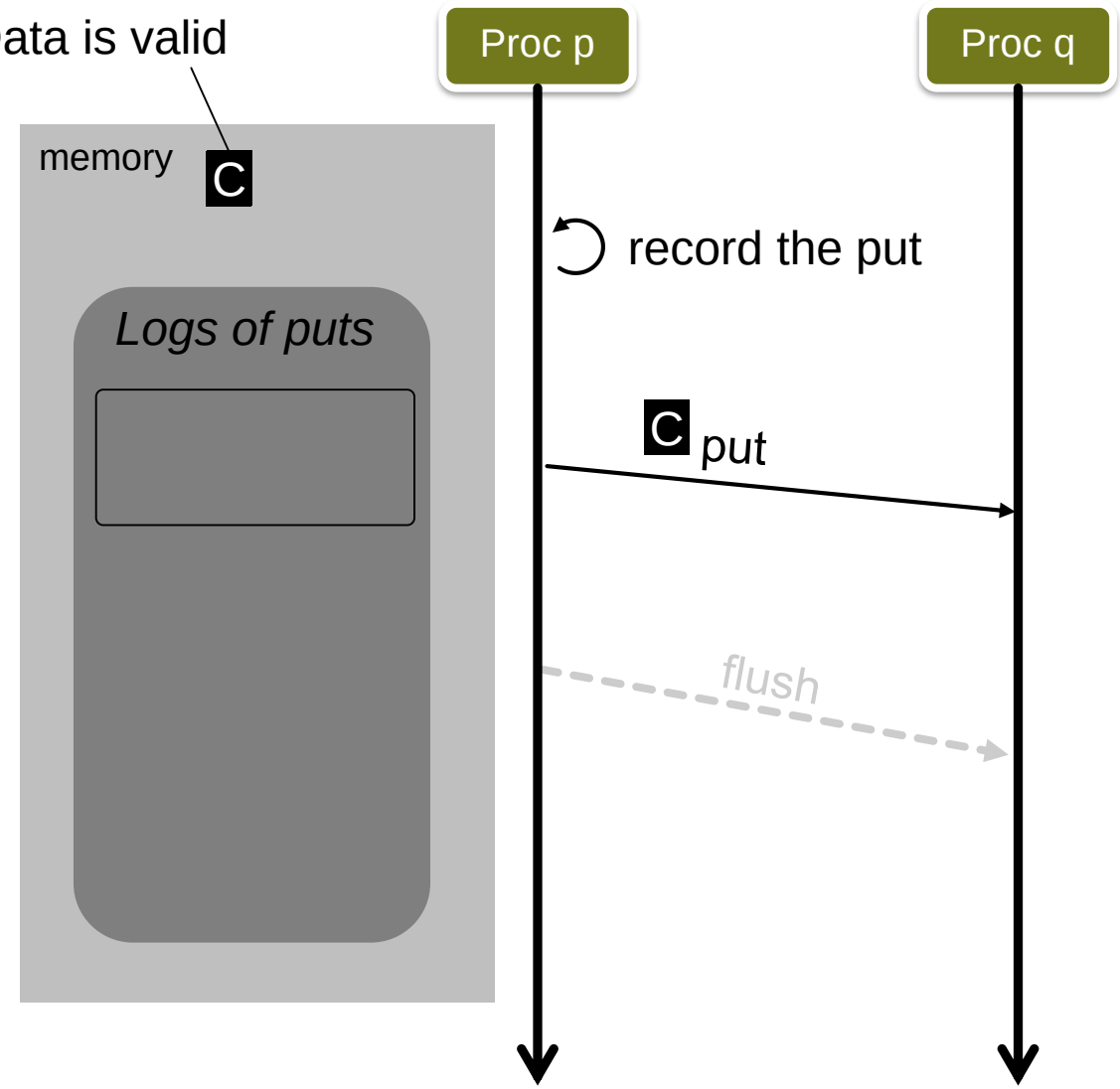
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



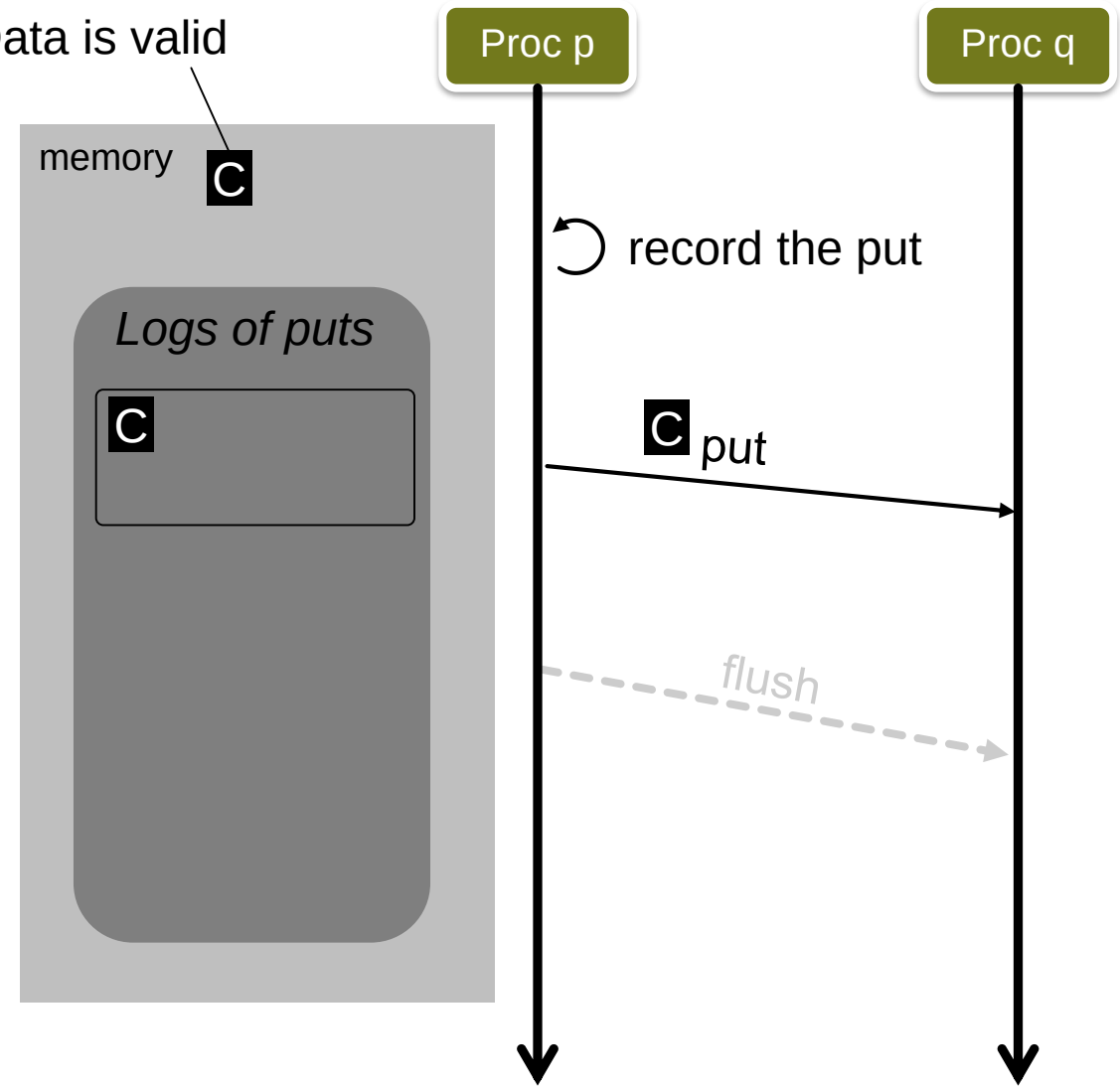
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



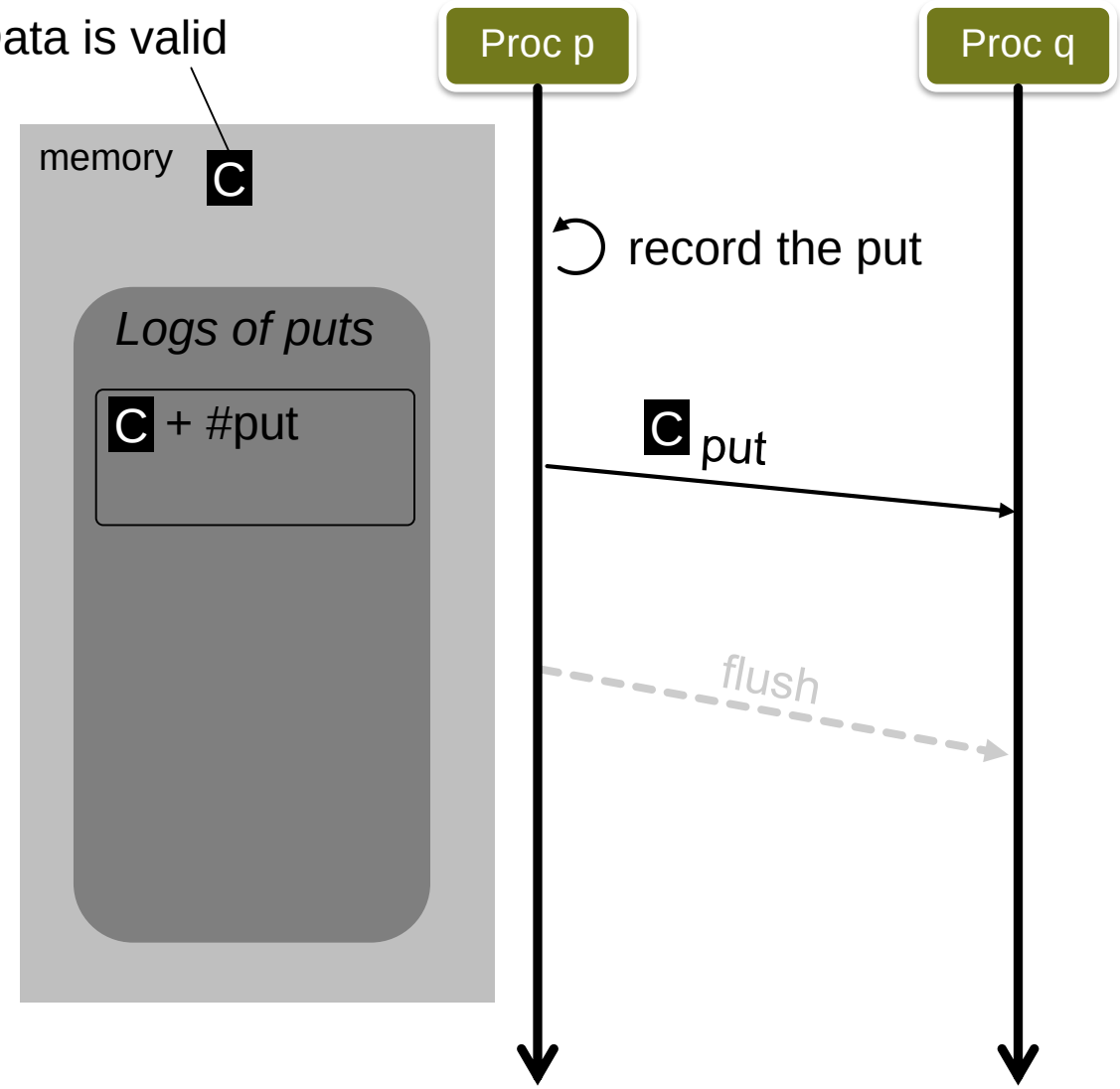
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



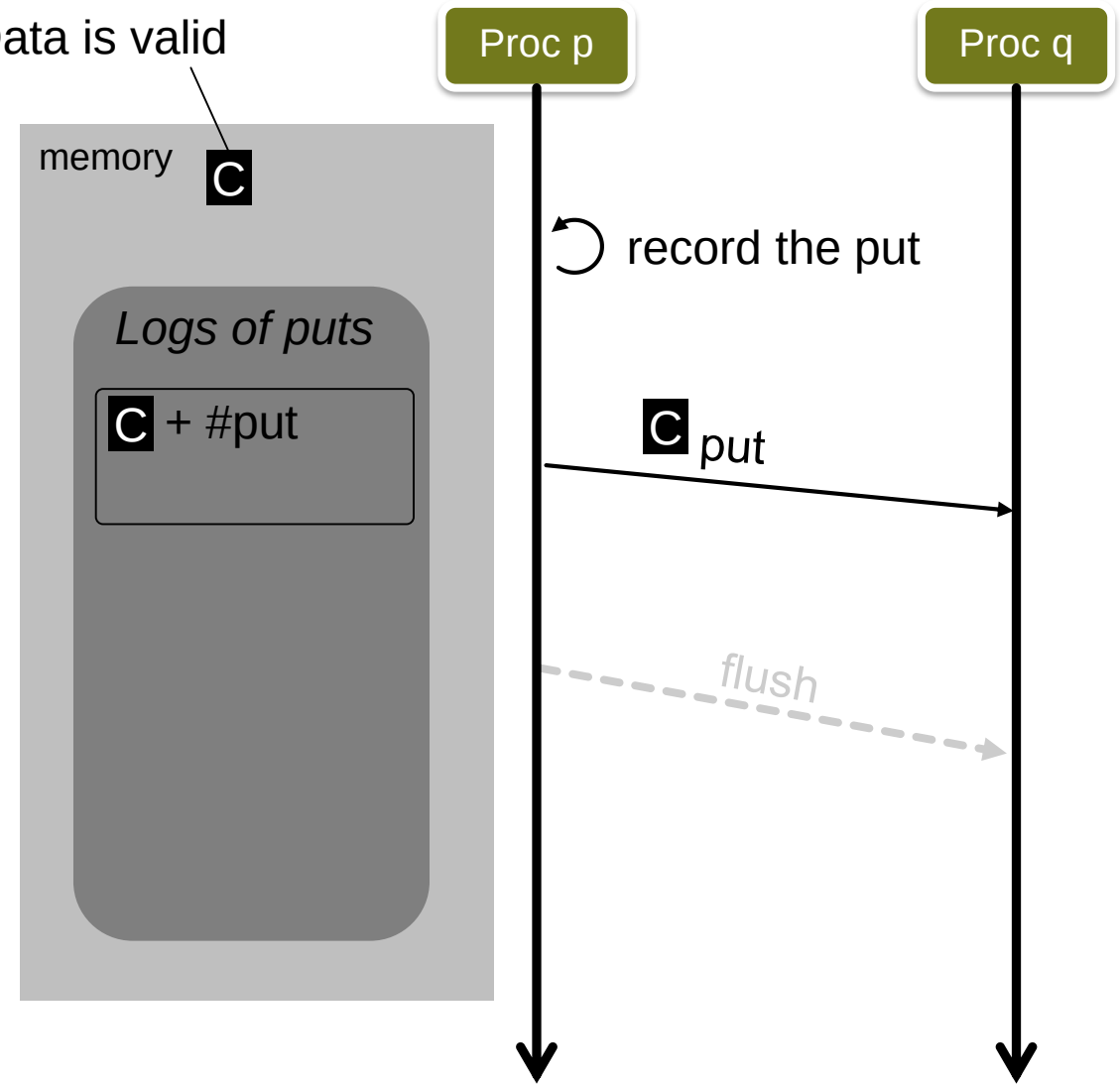
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



RMA: Logging Puts

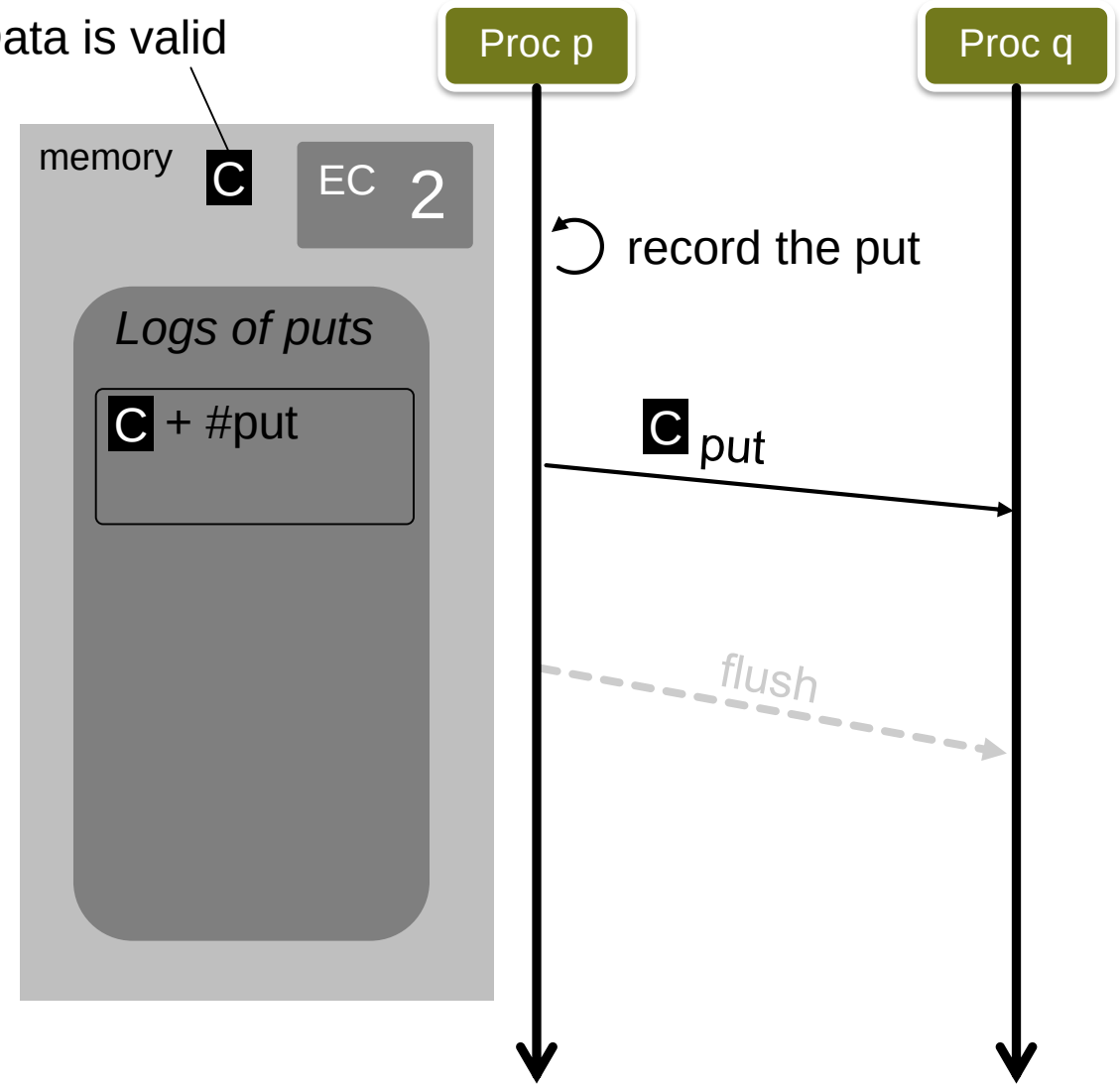
$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



To replay the actions preserving $\xrightarrow{co}$, we also record **epoch counters**

RMA: Logging Puts

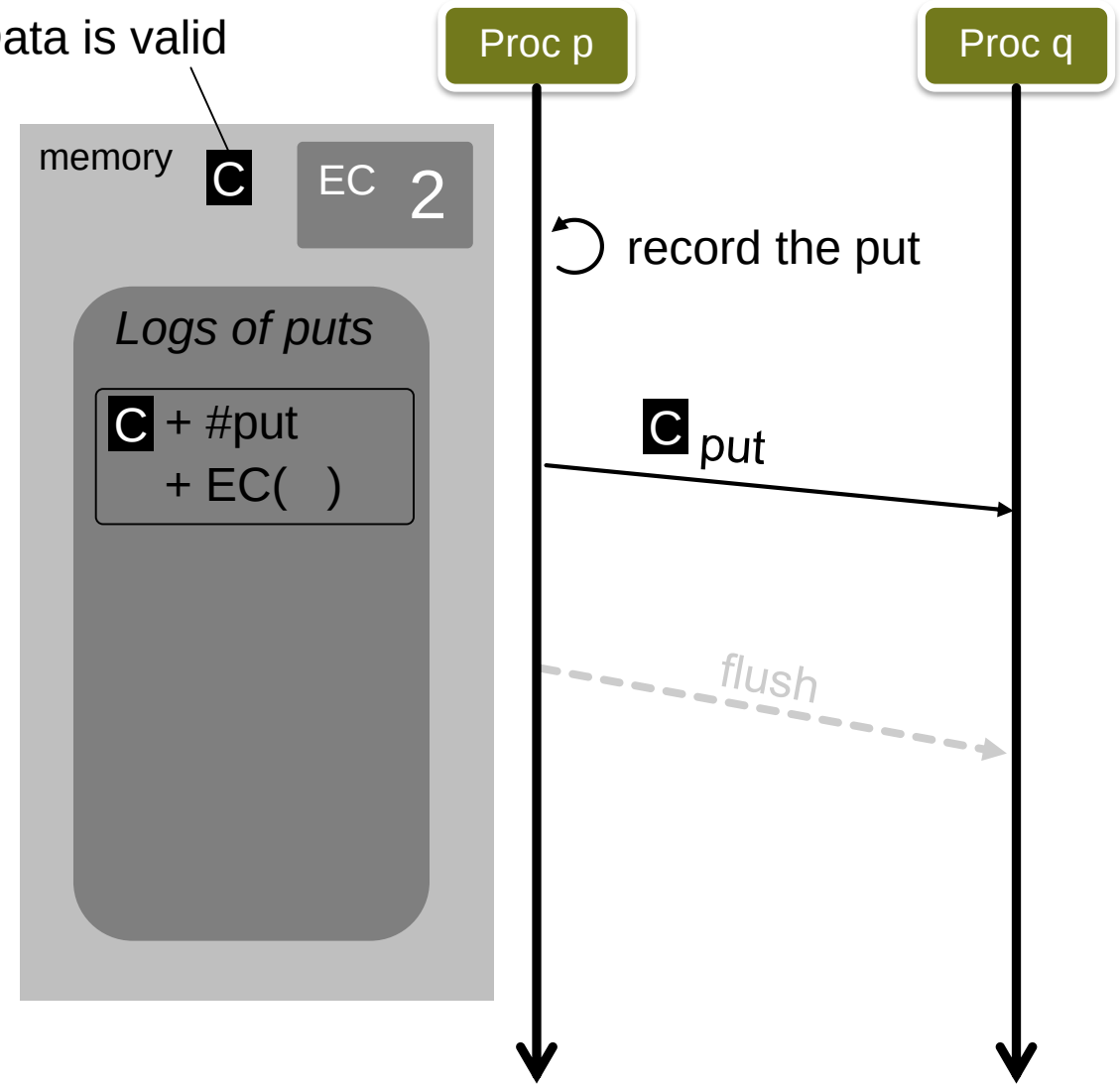
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



