On Distributed Recovery for Send-Deterministic-Aware MPI Applications

Thomas Ropars, Amina Guermouche, Marc Snir and Franck Cappello

Fault Tolerance in Large Scale HPC Systems

- At very large scale, failures are not rare anymore:
  - Exascale : MTBF between one day and a few hours.

- How to provide suitable fault tolerance solutions for MPI applications ?
  - Replication is too costly:
    - Duplication of the workload.

  - Should be based on process checkpointing:
    - Rollback-recovery protocol.

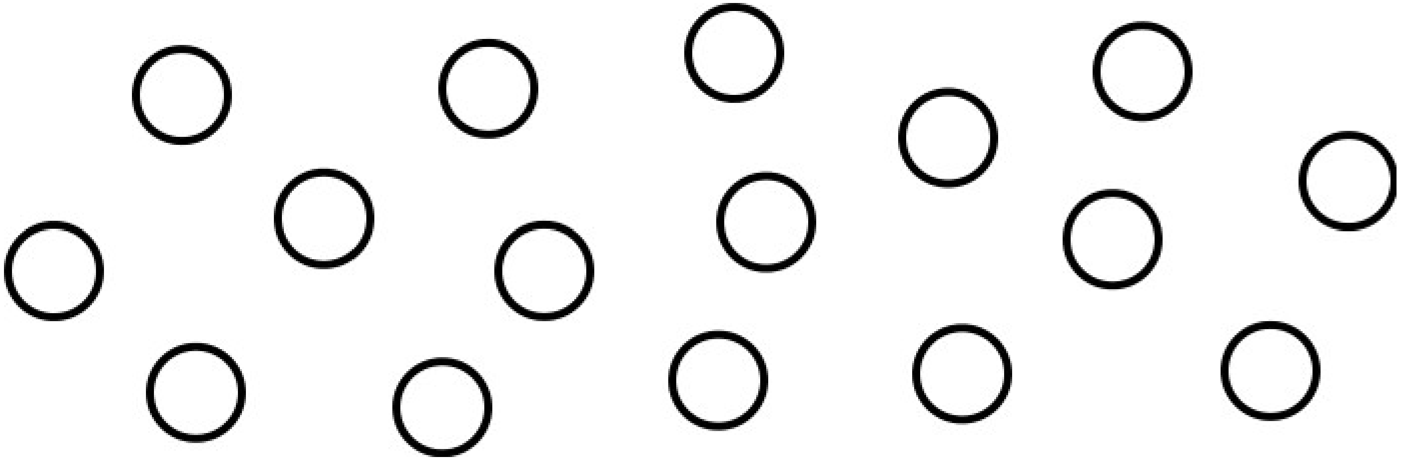
System Model

- A parallel application
  - A set of n processes.
  - A set of channels connecting any ordered pair of processes:
    - FIFO and reliable.
- An asynchronous distributed system
  - Processes communicate by exchanging messages.
    - Causal dependencies between processes states (Lamport's happened-before relation).
- Crash Failure Model
  - Multiple concurrent failures are possible

Goal: Failure Containment

- International Exascale Software Project road-map mentions **extending the applicability of fault tolerance techniques towards more local recovery** as one of the main research directions.

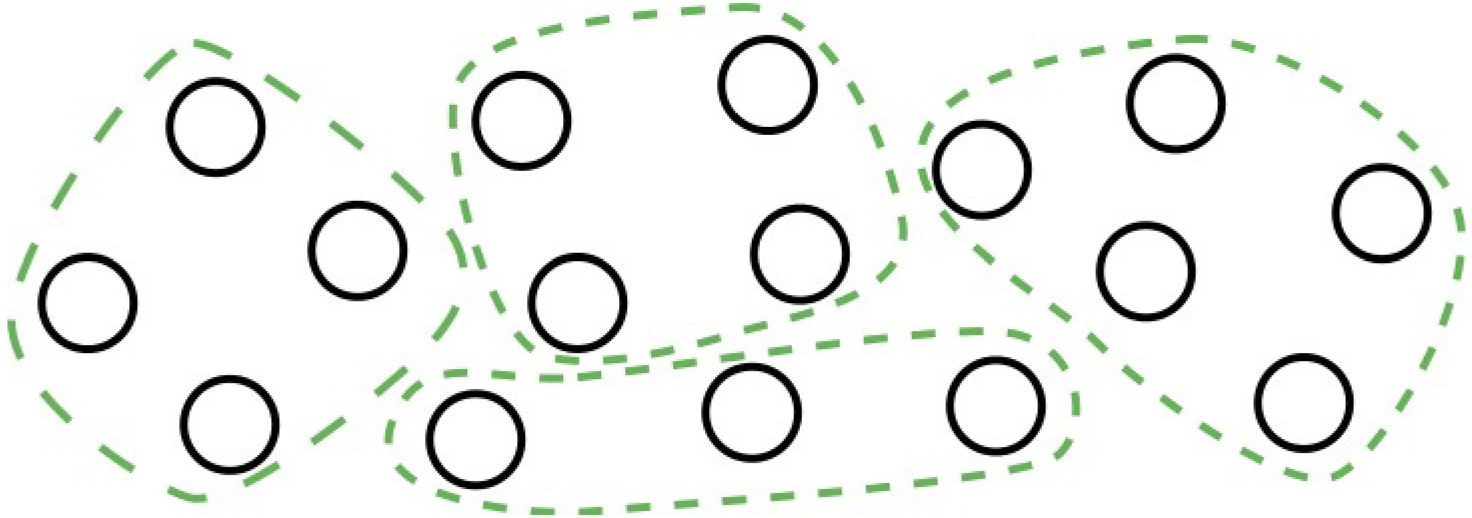
- We focus on failure containment:
  - Limiting the consequences of a failure to a subset of the processes.



Goal: Failure Containment

- International Exascale Software Project road-map mentions **extending the applicability of fault tolerance techniques towards more local recovery** as one of the main research directions.