To replay the actions preserving $\xrightarrow{co}$, we also record **epoch counters**

RMA: Logging Puts

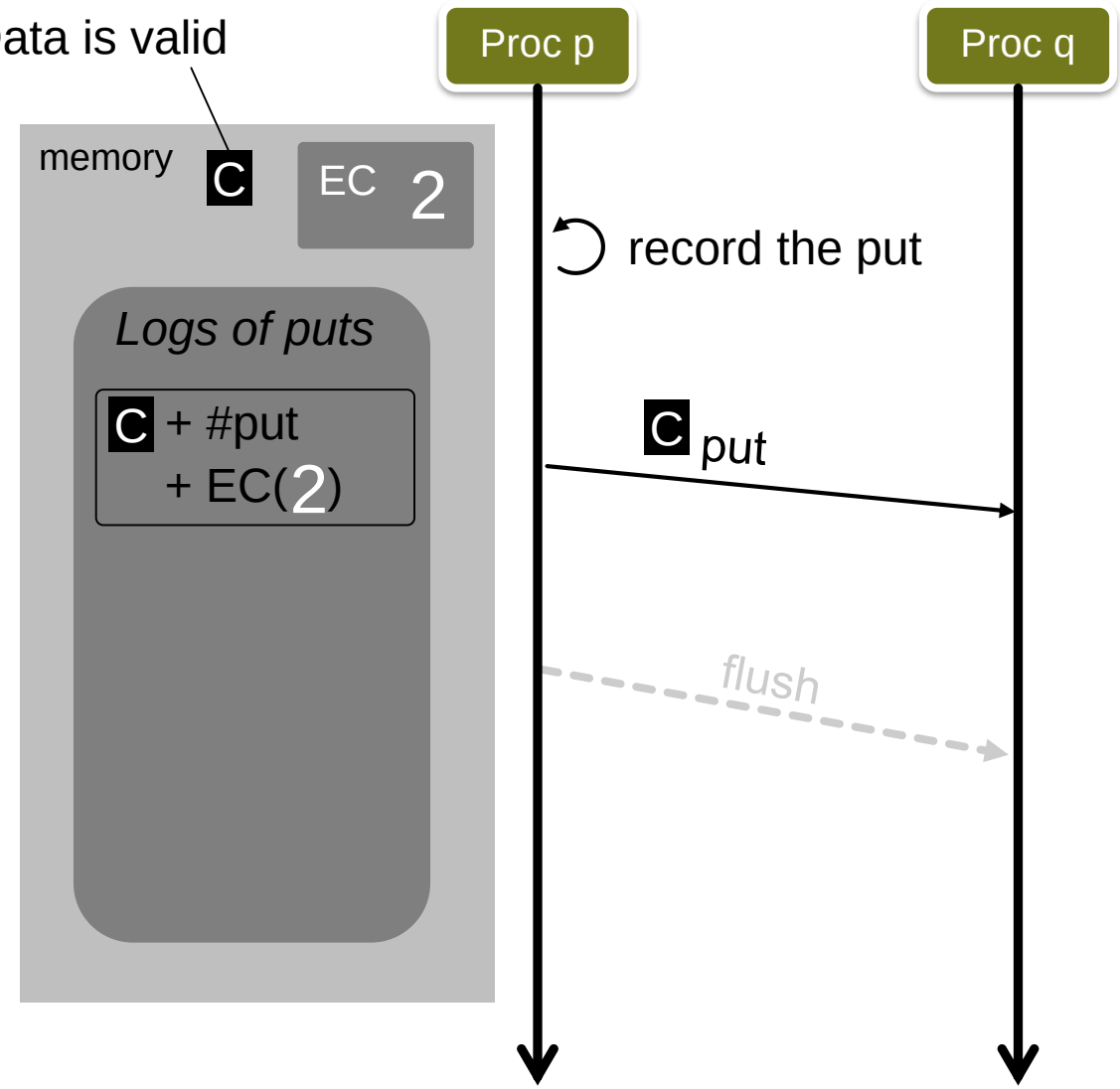
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



To replay the actions preserving $\xrightarrow{co}$, we also record **epoch counters**

RMA: Logging Puts

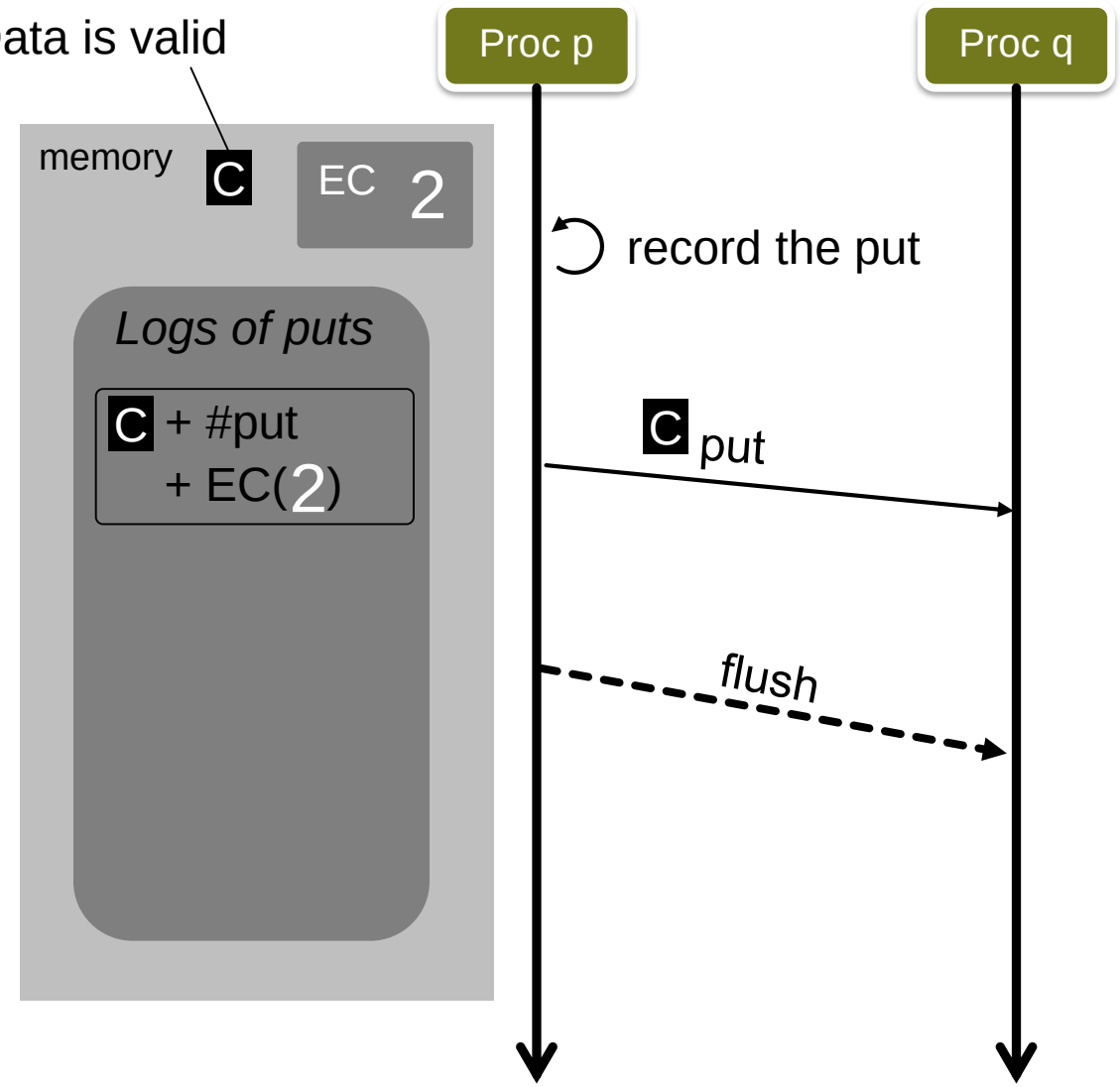
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



To replay the actions preserving $\xrightarrow{co}$, we also record **epoch counters**

RMA: Logging Puts

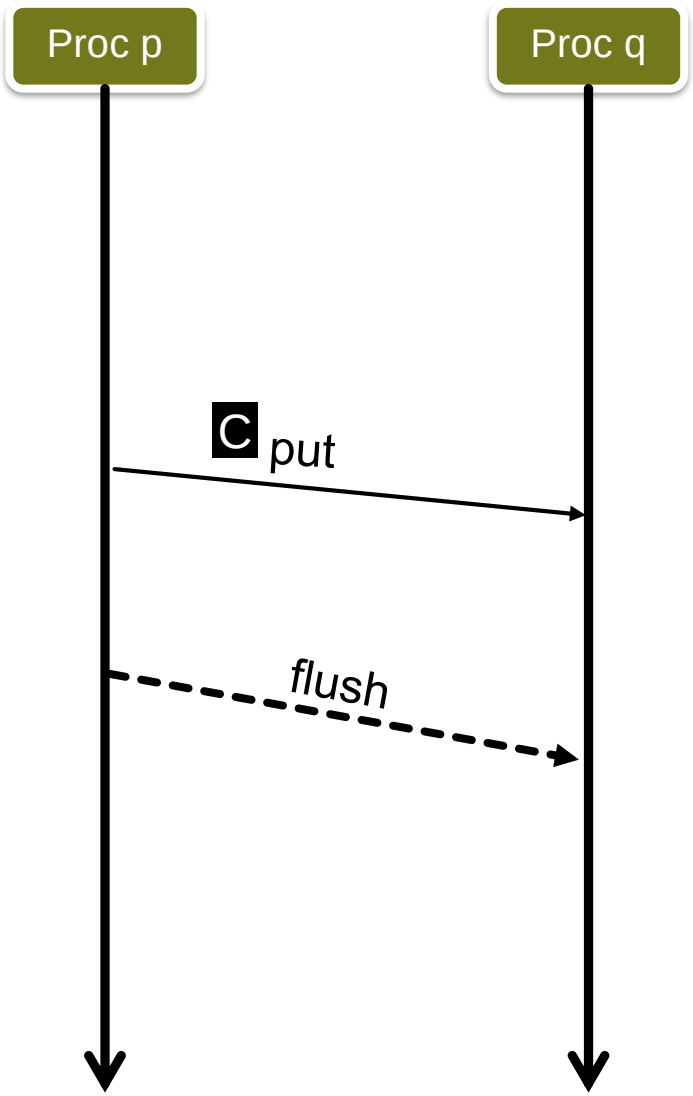
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$



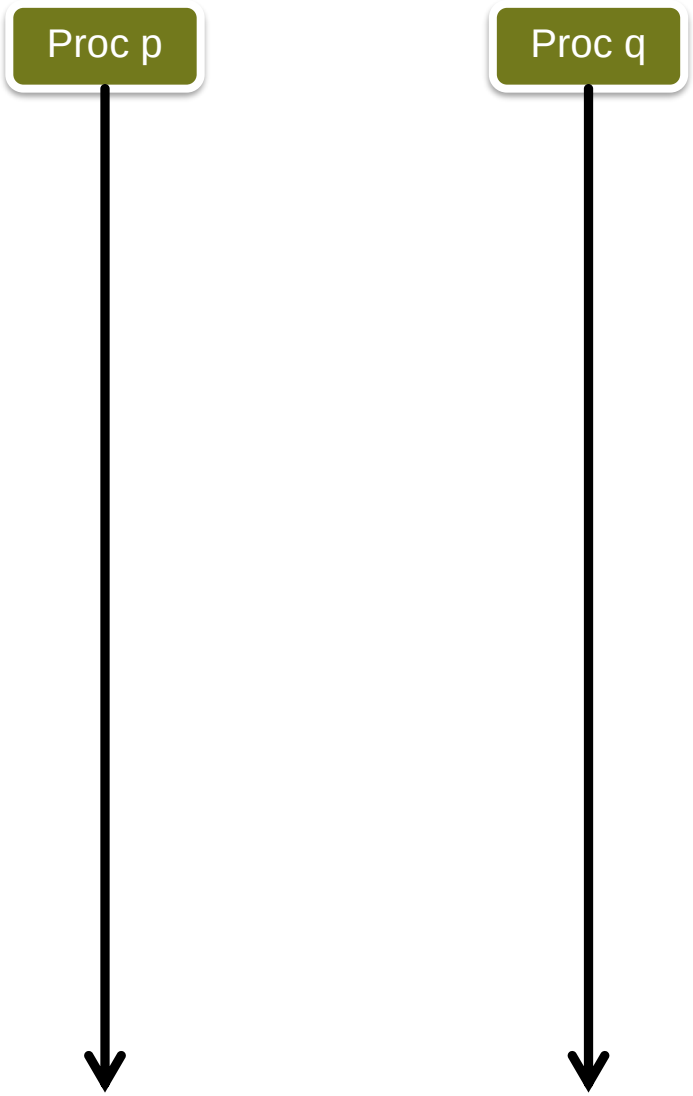
To replay the actions preserving $\xrightarrow{co}$, we also record **epoch counters**

RMA: Logging Puts

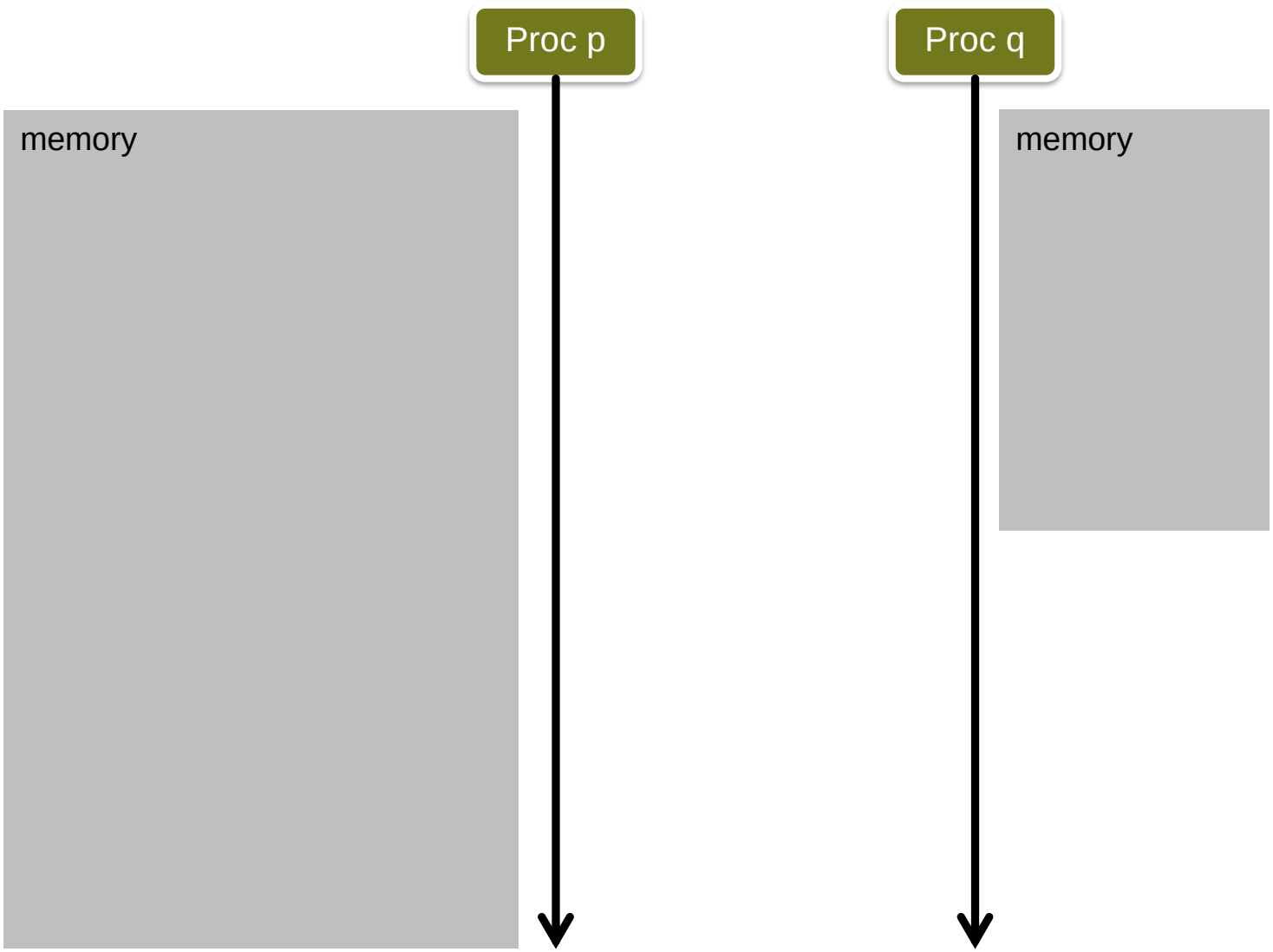
$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