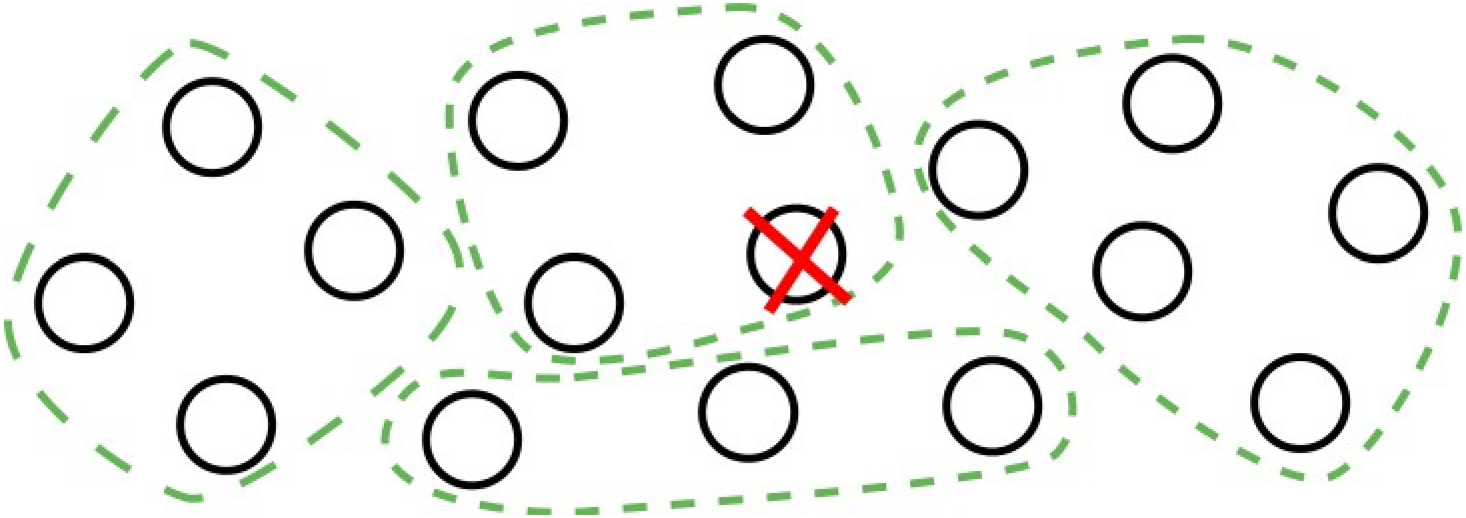
- We focus on failure containment:
  - Limiting the consequences of a failure to a subset of the processes.



Goal: Failure Containment

- International Exascale Software Project road-map mentions **extending the applicability of fault tolerance techniques towards more local recovery** as one of the main research directions.

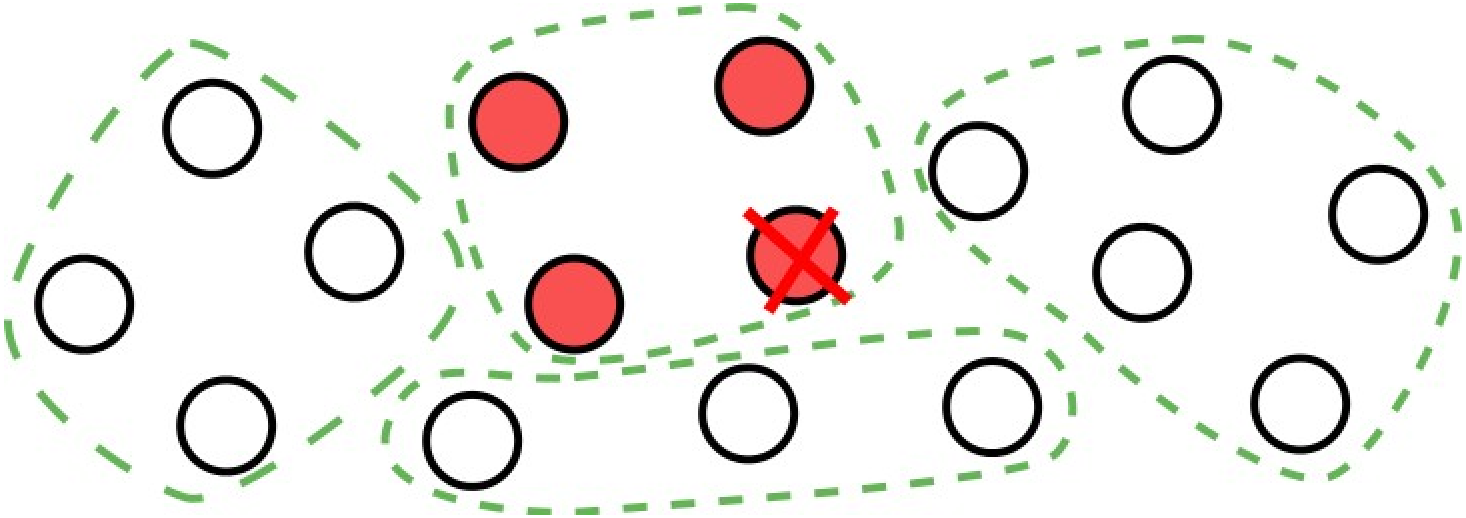
- We focus on failure containment:
  - Limiting the consequences of a failure to a subset of the processes.



Goal: Failure Containment

- International Exascale Software Project road-map mentions **extending the applicability of fault tolerance techniques towards more local recovery** as one of the main research directions.

- We focus on failure containment:
  - Limiting the consequences of a failure to a subset of the processes.



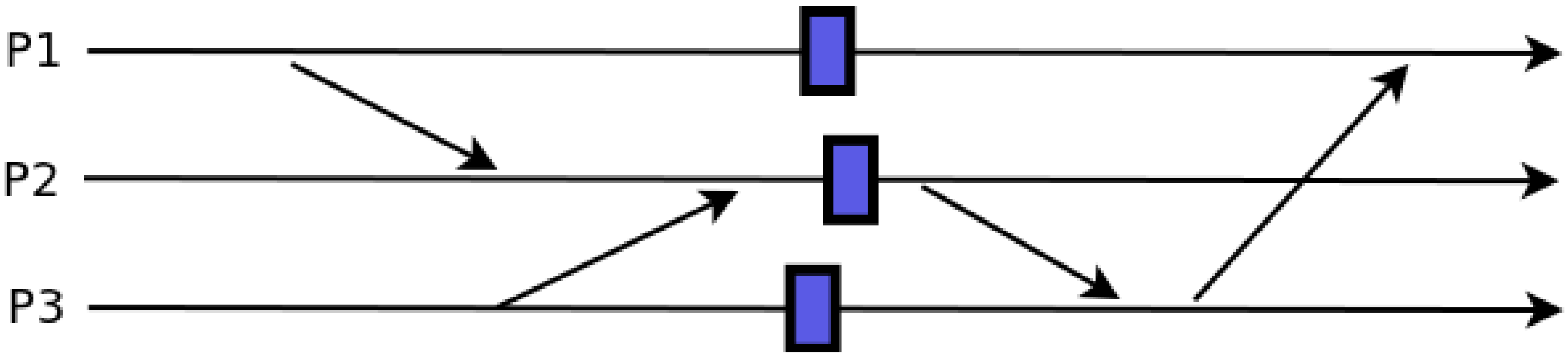
Failure Containment: Expected Benefits

- **Limiting the amount of rolled-back computation.**
  - Reduces the amount of energy that is wasted.
- **Speeding up recovery.**
  - [Rao, Alvisi and Vin 1998]: The Cost of Recovery in Message Logging Protocols
- **Improving the overall system utilization.**
  - Resources not involved in the recovery could be used by other applications.