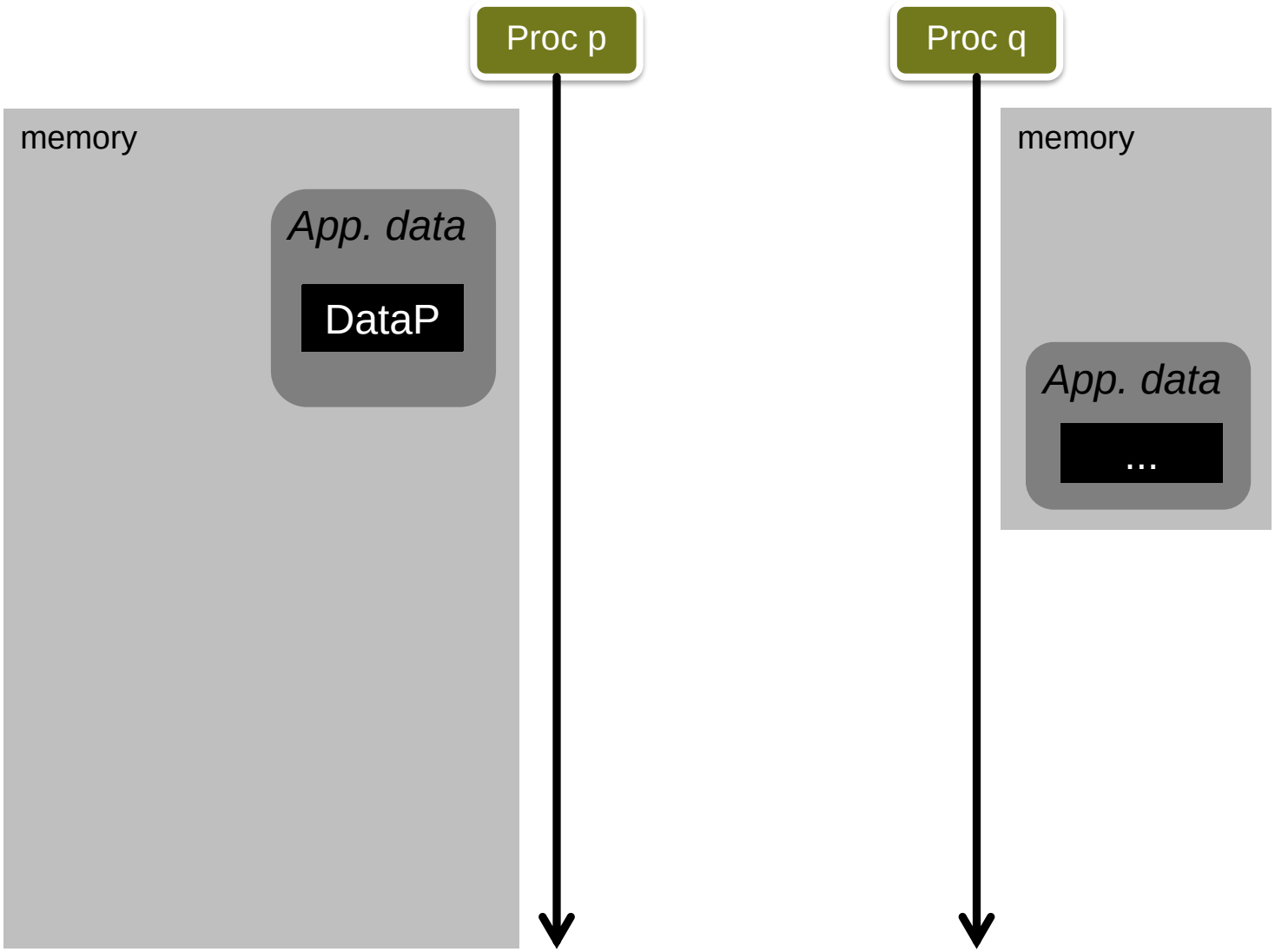
RMA: CHECKPOINTING



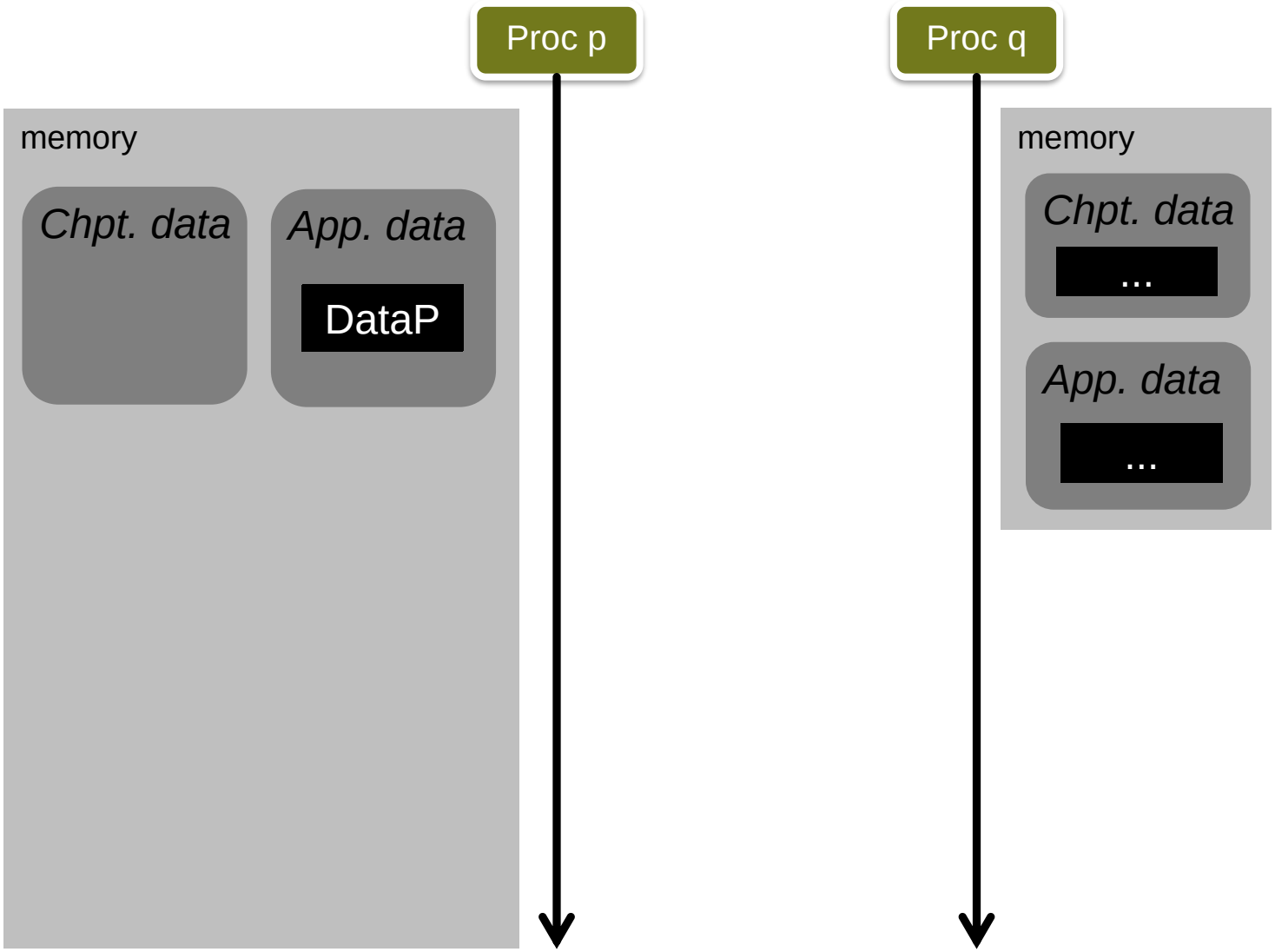
RMA: CHECKPOINTING



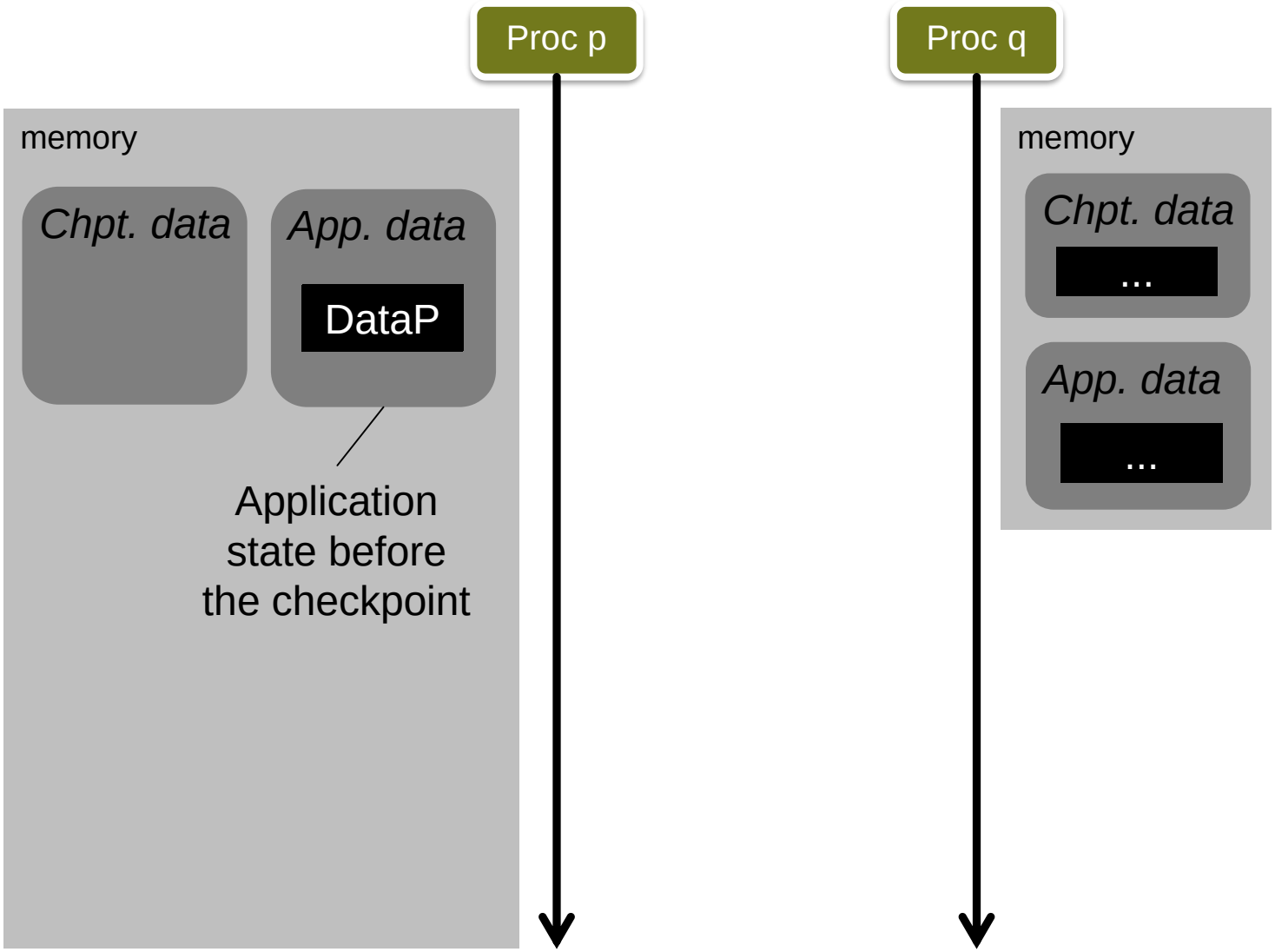
RMA: CHECKPOINTING



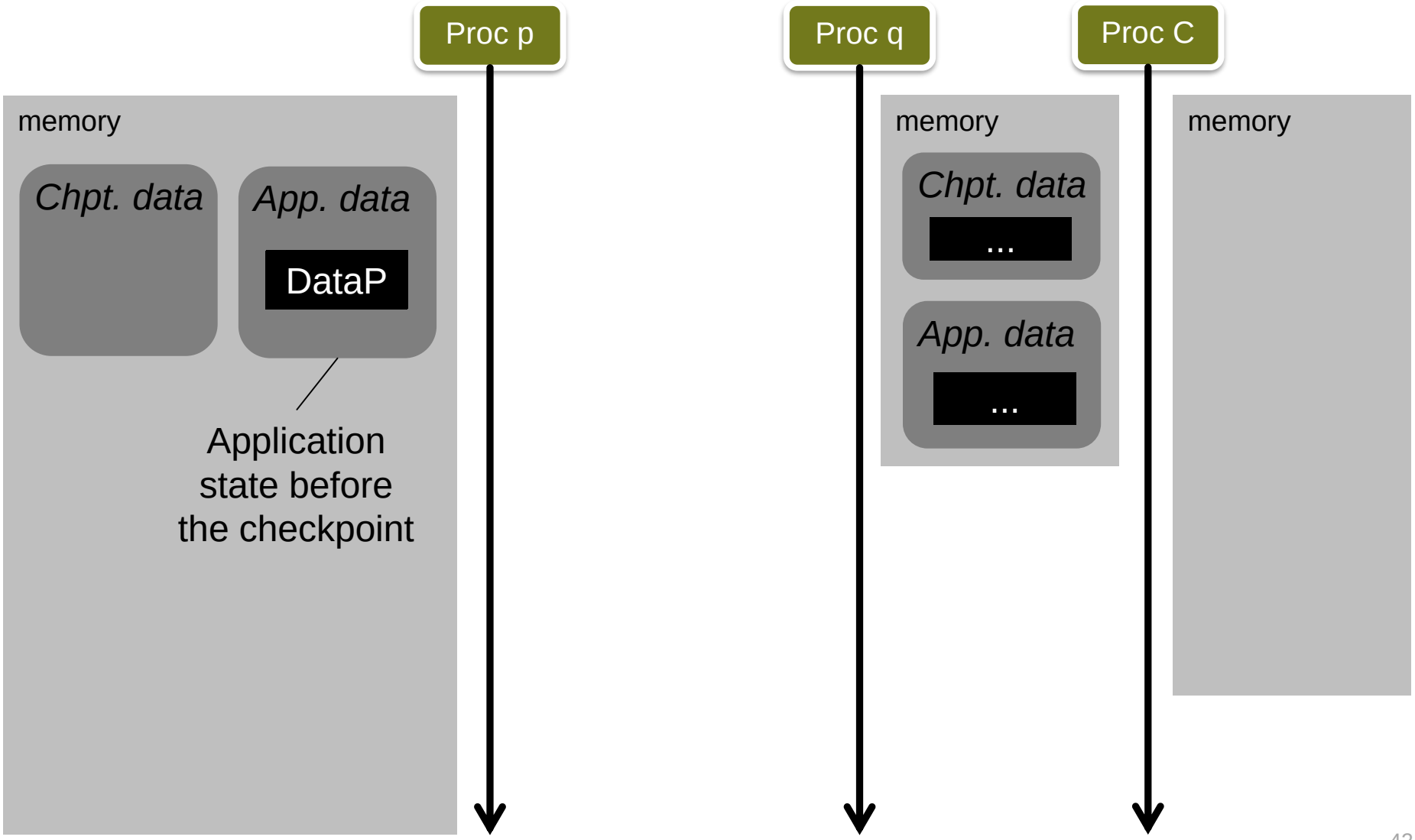
RMA: CHECKPOINTING



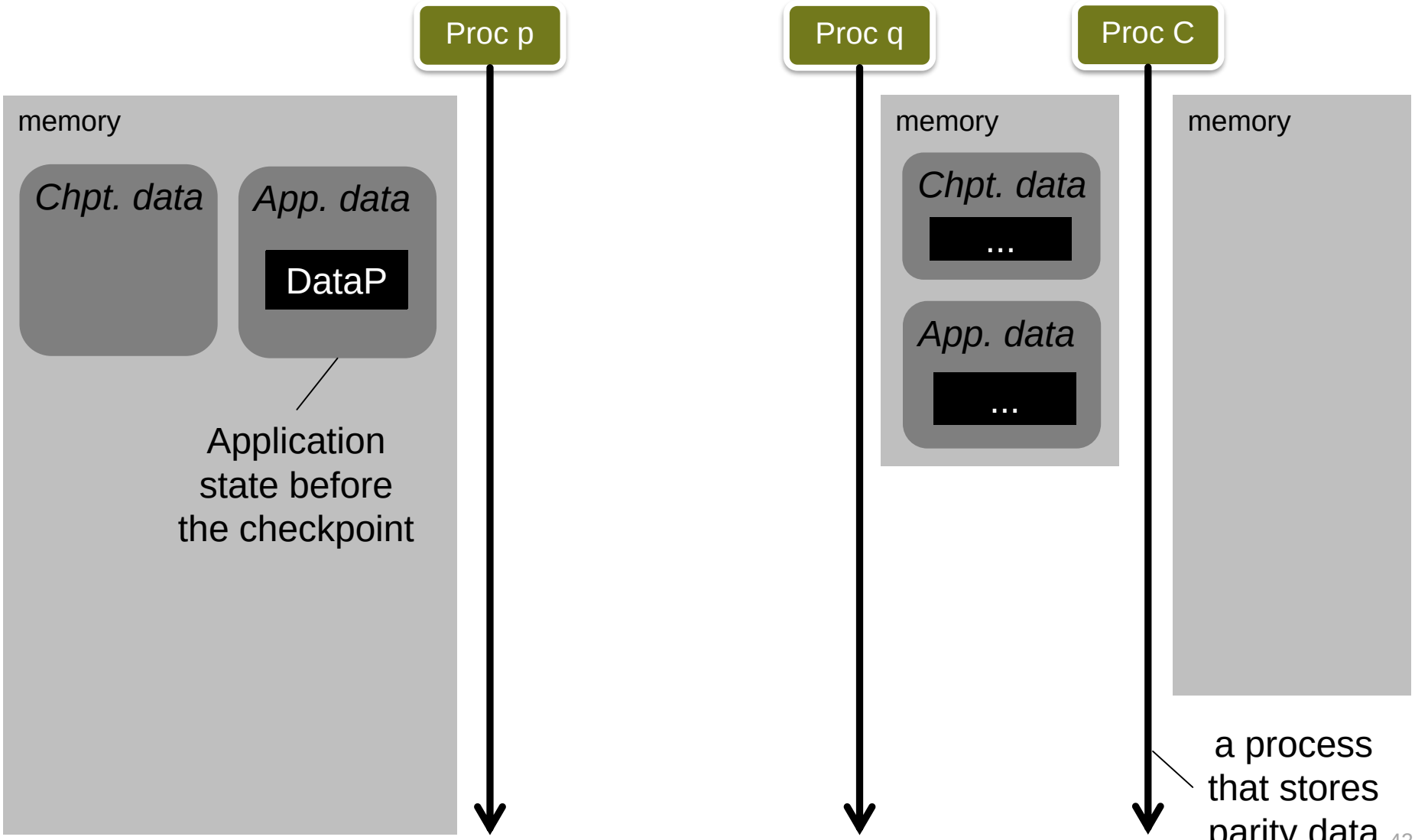
RMA: CHECKPOINTING



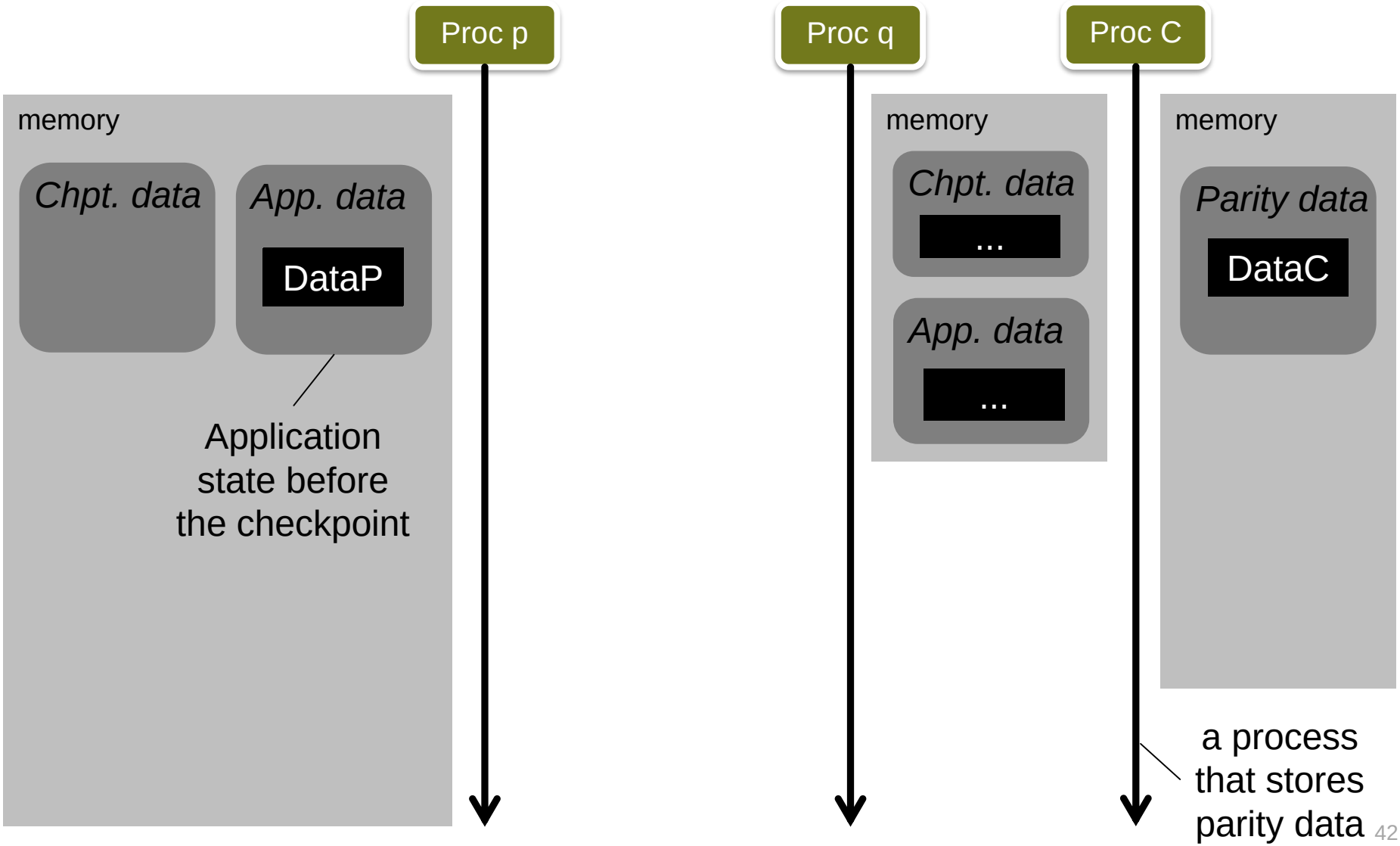
RMA: CHECKPOINTING



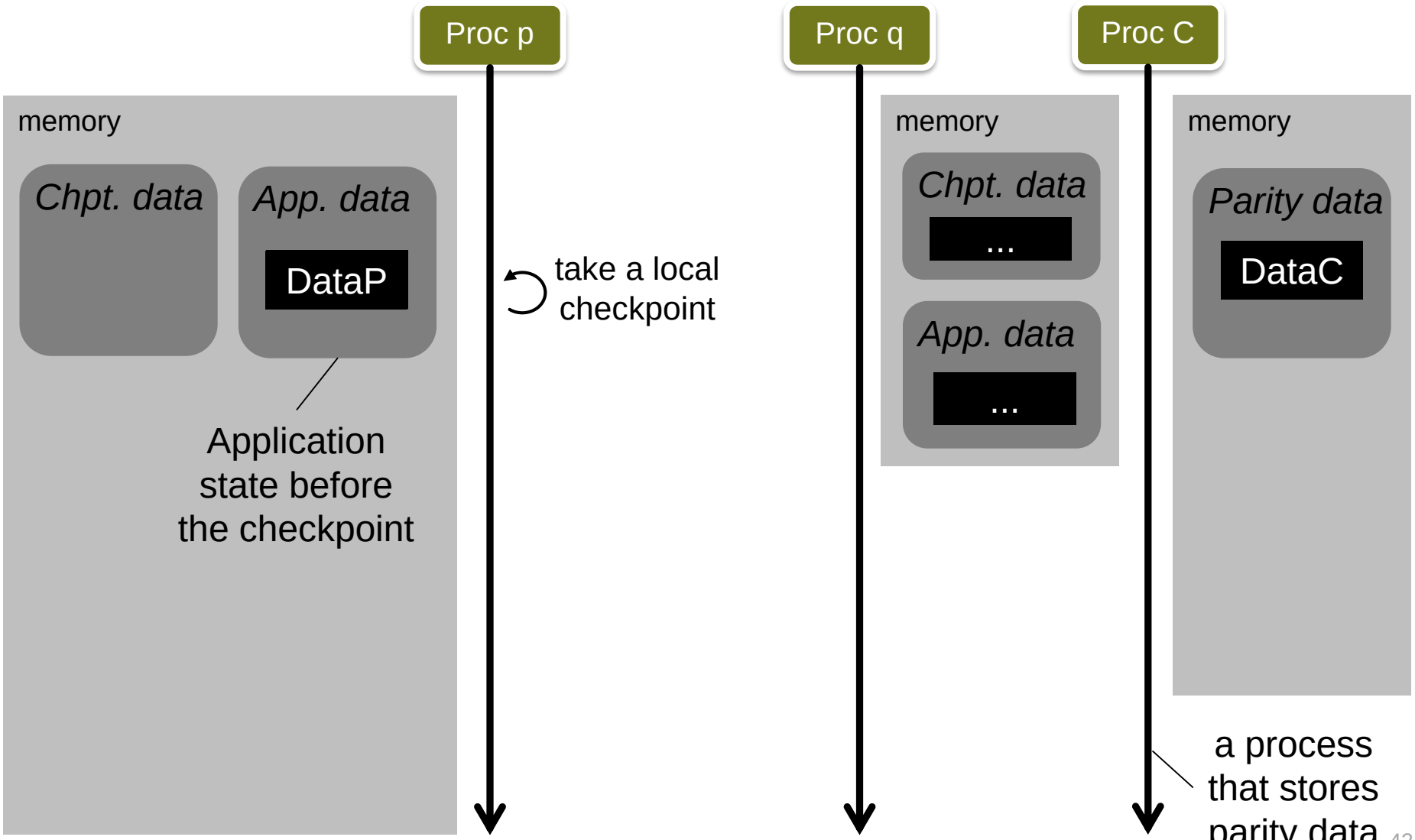
RMA: CHECKPOINTING



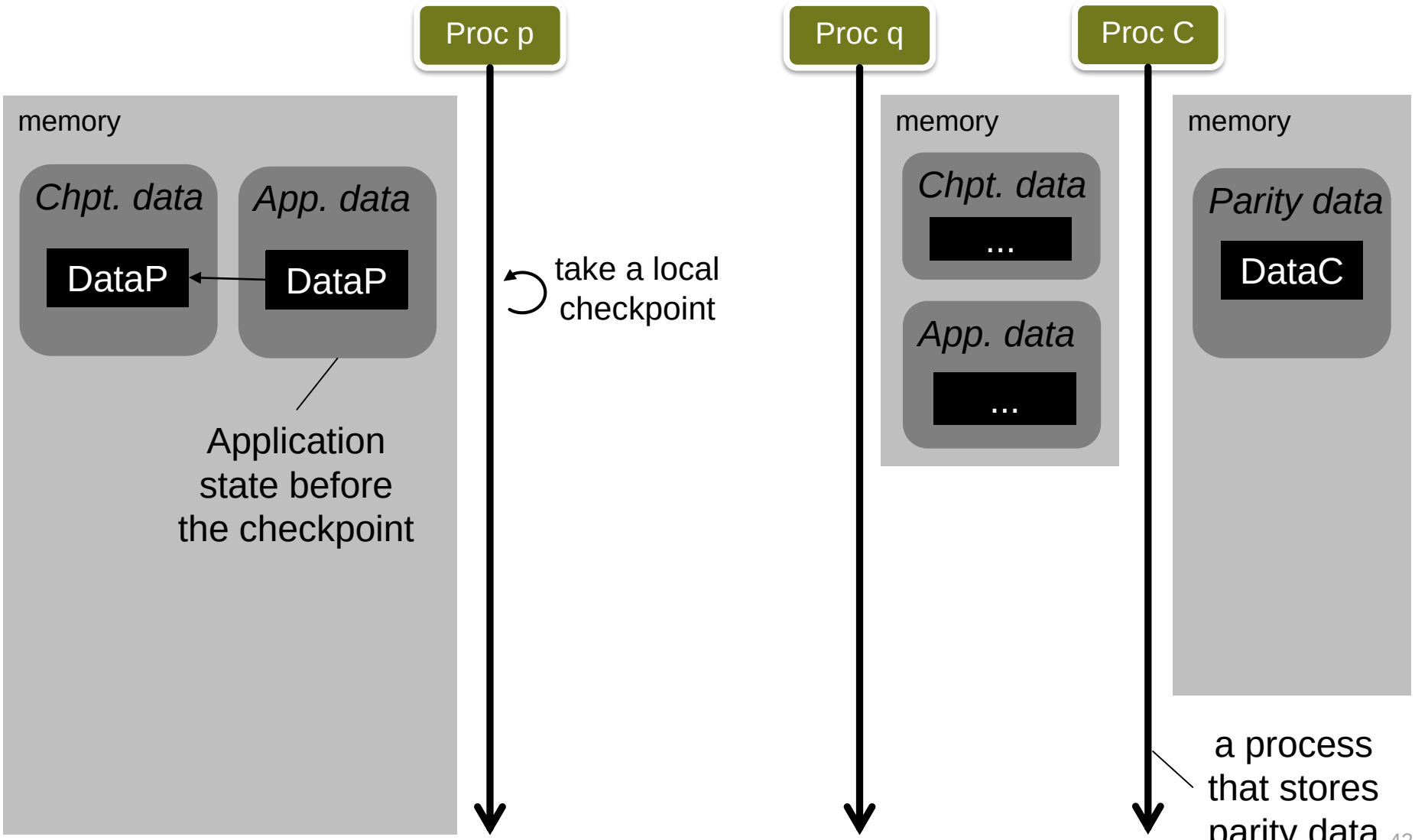