Our Work: Current Status

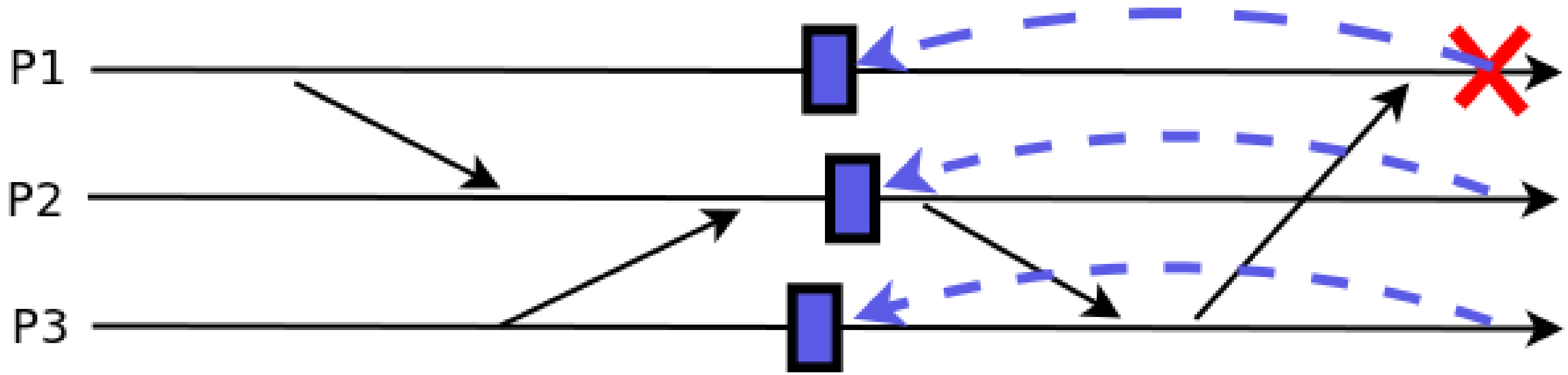
- **Limiting the amount of rolled-back computation.**
  - HydEE: Hybrid Rollback-Recovery Protocol
    - Coordinated checkpointing + sender-based message logging.
    - Send-deterministic applications (no data saved on stable storage).
- **Speeding up recovery.**
  - Can we achieve this with HydEE ?
    - Current solution: centralized (scalability issue).
    - New research direction:
      - Send-deterministic-aware applications (fully distributed recovery).
- **Improving the overall system utilization.**
  - Out of scope

Coordinated Checkpointing



- Simple to implement
- Good performance in failure free execution
- Efficient garbage collection:
  - Limited storage utilization
- No assumption on the determinism of the application

Coordinated Checkpointing

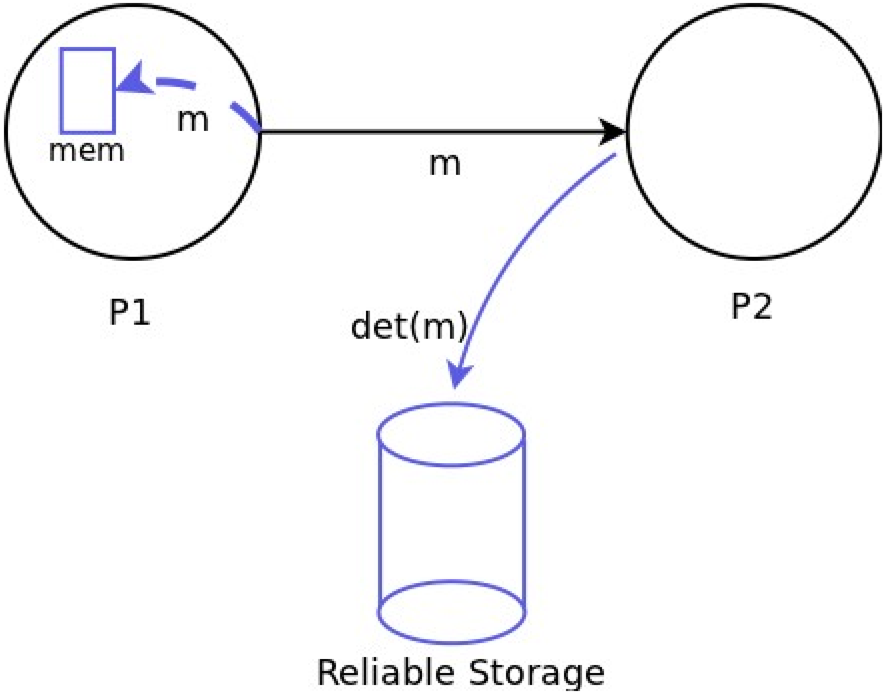


- Simple to implement
- Good performance in failure free execution
- Efficient garbage collection:
  - Limited storage utilization
- No assumption on the determinism of the application

No failure Containment

To Provide Failure Containment, Message Logging is Needed

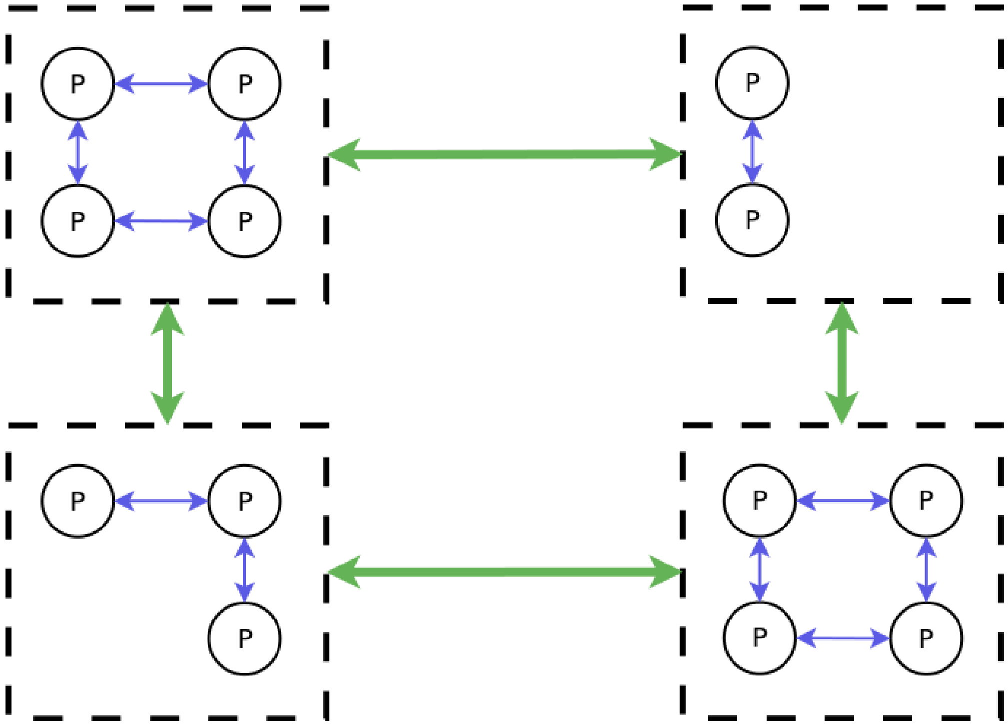
- Message logging (Pessimistic or Causal) provides perfect failure containment.
  - To be able to replay a message:
    - Message content saved in the sender memory.
    - Reception order saved in a reliable storage.



- Large amount of data to log
- Communication performance degradation
  - Due to event logging (stable storage)
- Assumption: piecewise deterministic applications

Hybrid Rollback-Recovery Protocols

- Clustering of the application processes
  - Coordinated checkpointing inside each cluster
  - Logging of inter-cluster messages



Related Work

- Assumption: piecewise deterministic applications
  - All non deterministic events have to be logged reliably [Bouteiller *et al*, 2011]
    - Pessimistic approach: synchronization with a reliable storage
      - Meneses *et al, 2010*
      - Bouteiller *et al*, 2011
    - Causal approach: Piggybacking on messages
      - Yang *et al*, 2009
      - Meneses *et al*, 2011

HydEE: An Hybrid Protocol for Send-Deterministic Applications

- Coordinated checkpointing inside each cluster
- Message logging for inter-clusters communications
  - Sender-based message logging
- Relies on the send-deterministic execution model:
  - No event logging
    - No need for a stable storage
- Proof that it can tolerate multiple concurrent failures

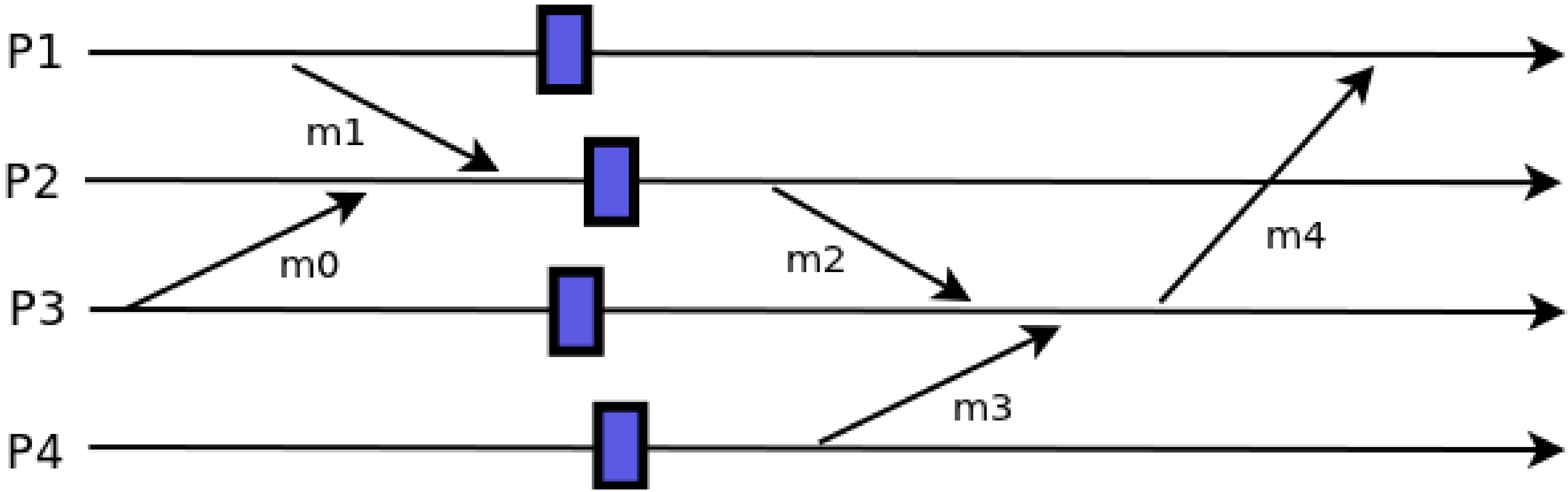
Most MPI HPC Applications are Send-Deterministic

- Definition:
  - Given a set of input parameters, sequences of message sendings are always the same in any correct execution.
  - Messages reception order does not change processes behavior.

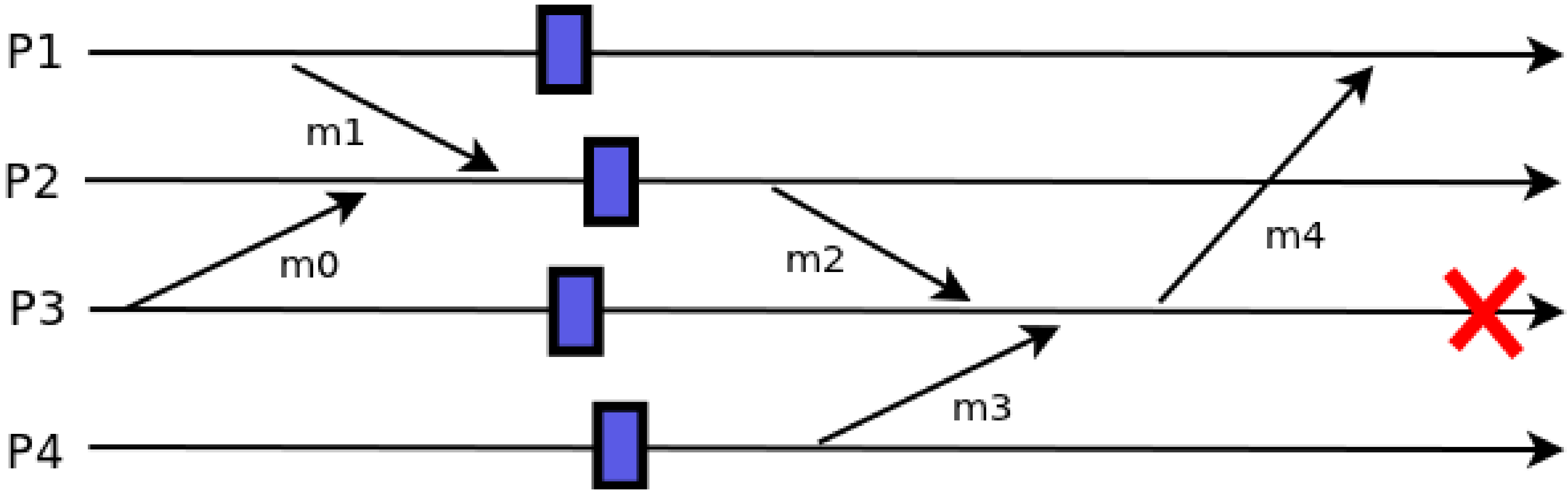
- Static analysis of 27 HPC applications [Cappello *et al*, 2010]
  - NAS Benchmarks
  - 6 NERSC Benchmarks
  - 2 USQCD Benchmarks
  - 6 Sequoia Benchmarks
  - SpecFEM3D, Nbody, Ray2mesh
  - ScaLAPACK SUMMA