RMA: CHECKPOINTING



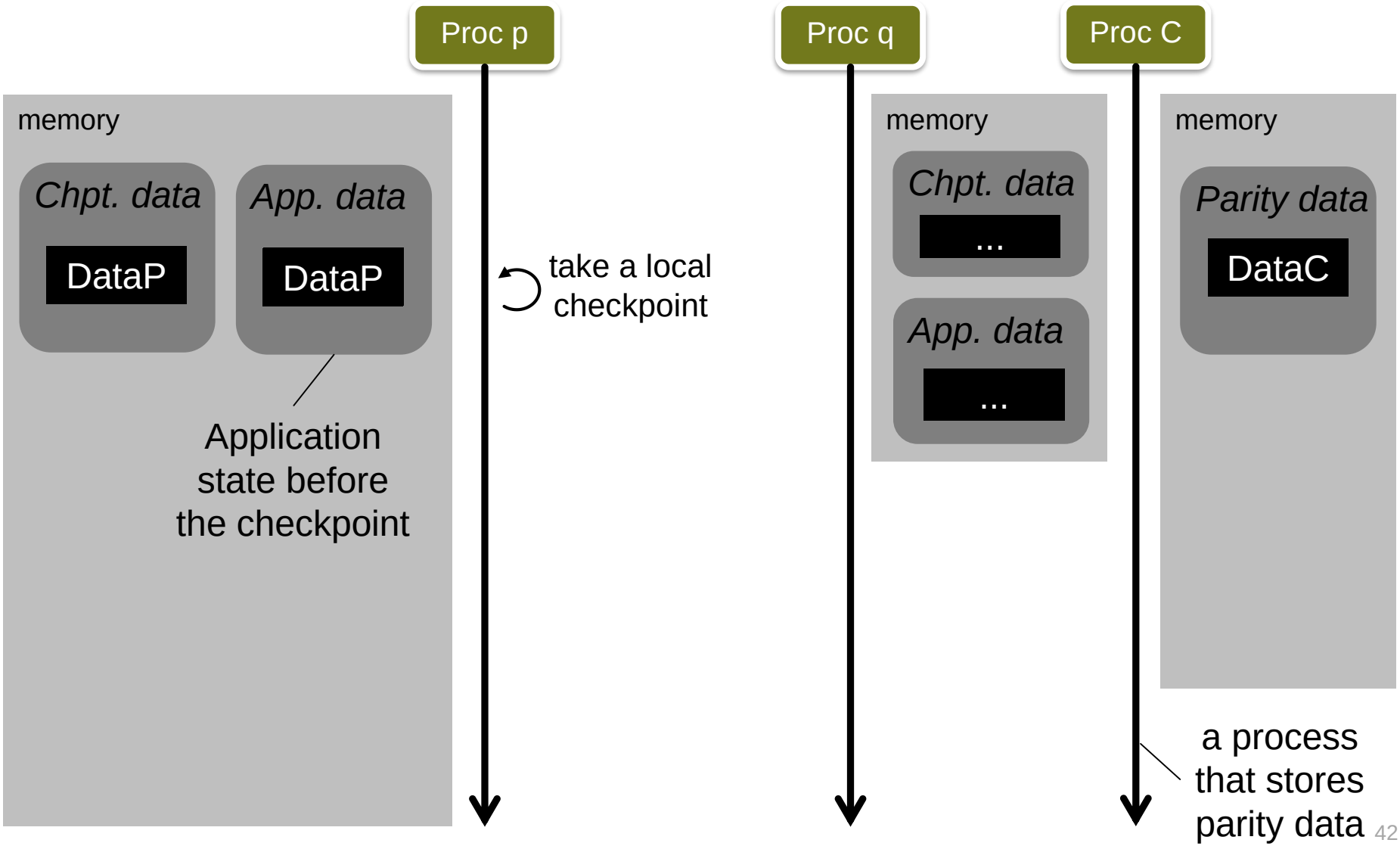
RMA: CHECKPOINTING



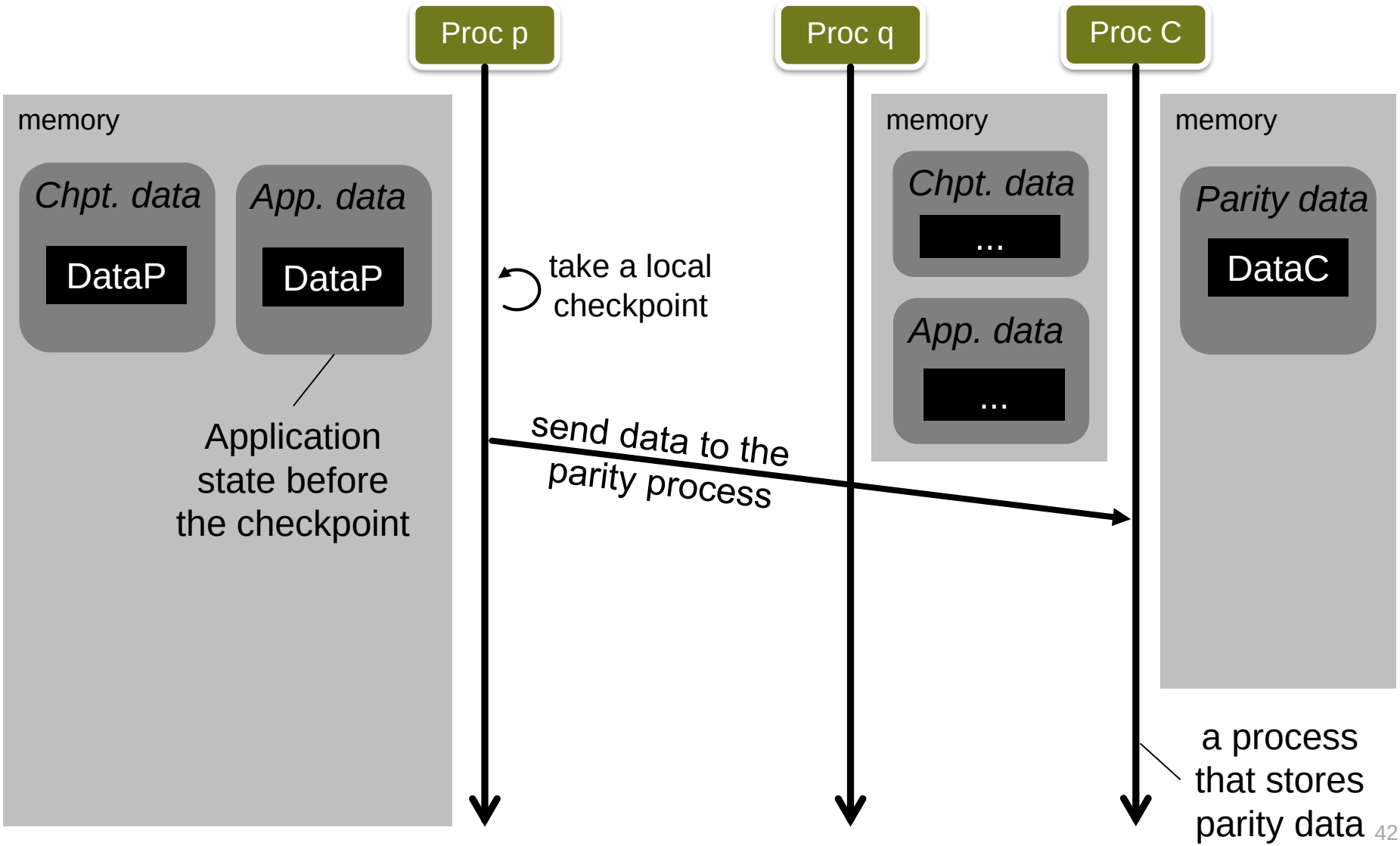
RMA: CHECKPOINTING



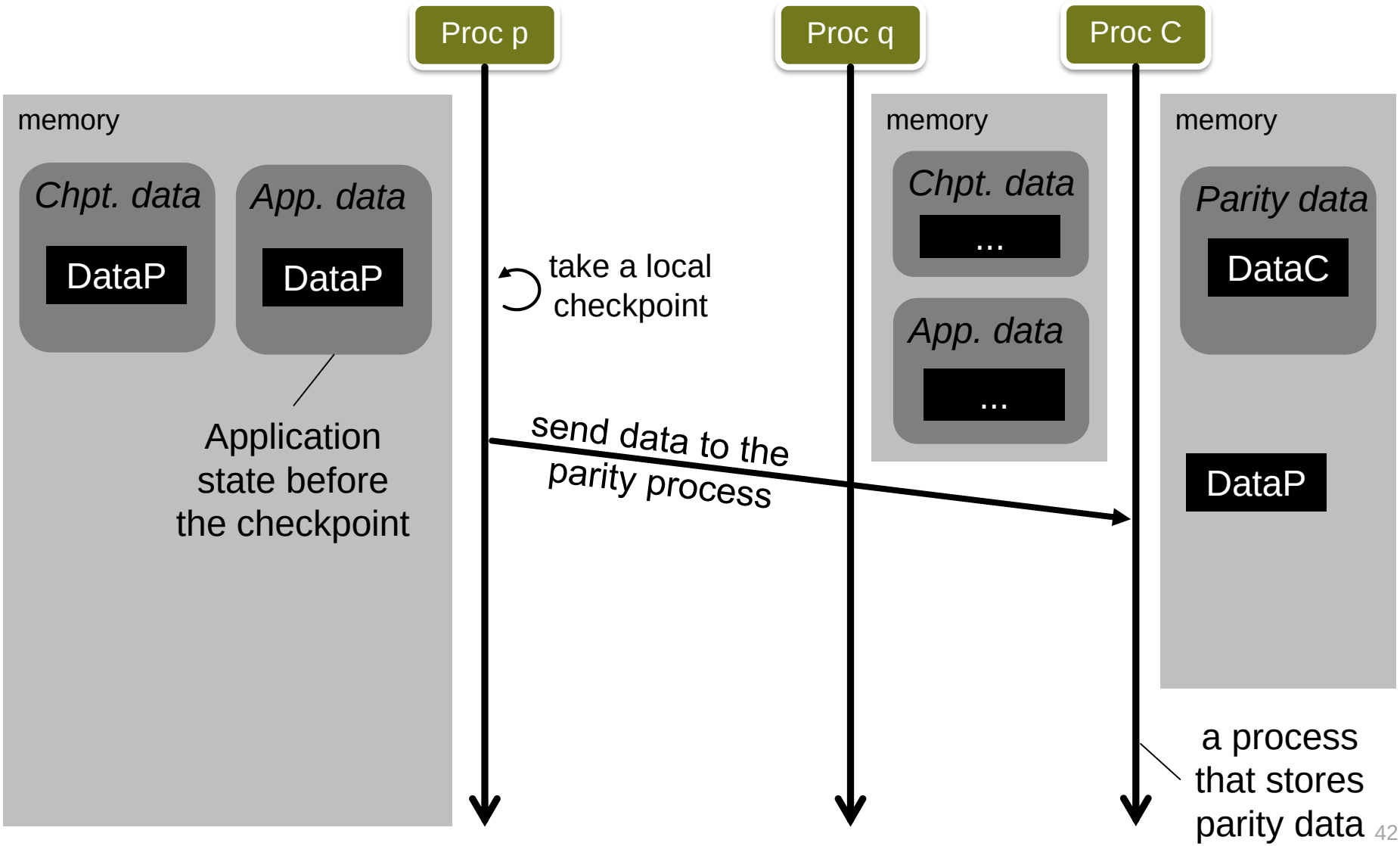
RMA: CHECKPOINTING



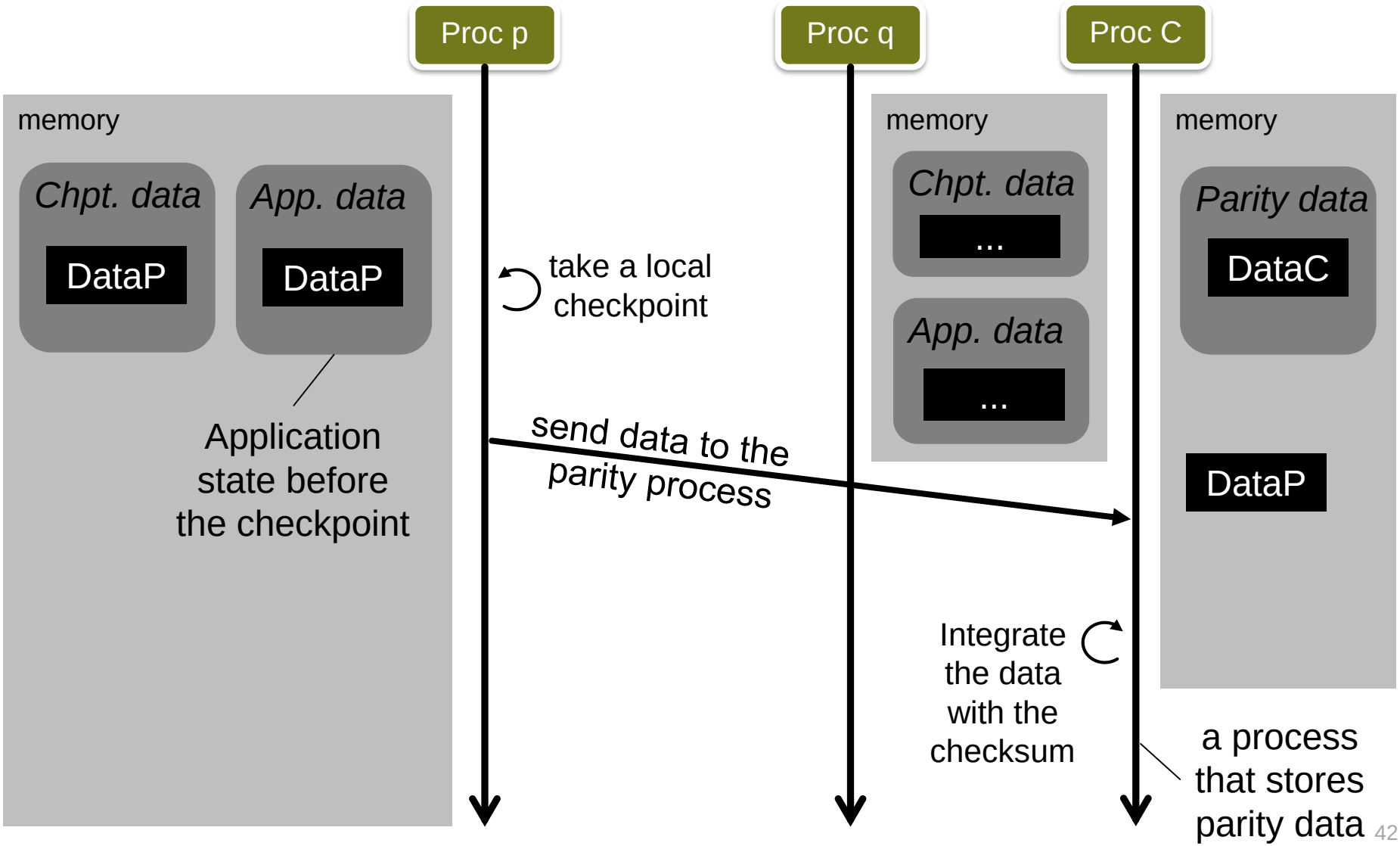
RMA: CHECKPOINTING



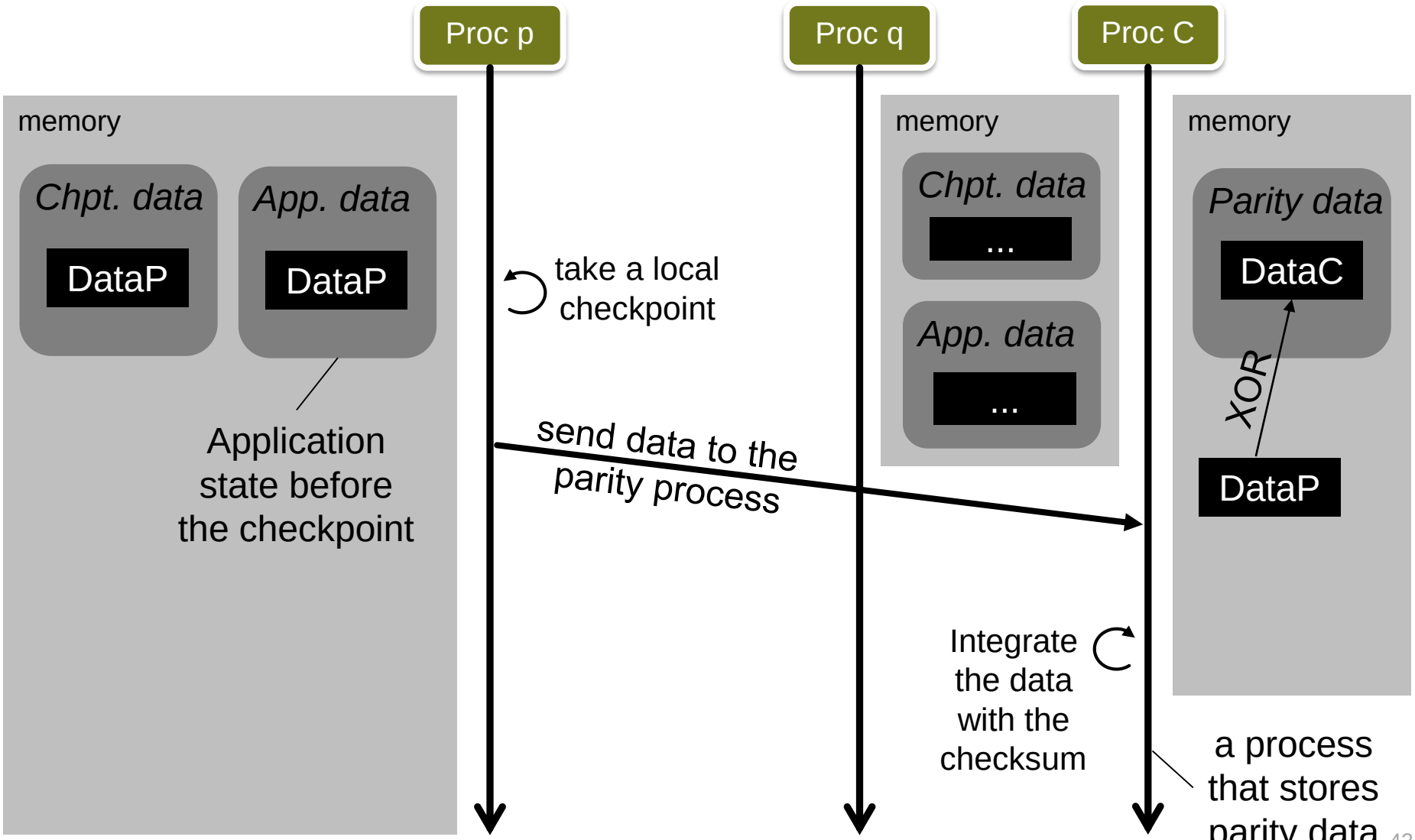
RMA: CHECKPOINTING



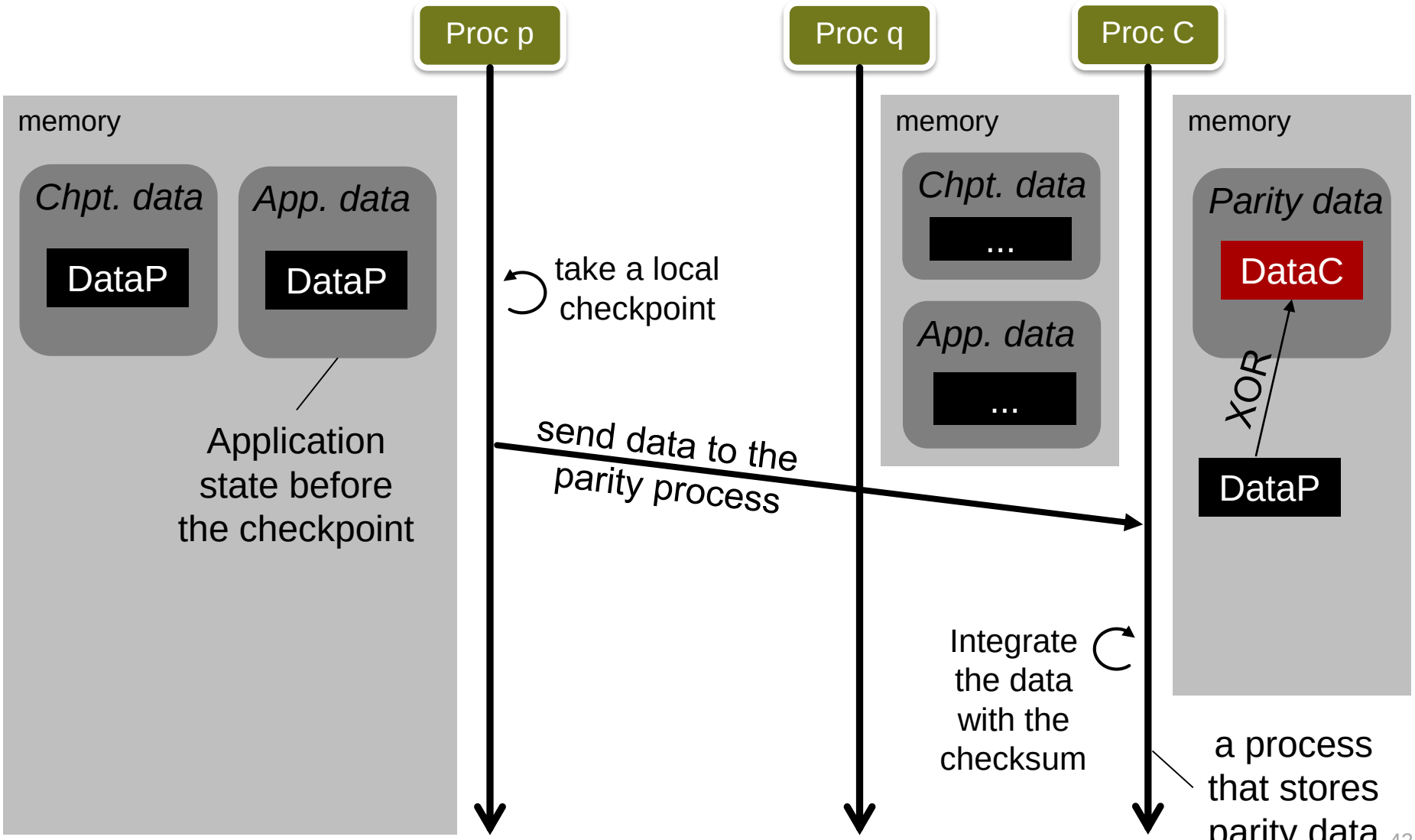
RMA: CHECKPOINTING



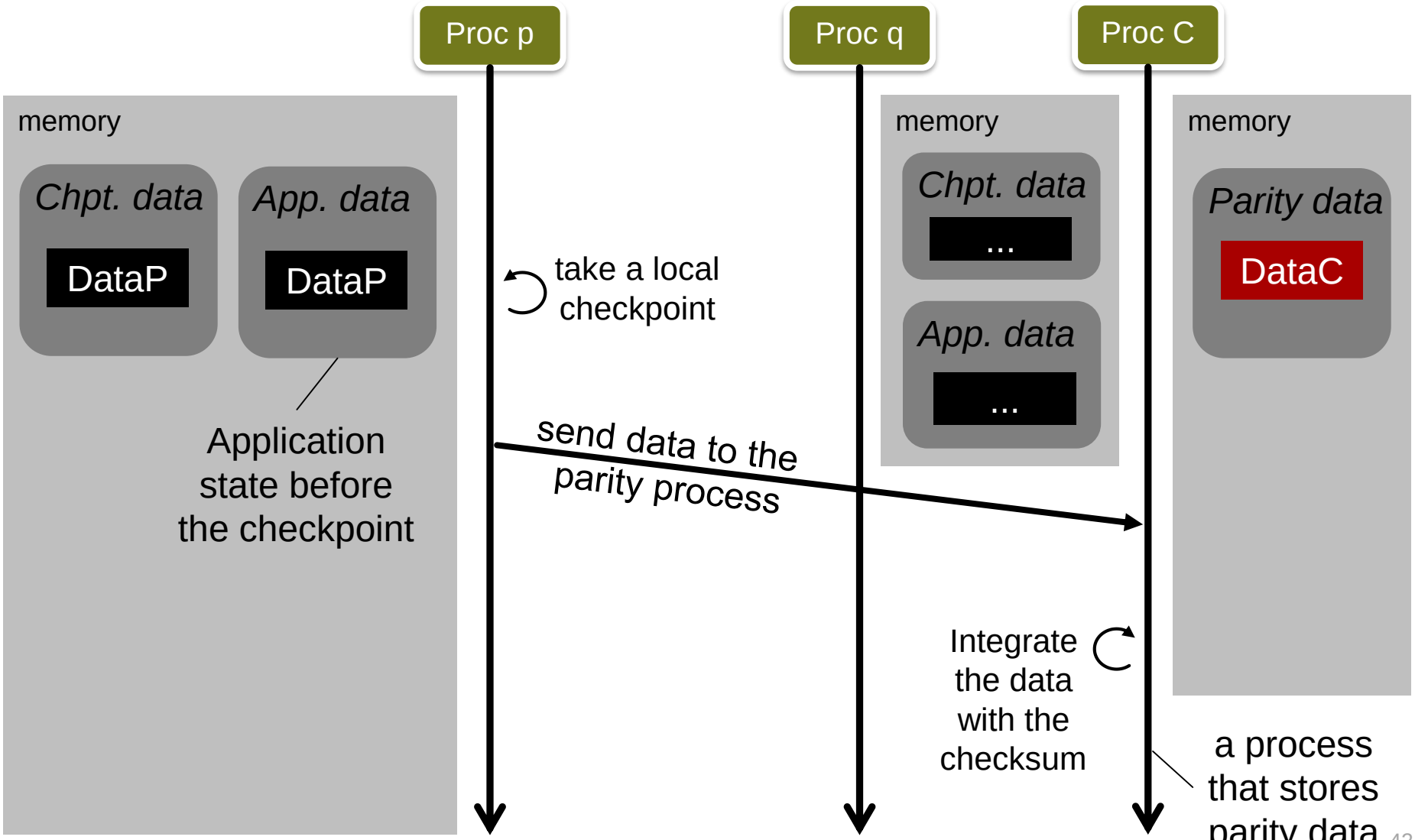
RMA: CHECKPOINTING