25 over 27 are send-deterministic

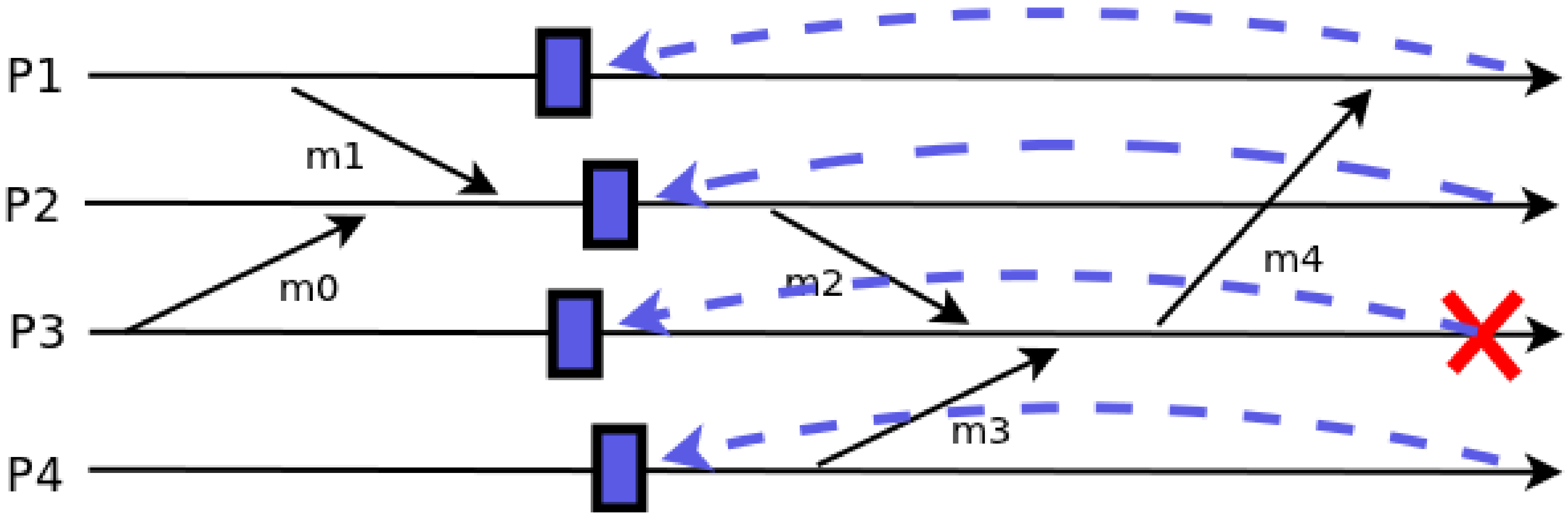
Send-Determinism Can Help Improving Rollback-Recovery Protocols



Send-Determinism Can Help Improving Rollback-Recovery Protocols

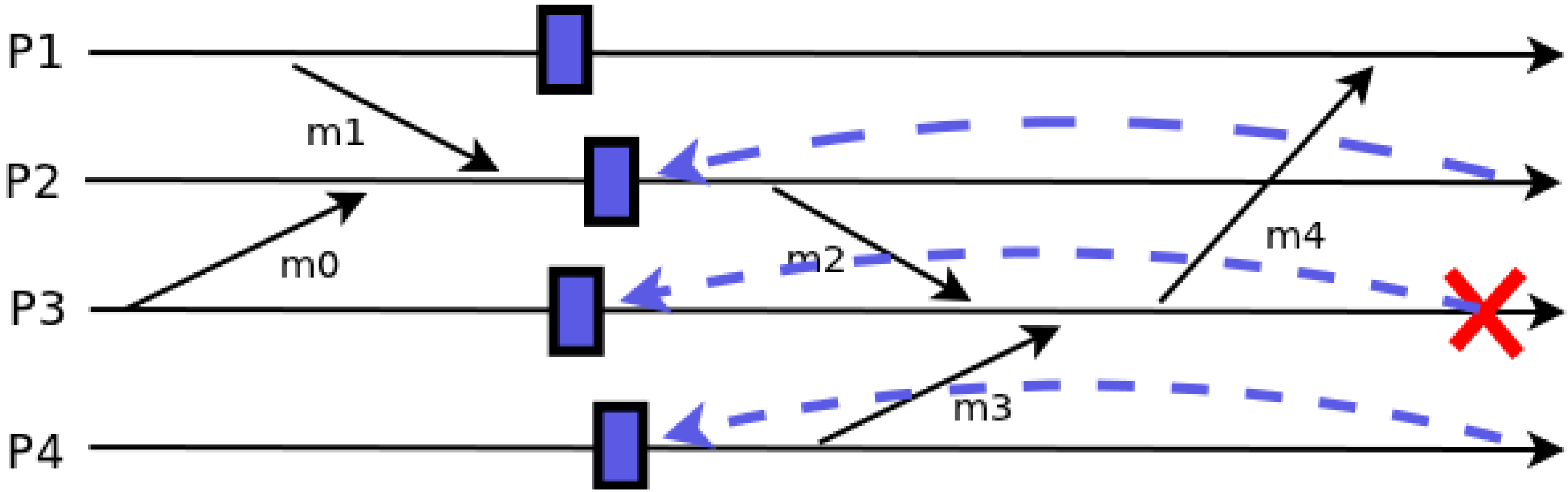


Send-Determinism Can Help Improving Rollback-Recovery Protocols



- If the application is not send-deterministic:
  - All processes have to rollback