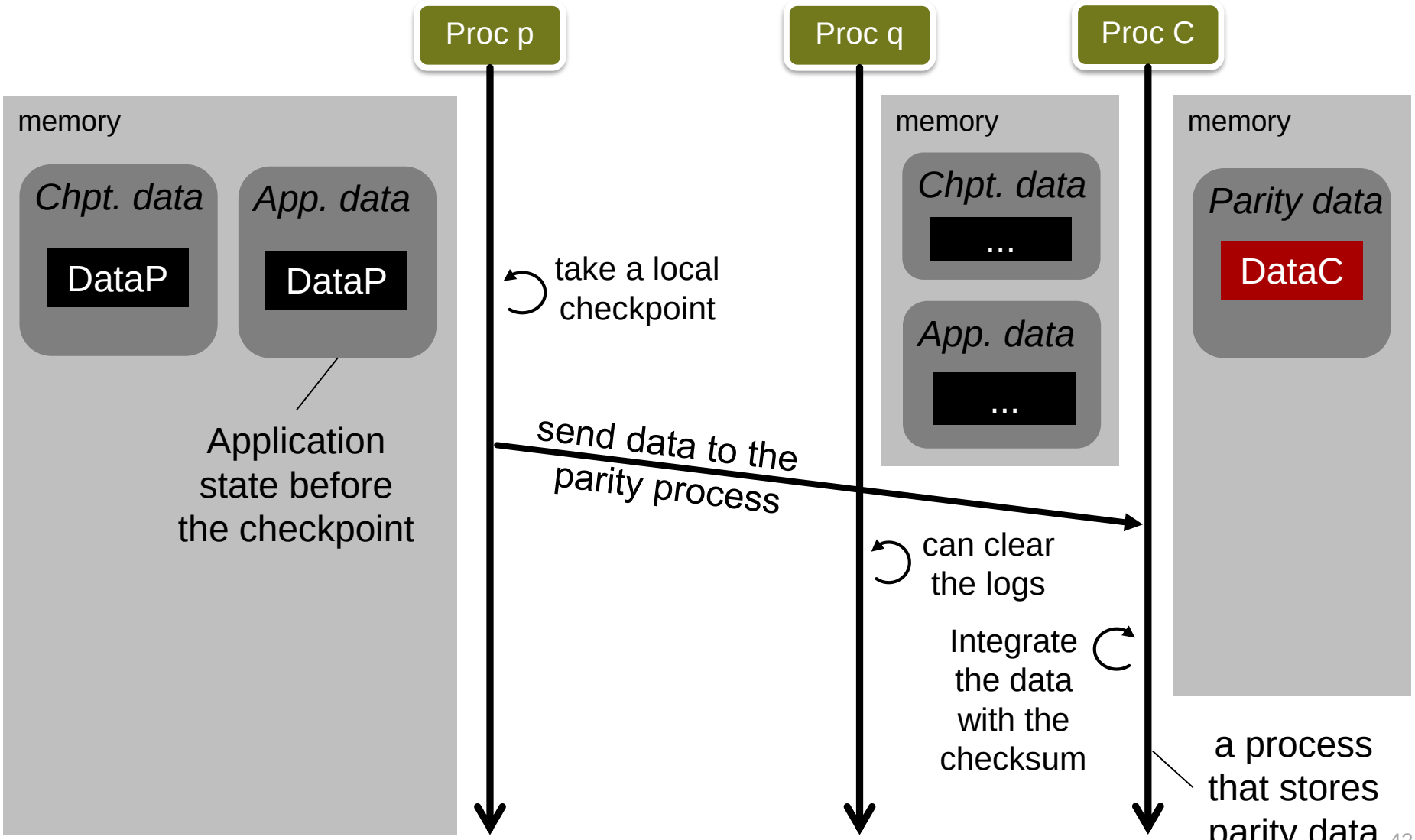
RMA: CHECKPOINTING



RMA: CHECKPOINTING

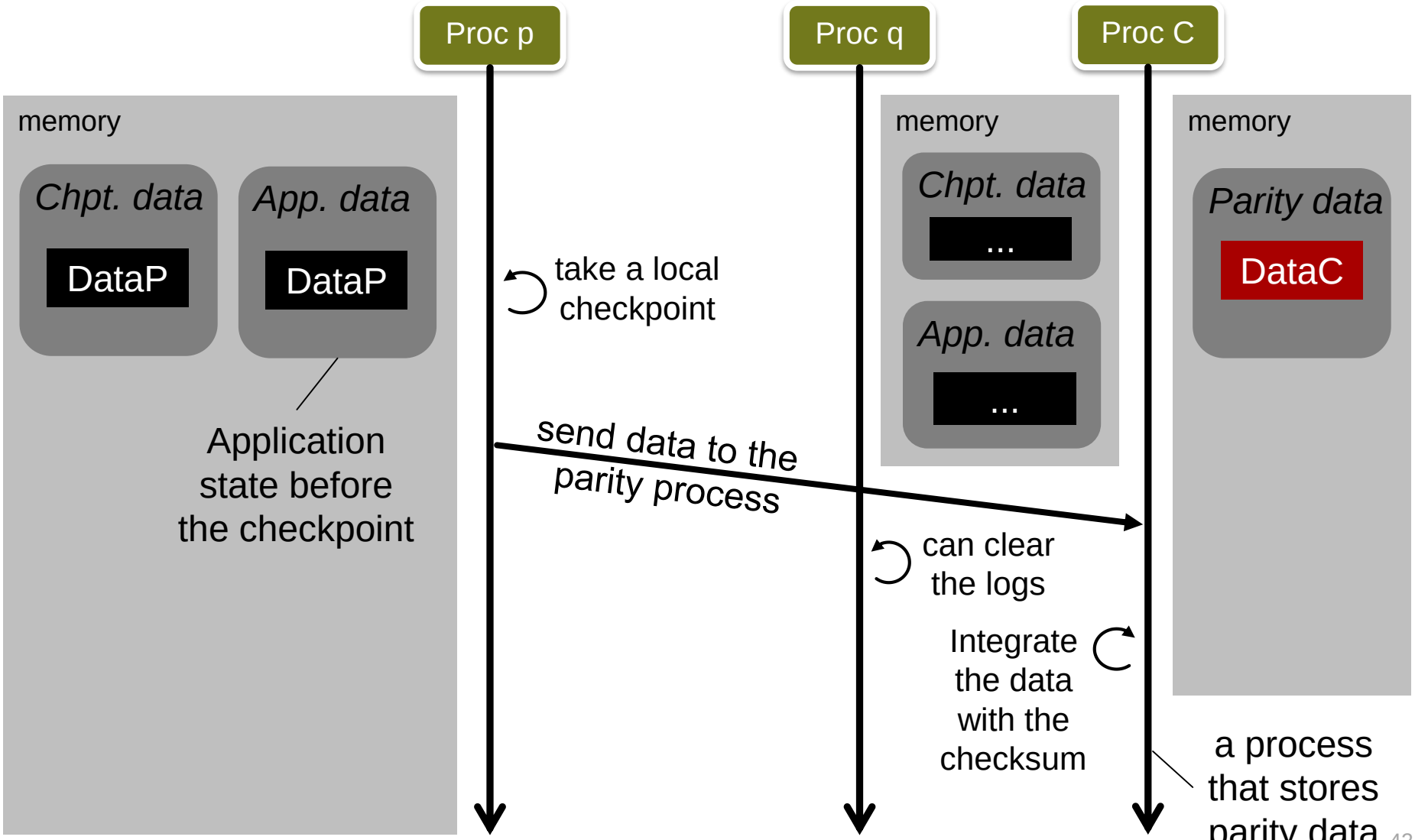


RMA: CHECKPOINTING



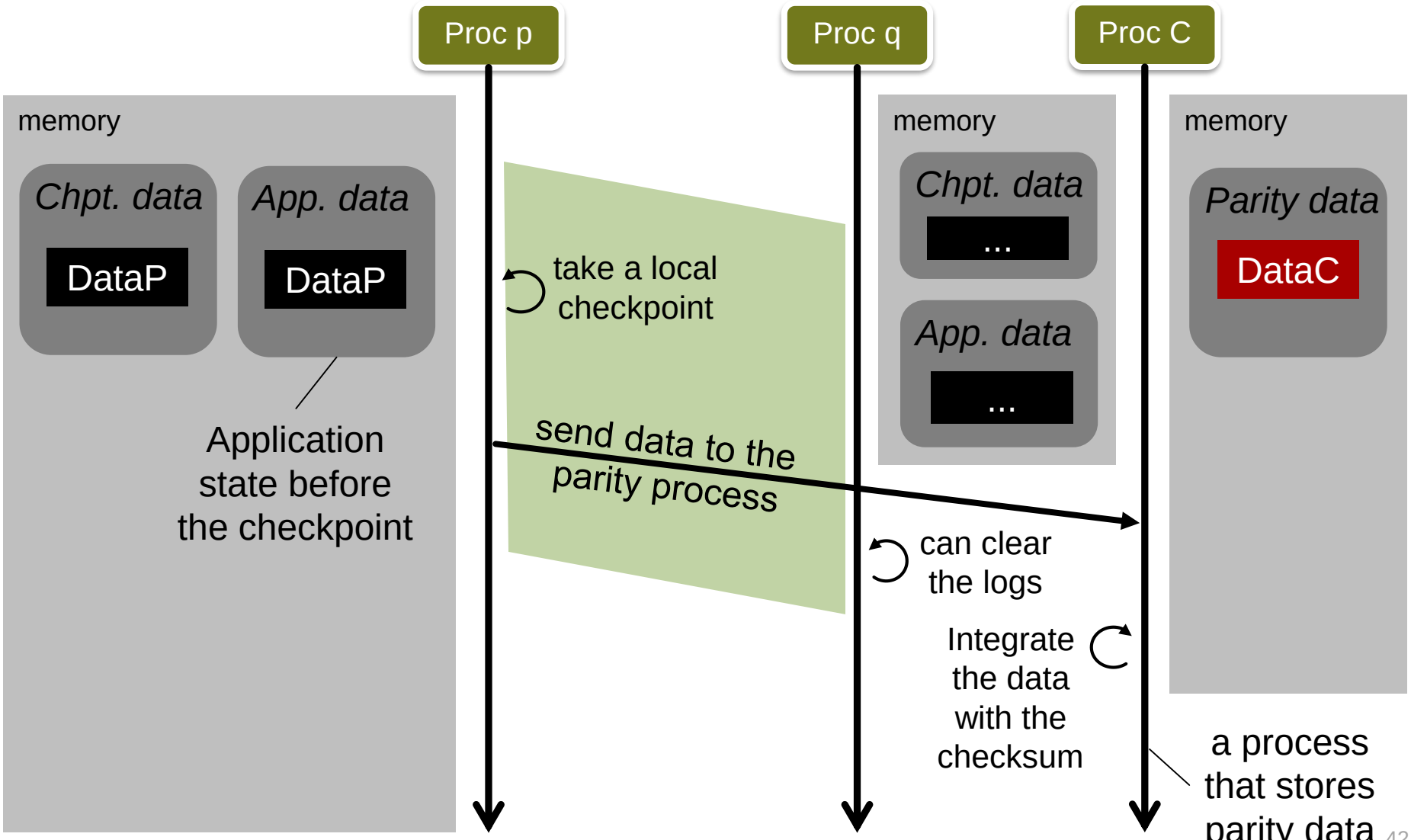
RMA: CHECKPOINTING

The Epoch Condition



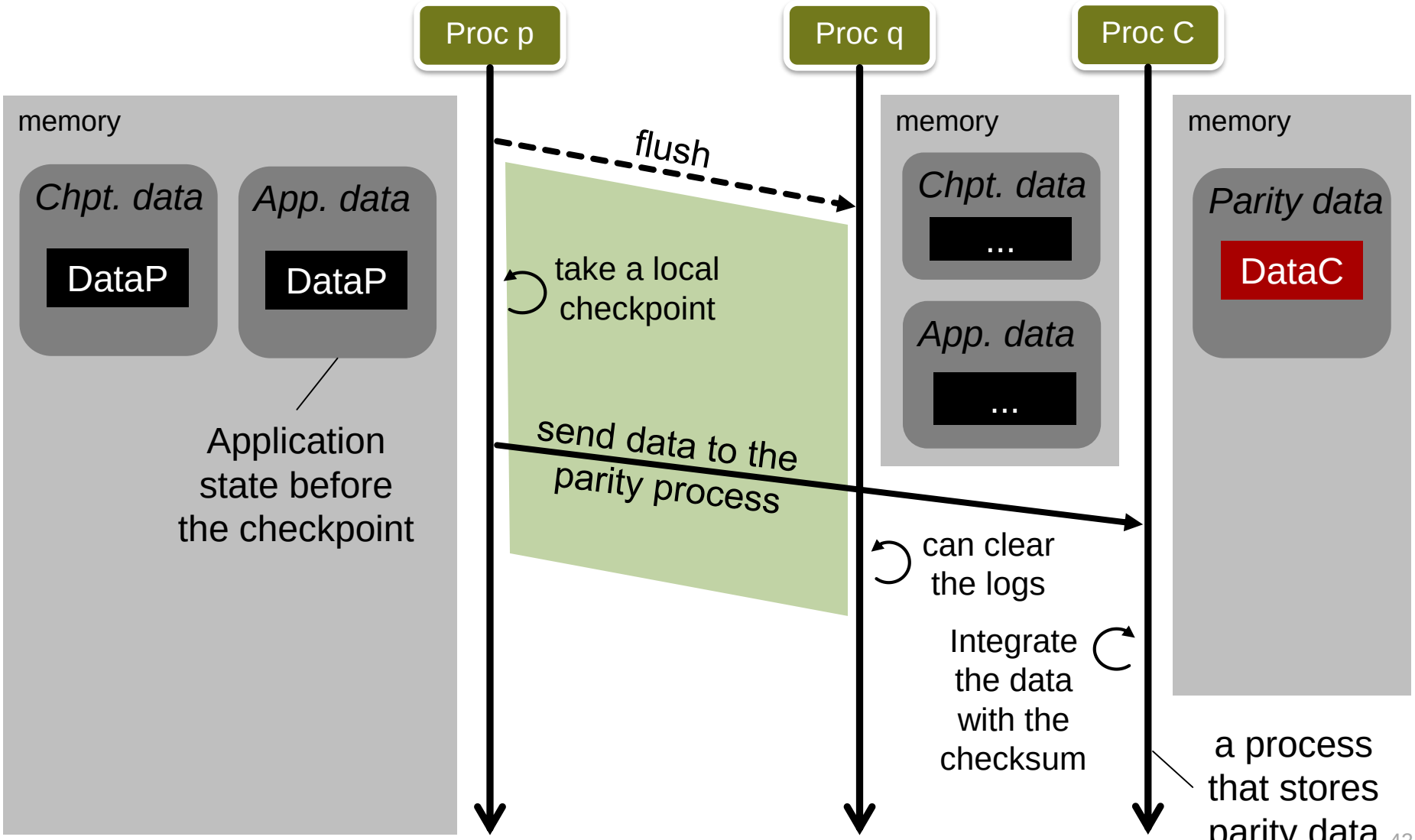
RMA: CHECKPOINTING

The Epoch Condition



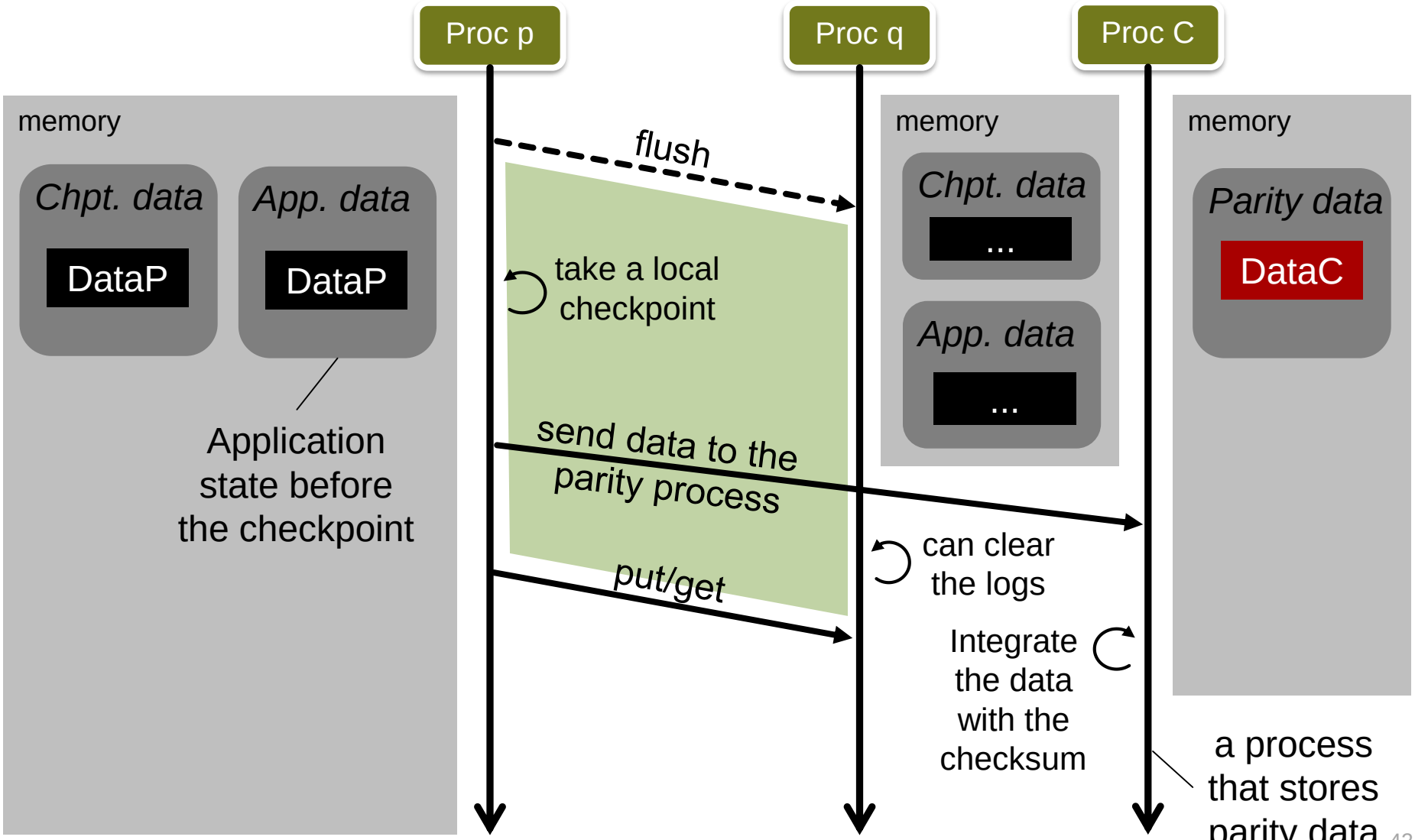
RMA: CHECKPOINTING

The Epoch Condition



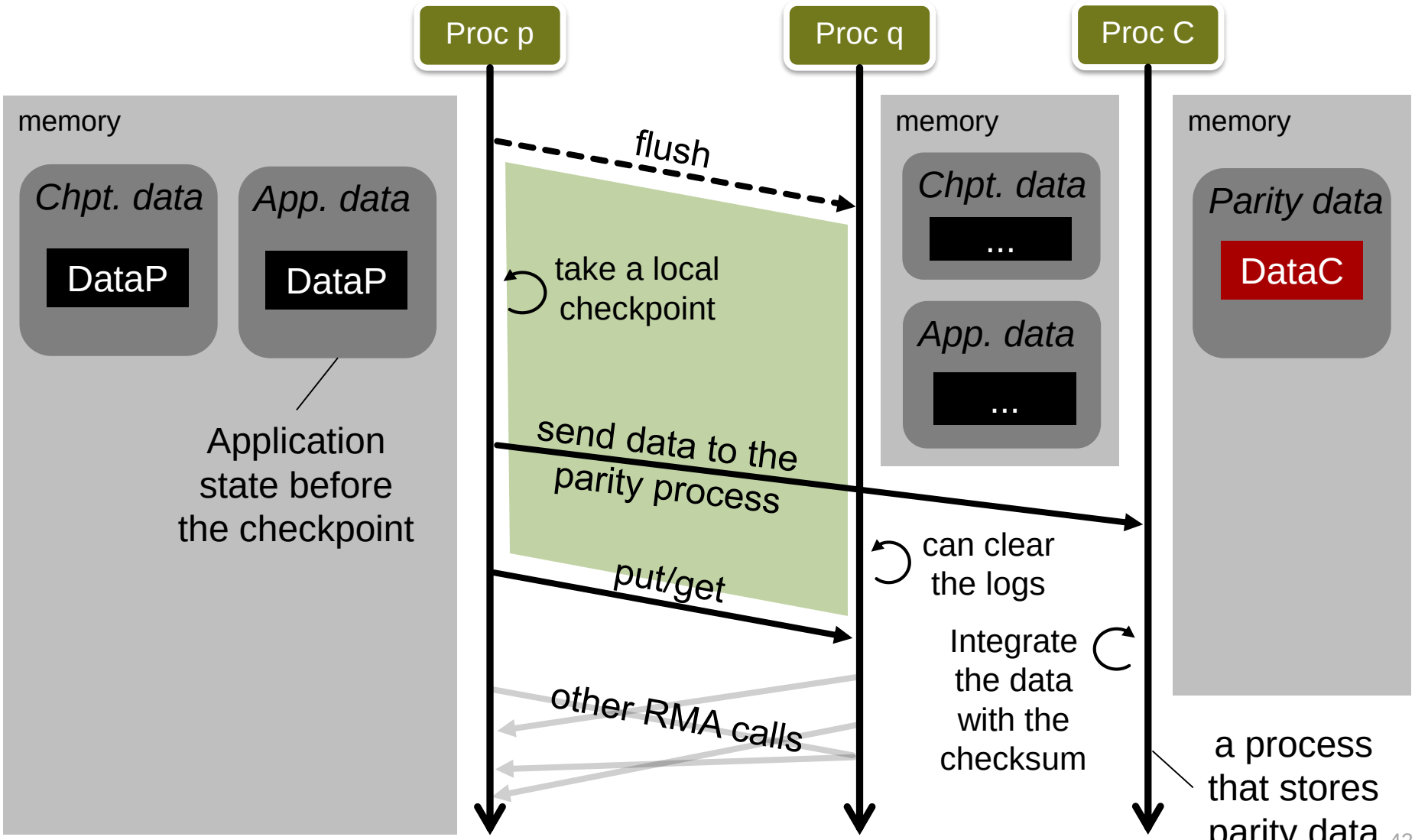
RMA: CHECKPOINTING

The Epoch Condition



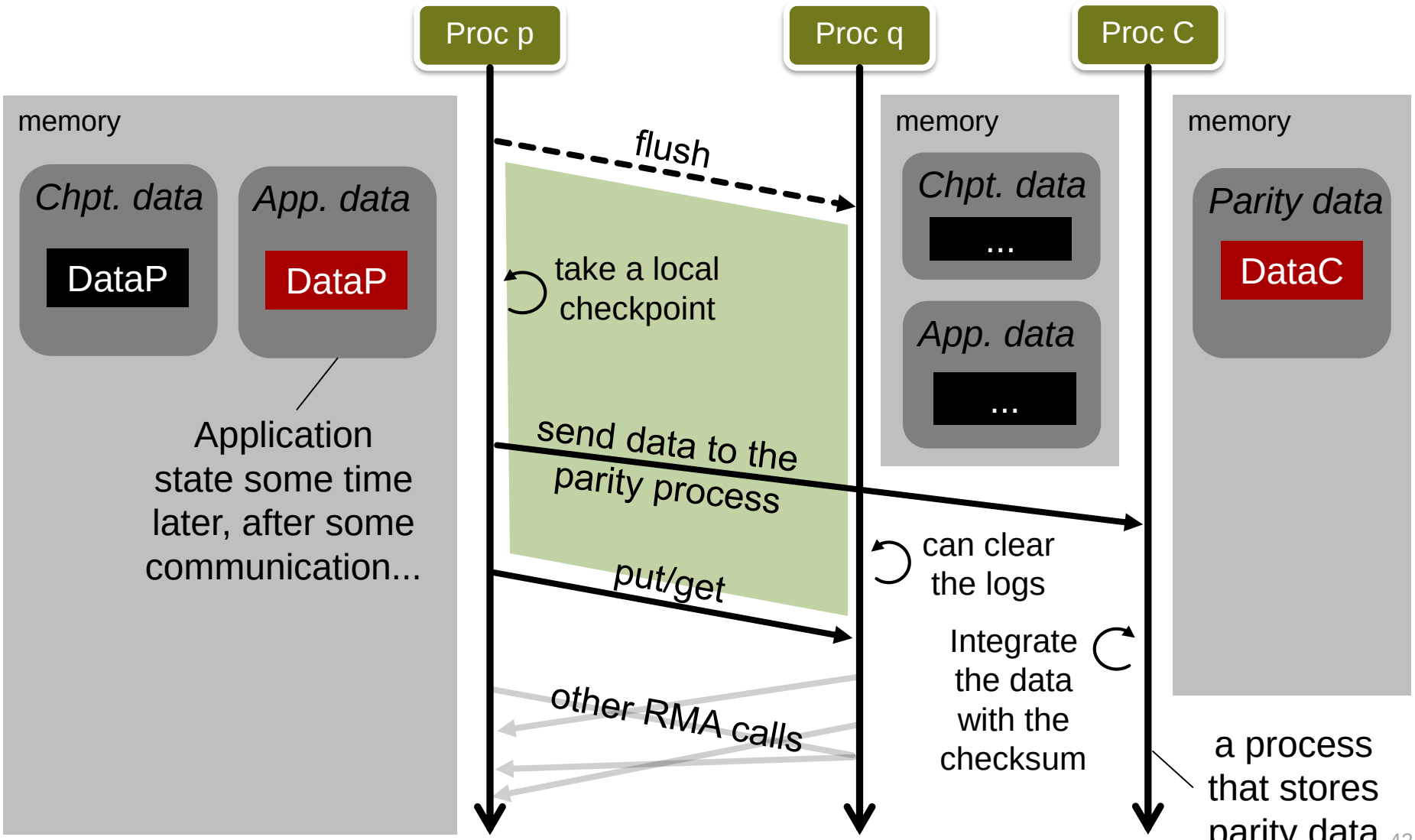
RMA: CHECKPOINTING

The Epoch Condition



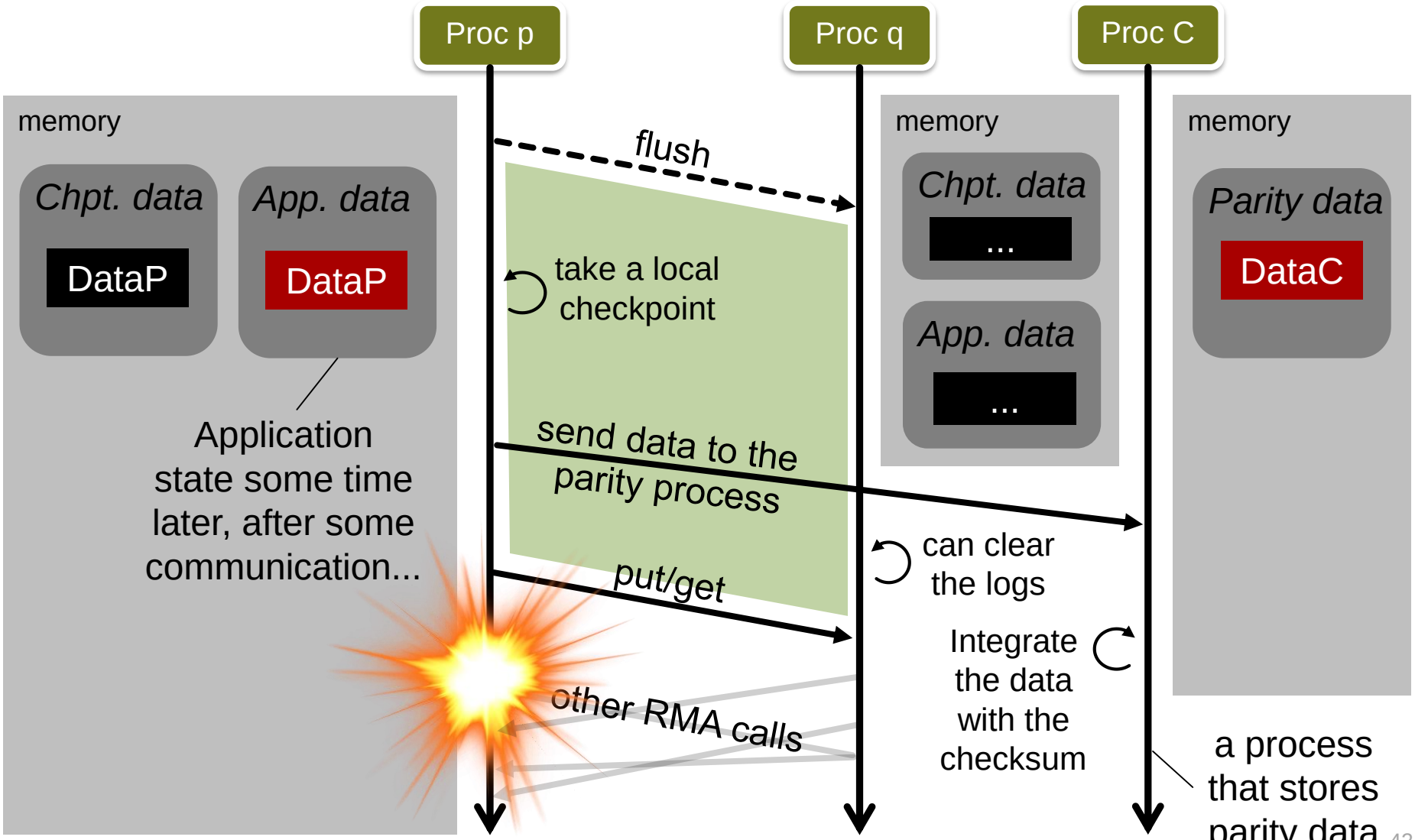
RMA: CHECKPOINTING

The Epoch Condition



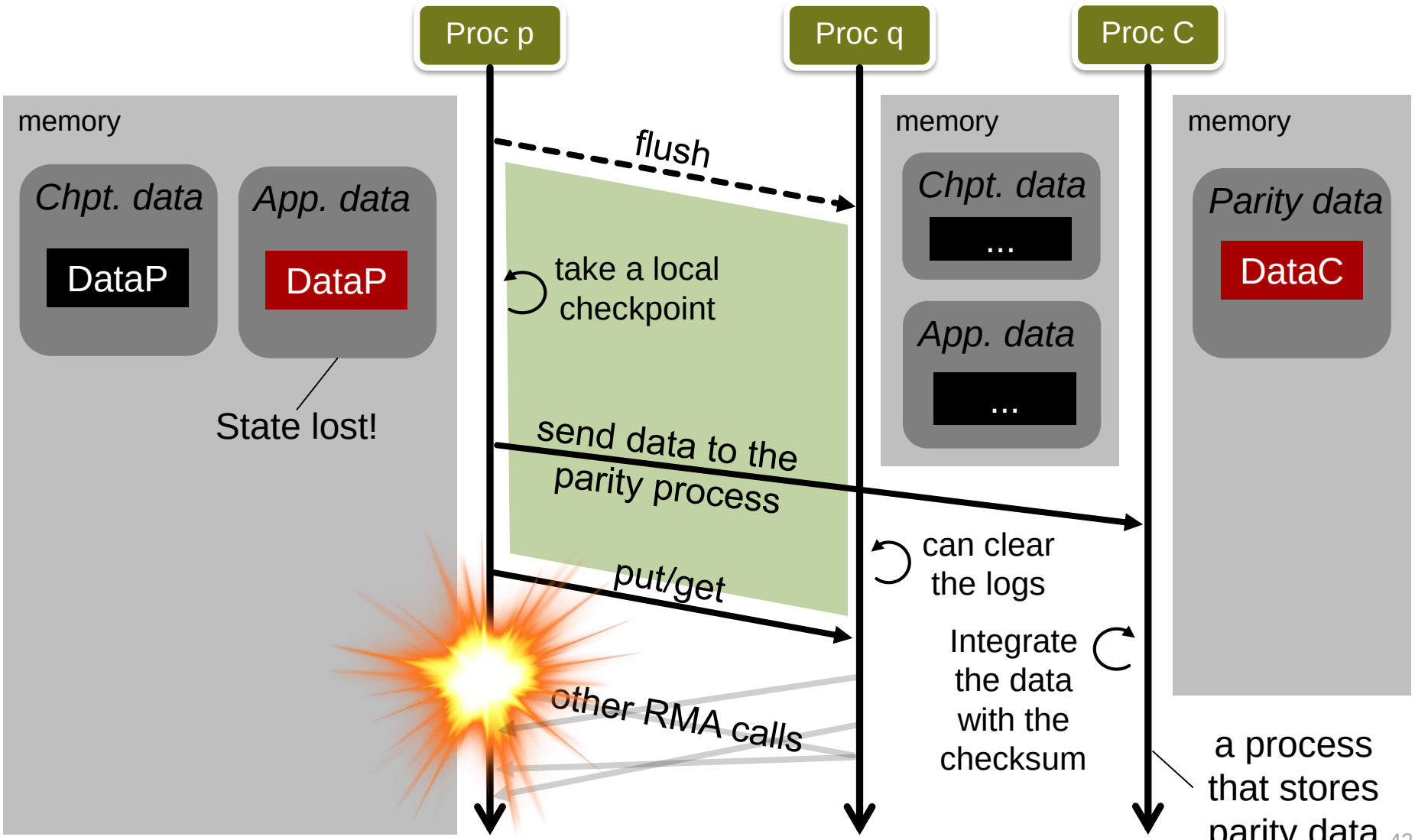
RMA: CHECKPOINTING

The Epoch Condition



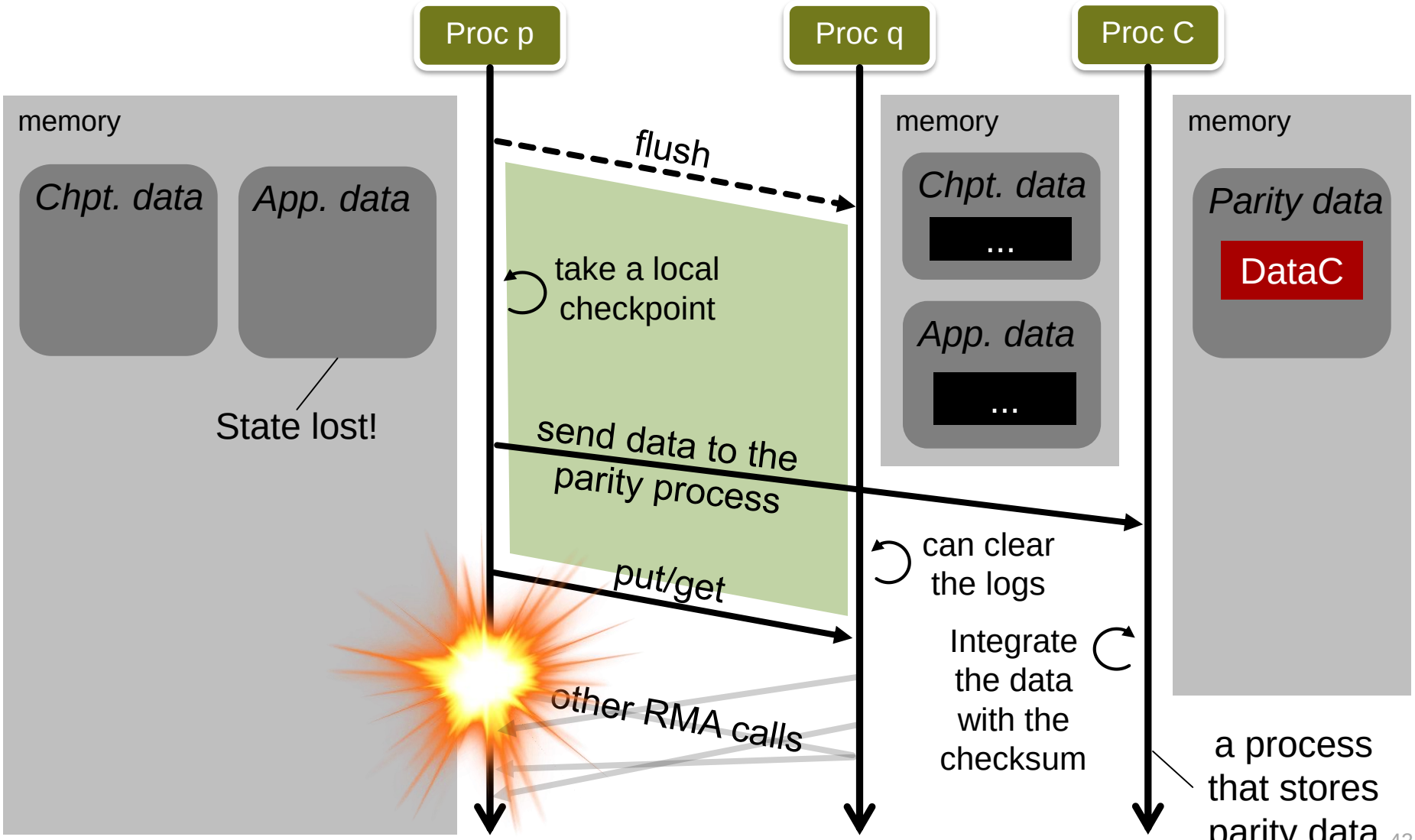
RMA: CHECKPOINTING

The Epoch Condition



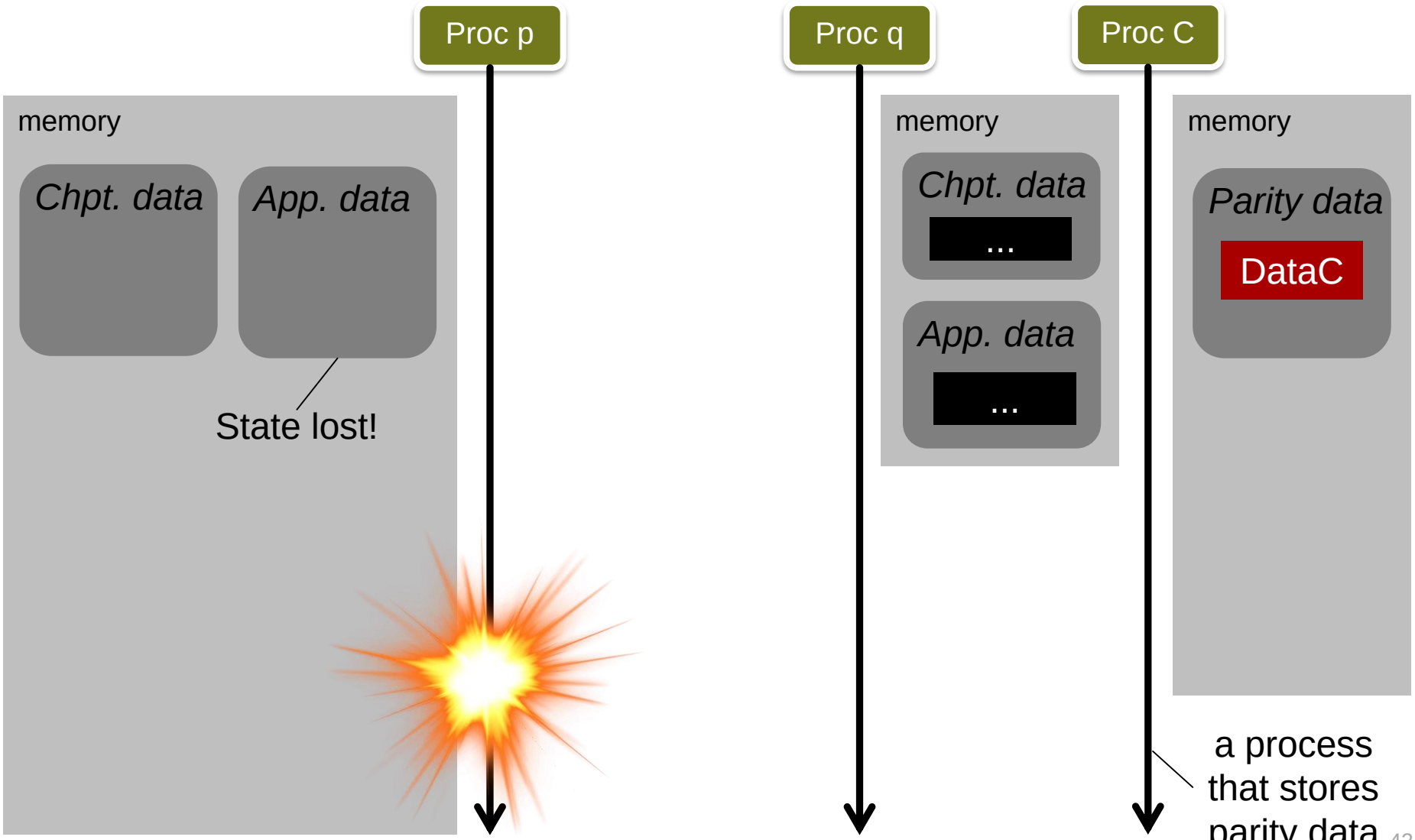
RMA: CHECKPOINTING

The Epoch Condition

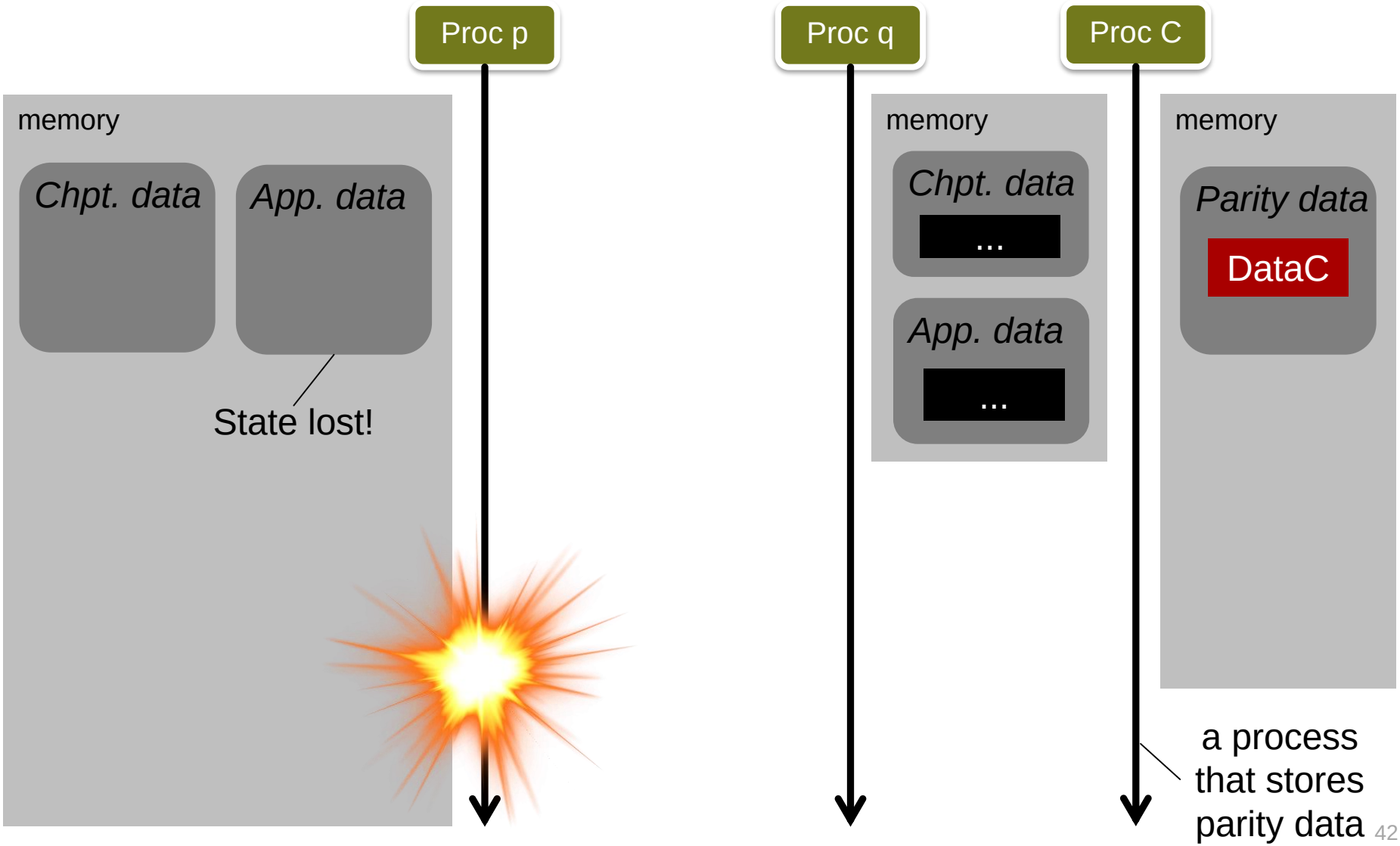