Send-Determinism Can Help Improving Rollback-Recovery Protocols



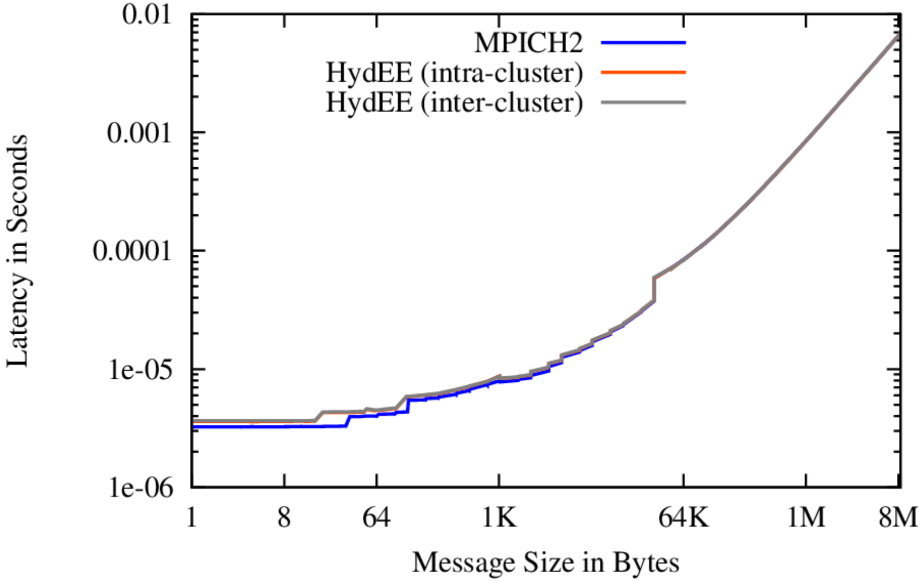
- If the application is send-deterministic:
  - Process P1 does not need to rollback
  - Message m4 will be sent in any correct execution

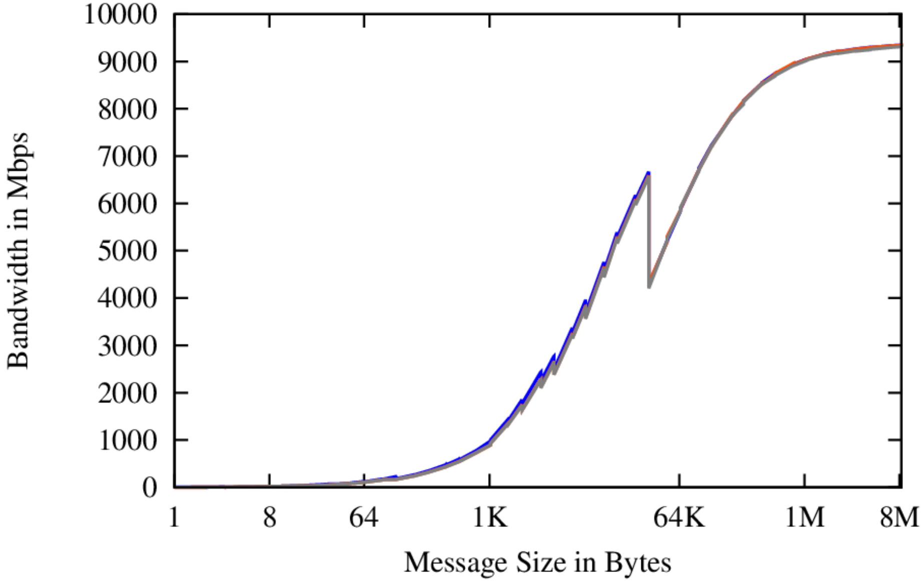
Prototype in MPICH2

- Protocol implemented in the Nemesis communication system
  - Works for TCP, Myrinet/MX, and shared memory.
  - Sender-based message logging implemented in memory.
  - Coordinated checkpointing not implemented yet.
- Support for cluster management added to the Hydra process manager

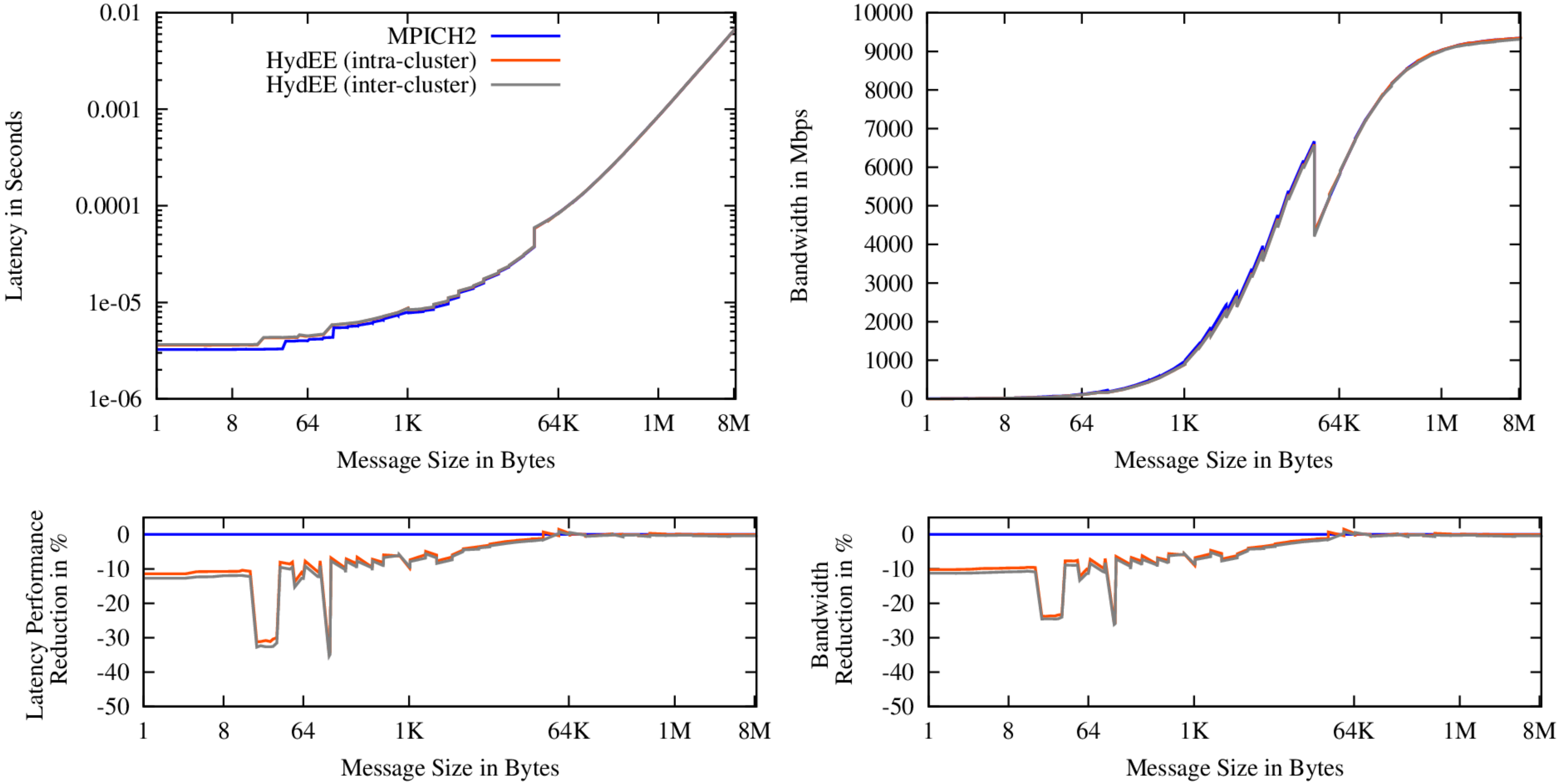
- Testbed: Lille Grid'5000 cluster
  - 25 nodes equipped with 2 AMD Opteron 285 (2 cores) processors, 4 GB of memory
  - 20 nodes equipped with 2 Intel Xeon E5440 QC (4 cores) processors, 8 GB of memory
  - 10G-PCIE-8A-C Myri-10G NIC