RMA: CHECKPOINTING

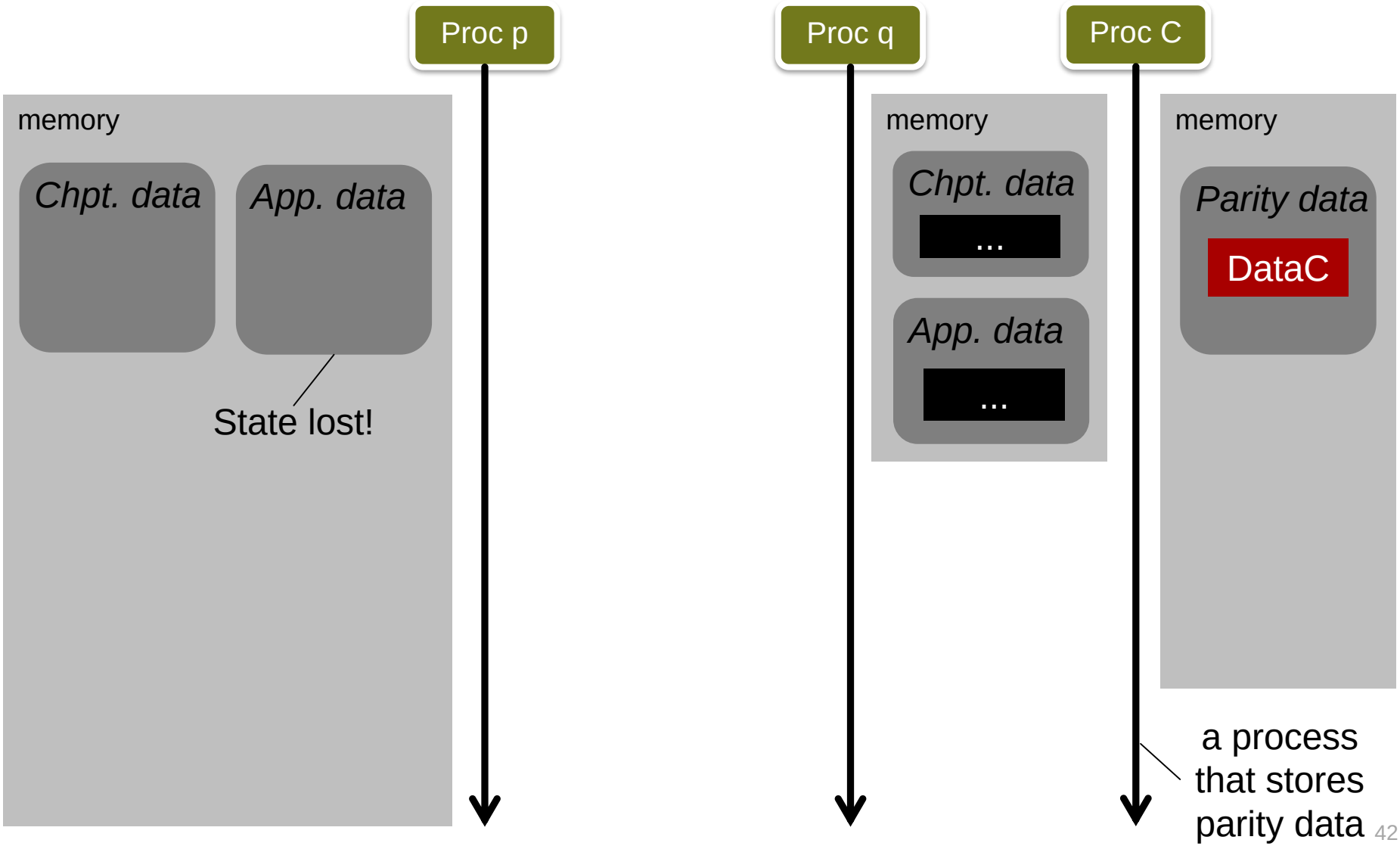
The Epoch Condition



RMA: RECOVERY

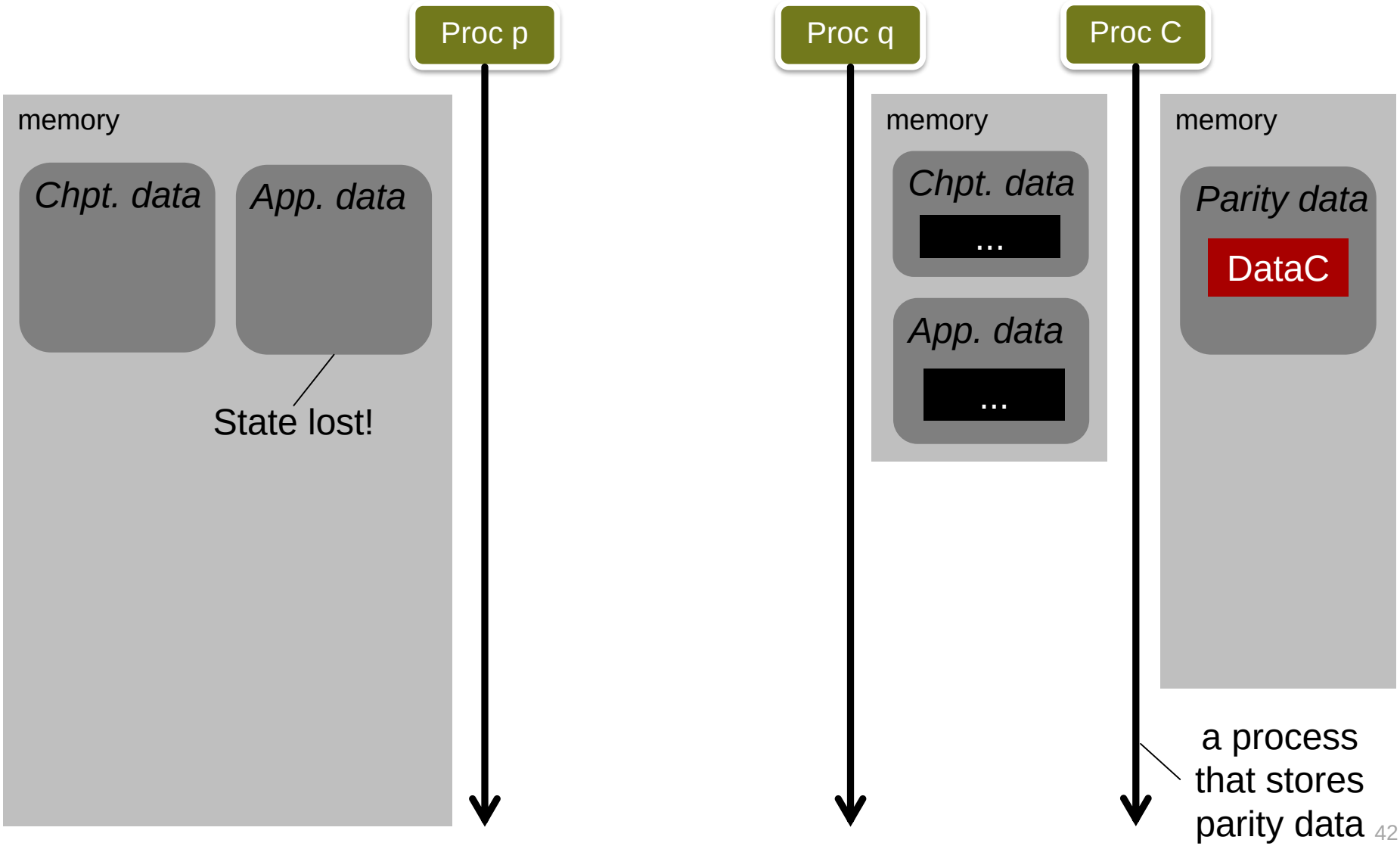


RMA: RECOVERY



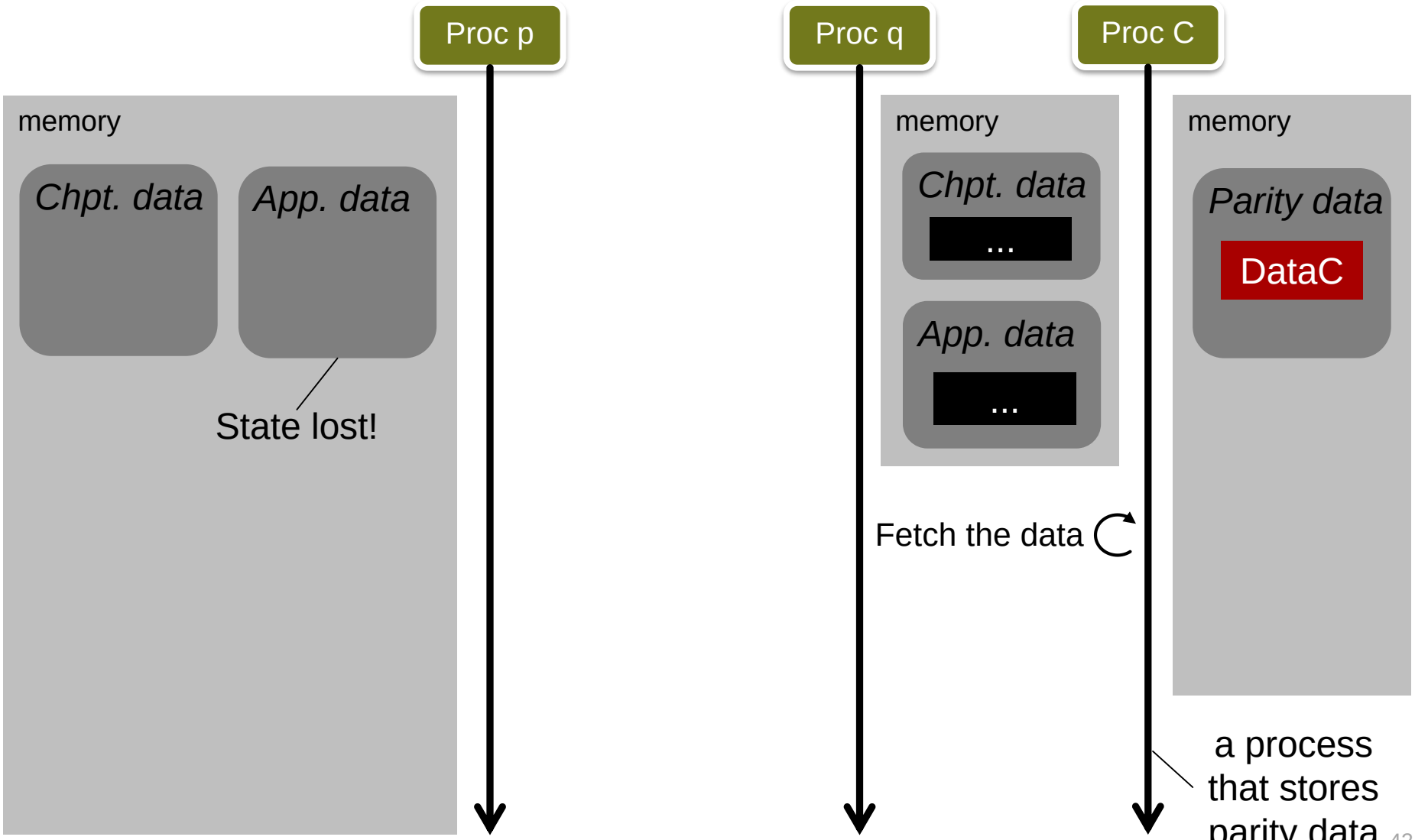
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



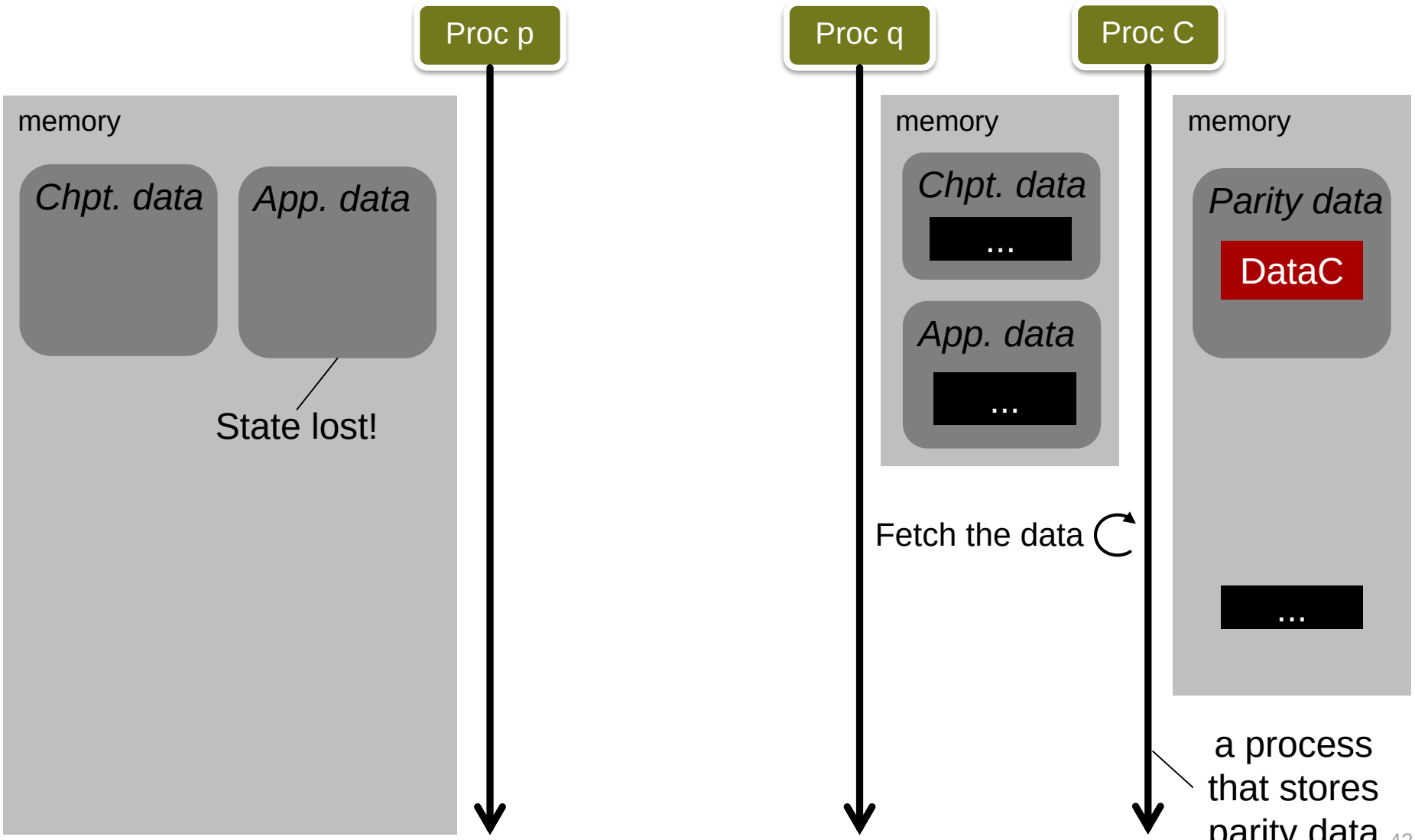
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



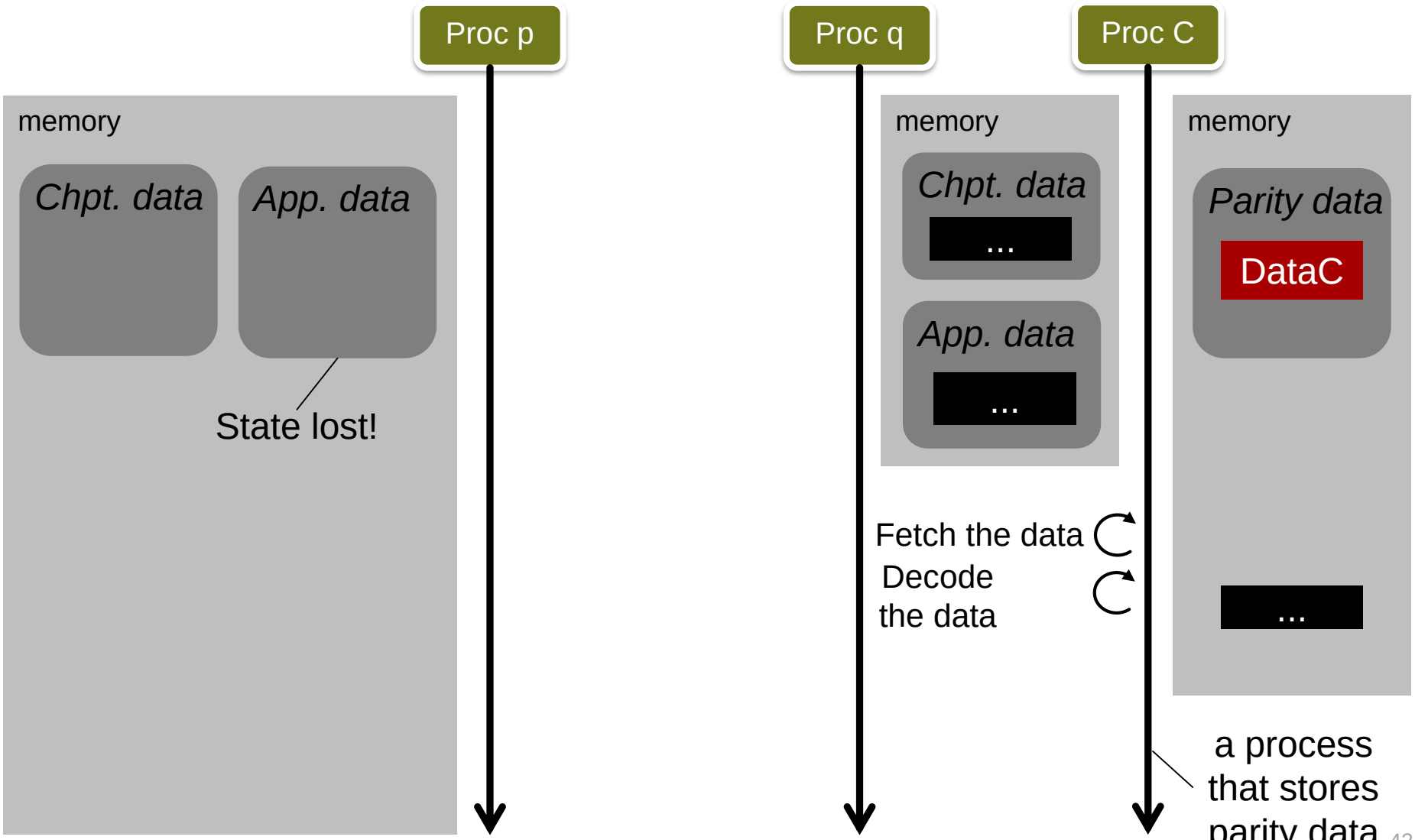
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