Communication Performance with NetPipe over MX





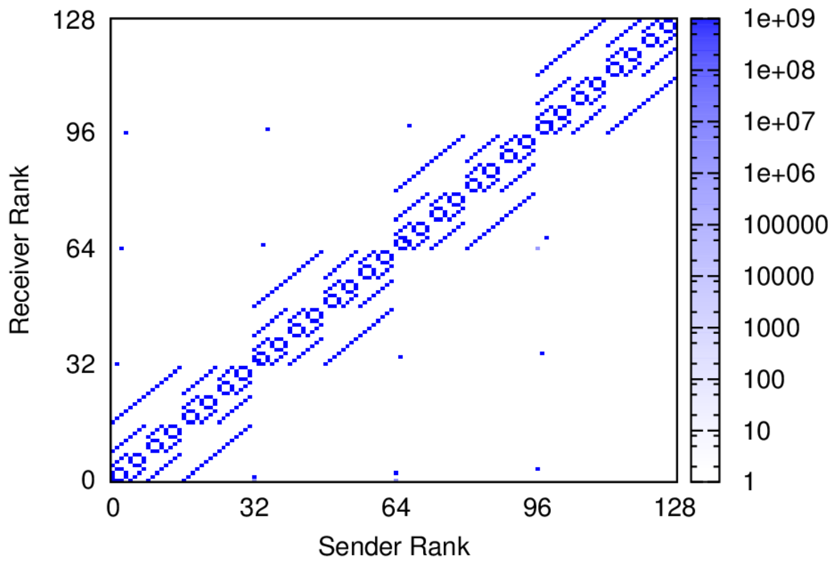
Communication Performance with NetPipe over MX



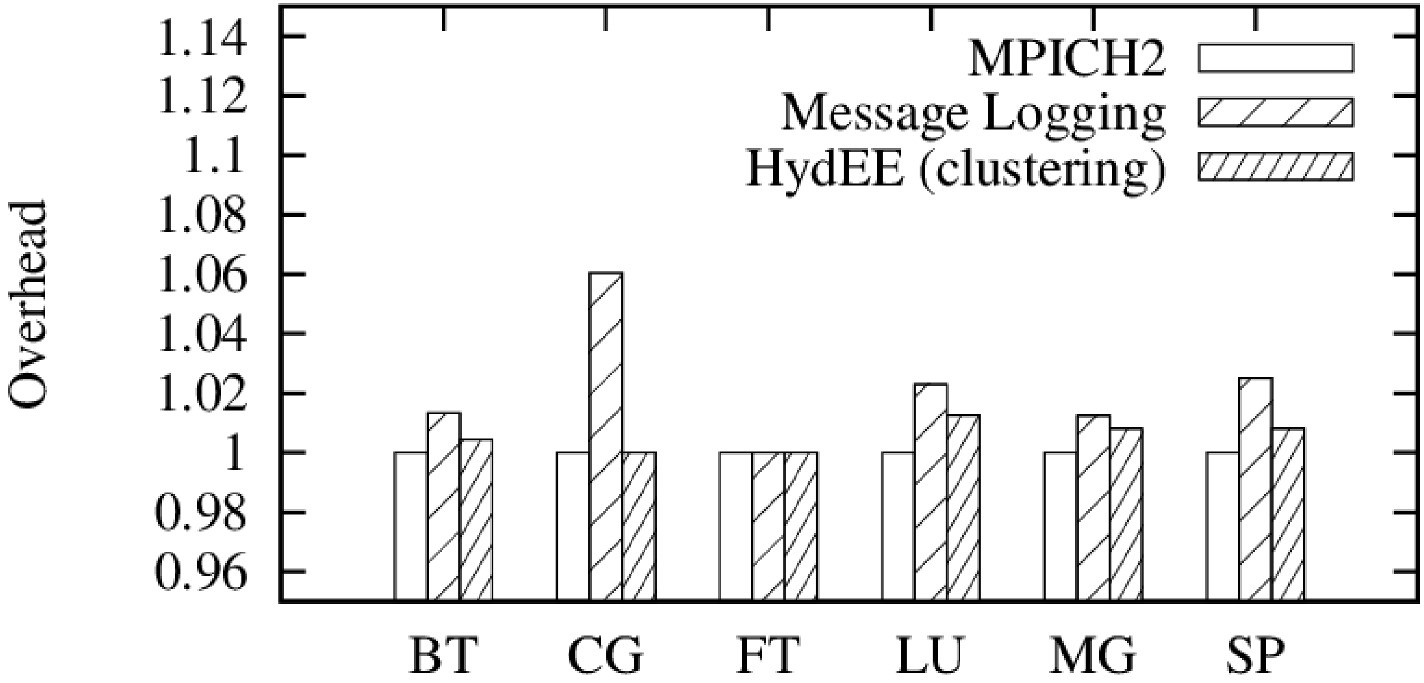
- Very little overhead (only for small messages)
  - Peaks due to data piggybacked on messages

Clustering Based on the Applications Communication Pattern

- Tool to compute the clustering based on the application communication pattern [Ropars *et al*, 2011].
  - Study of 10 representative benchmarks on 1024 processes (covering 6 out of the 7 main Berkeley's dwarfs)
    - < 15% of processes to rollback after a failure
    - < 15% of the communication data to log
  - Example of NAS CG
    - 3.2% of the processes to rollback
    - 16% of logged data



NAS Performances (Class D, 256 processes)