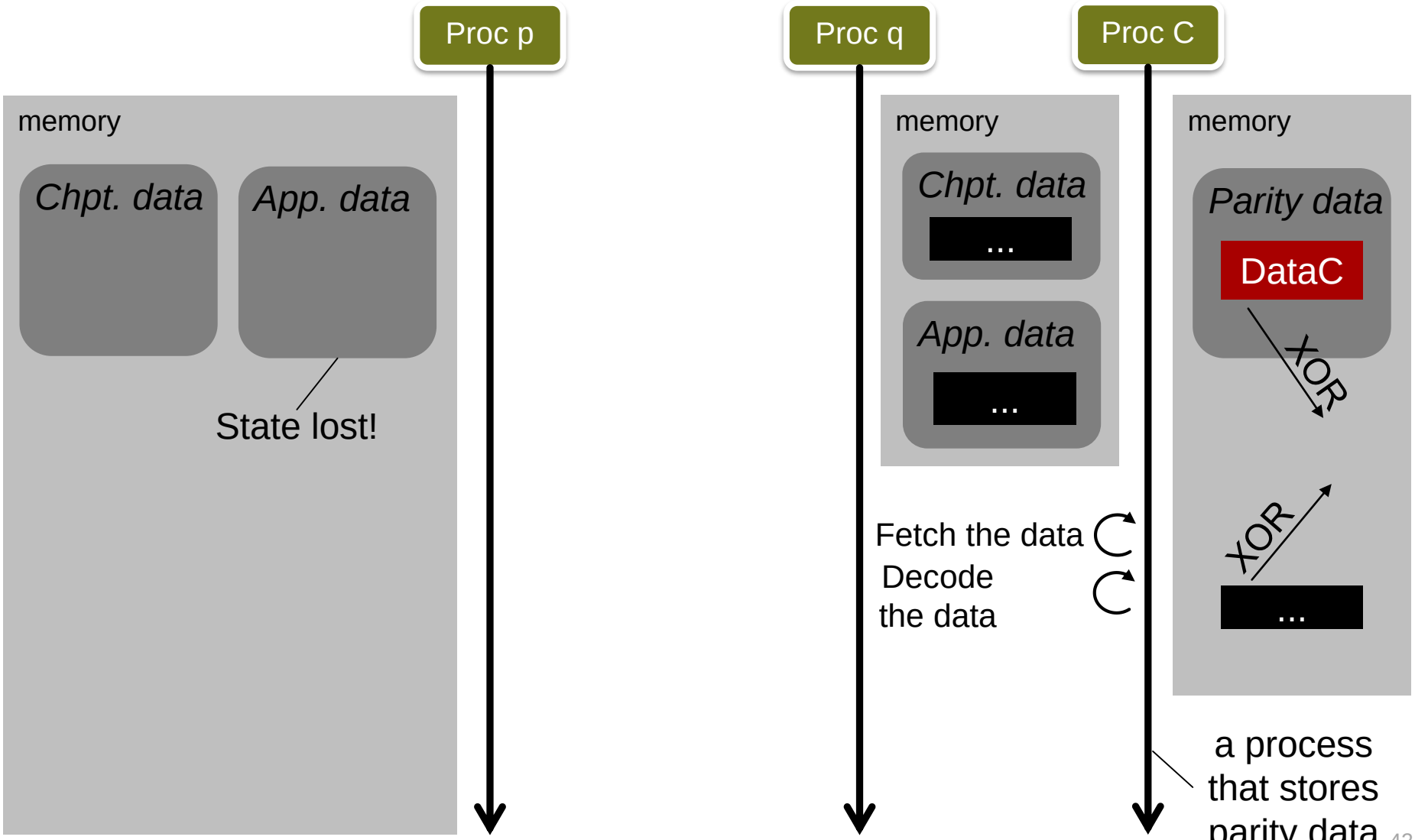
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



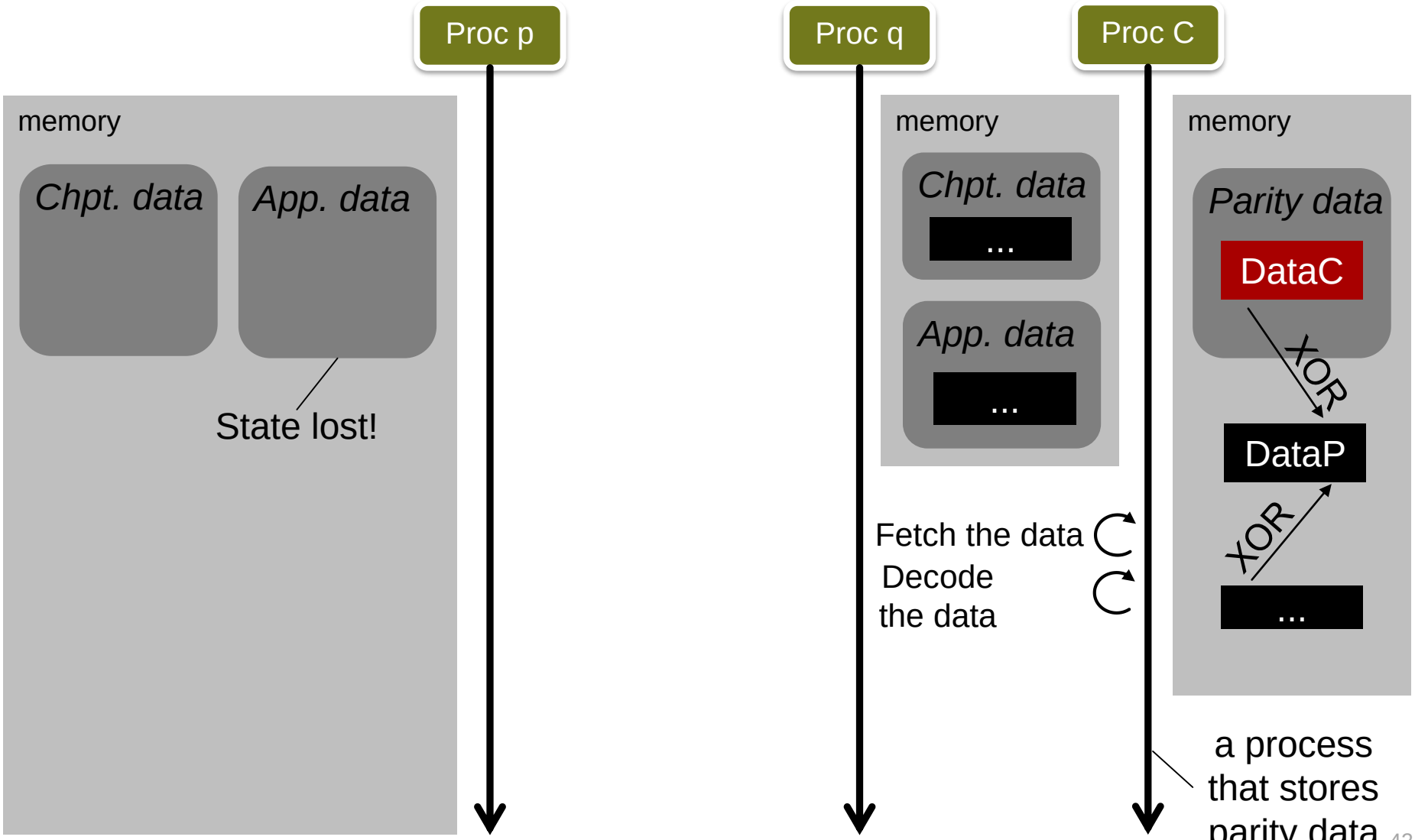
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



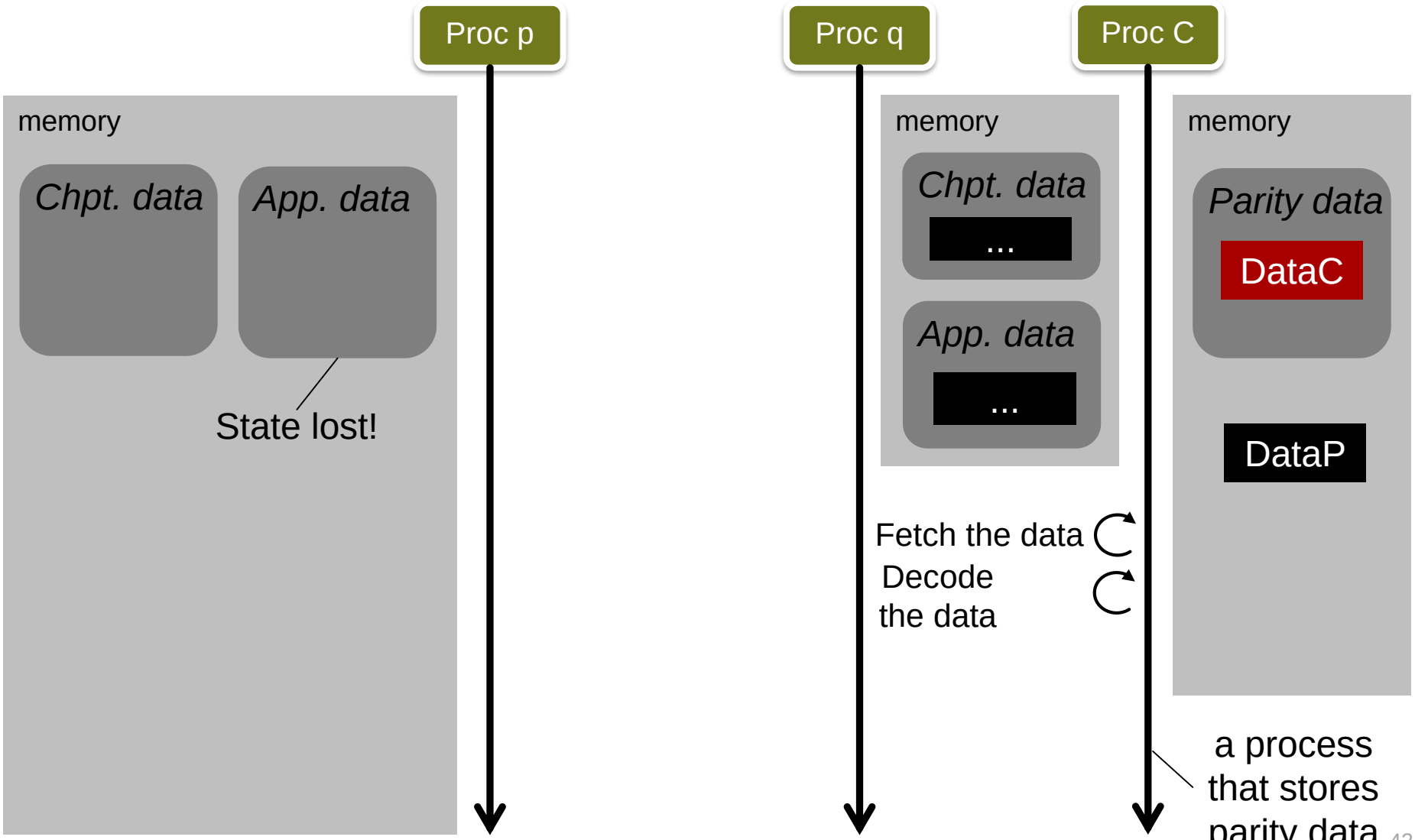
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



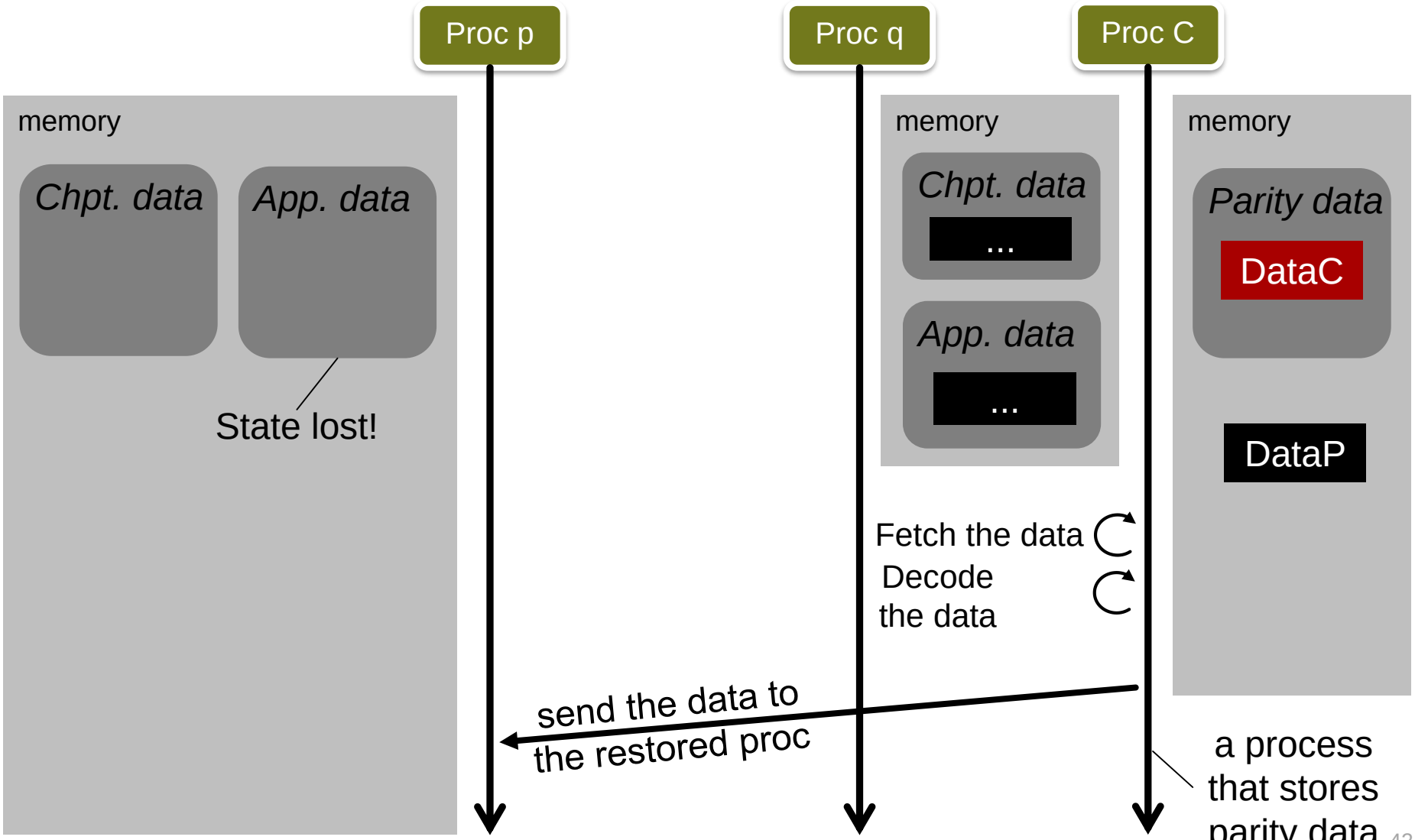
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



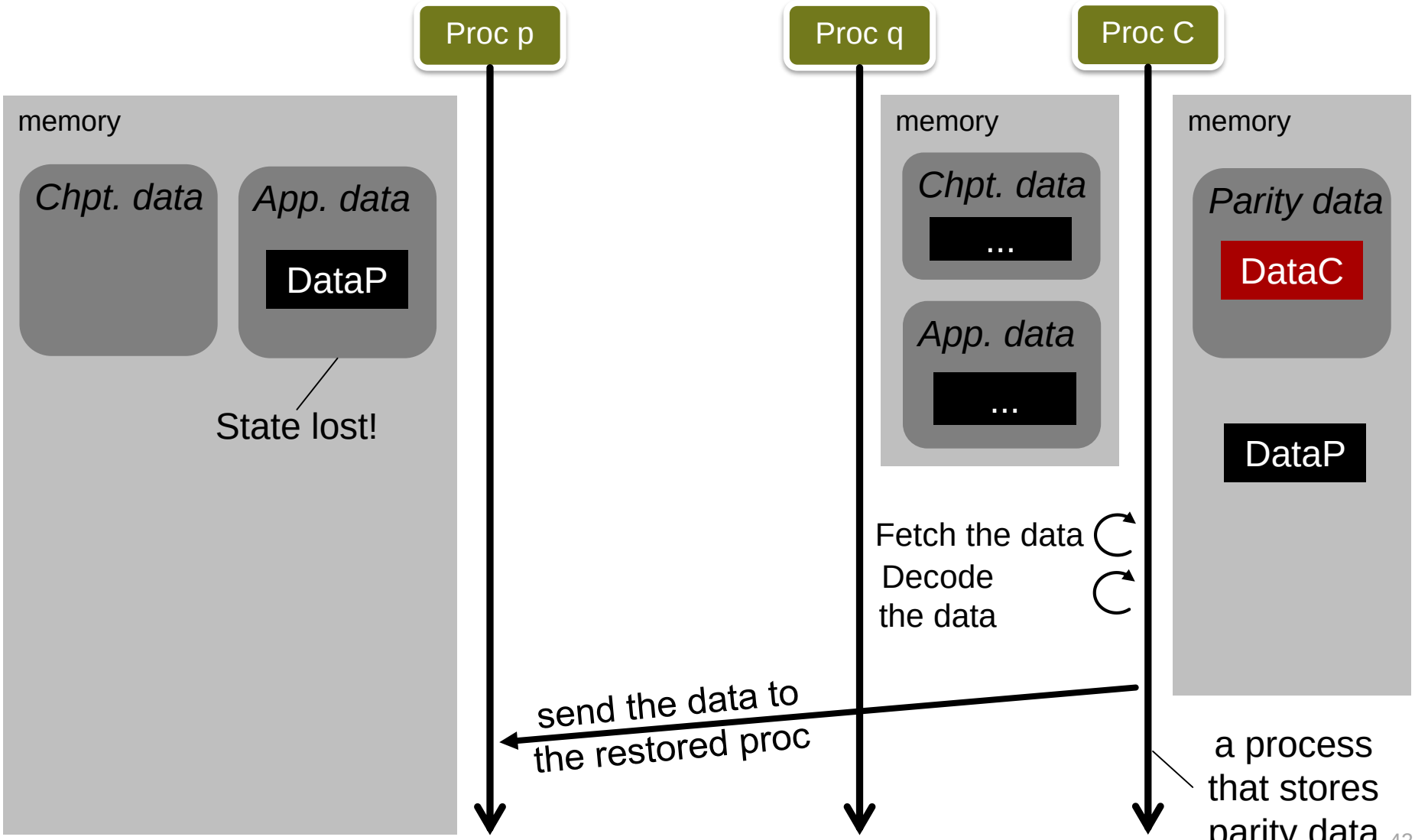
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



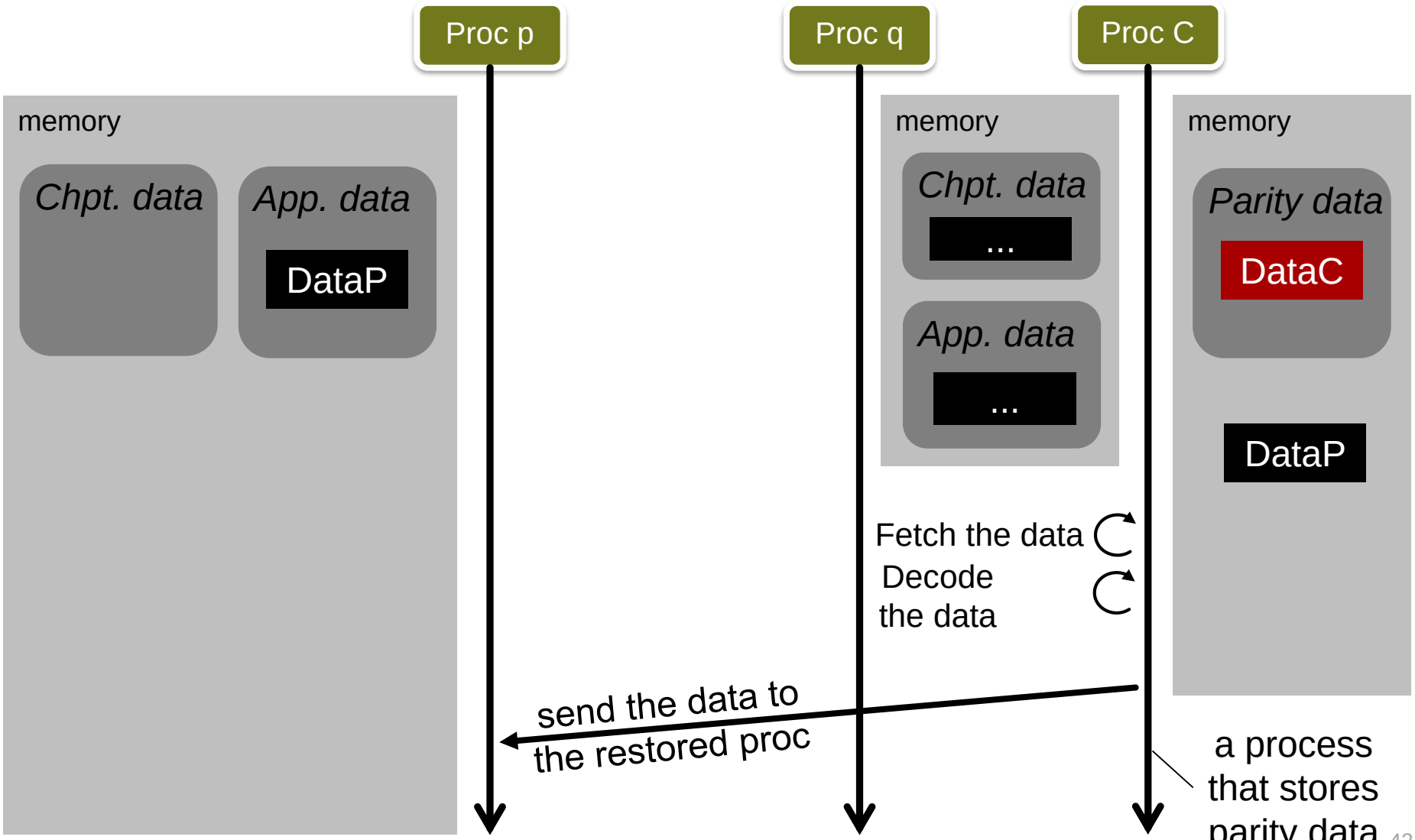
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



RMA: RECOVERY

Stage 1: restore the state upon checkpointing



RMA: RECOVERY