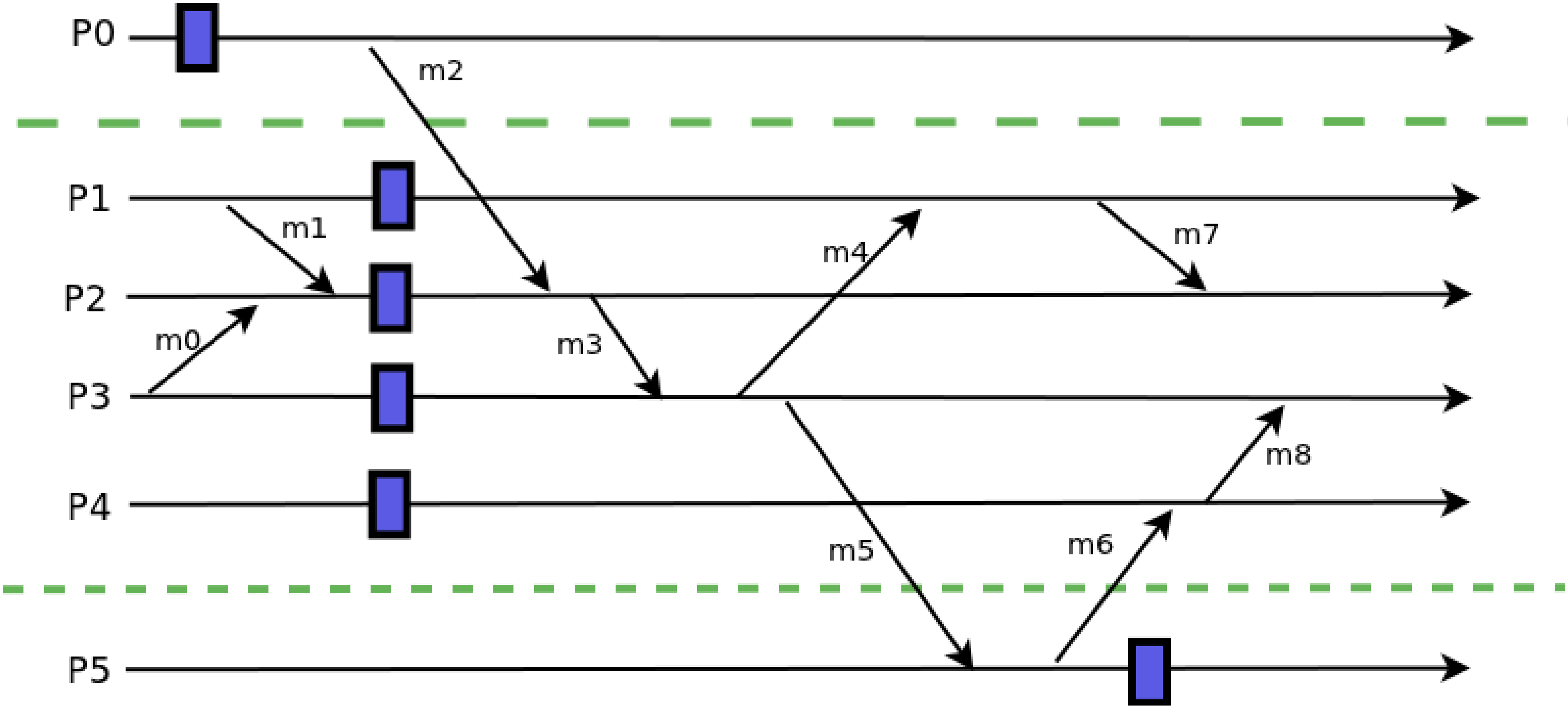


- Test run over Myrinet/MX
- No overhead with HydEE

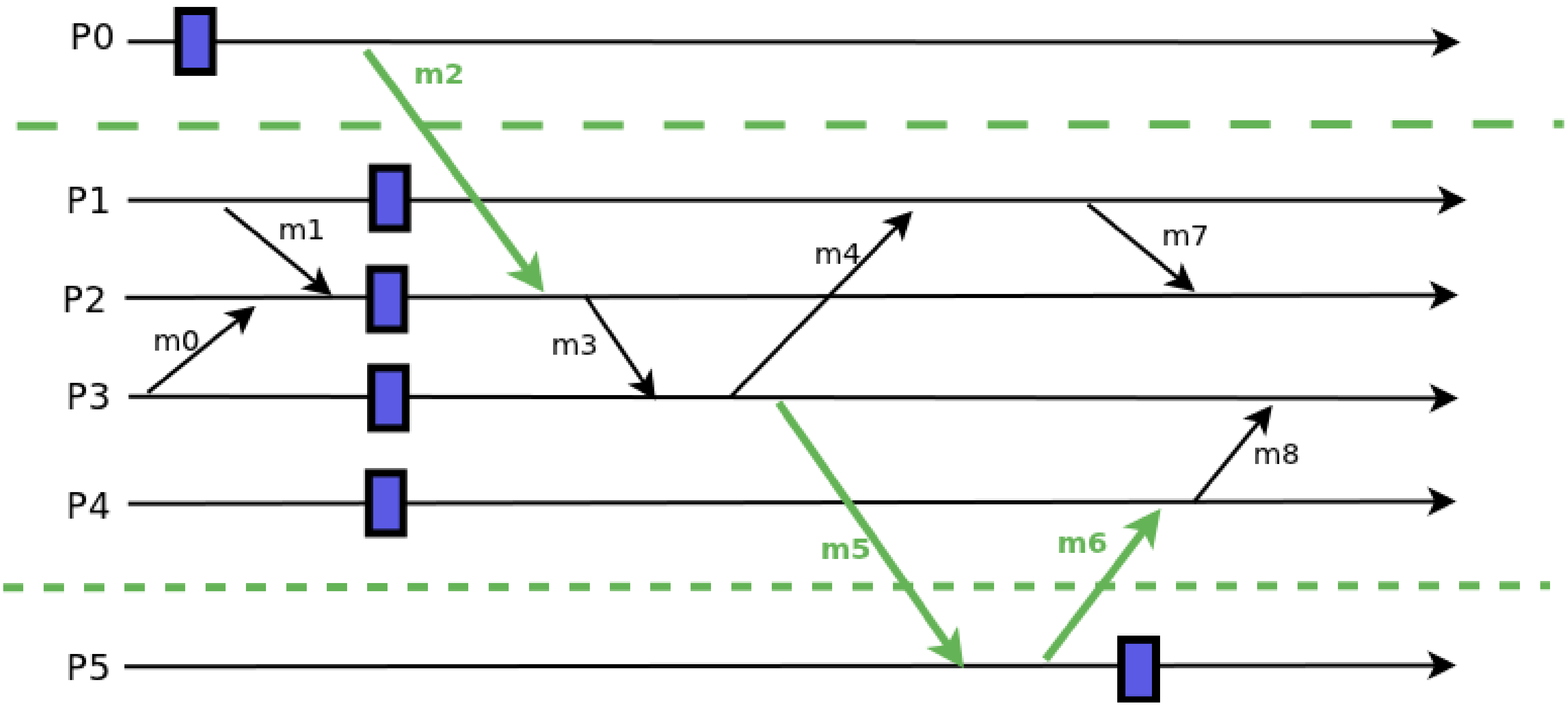
Recovery with HydEE

- Transparent solution
  - Based on phases
  - Requires the help of an additional recovery process
- Solution for Send-Deterministic-Aware applications
  - Fully distributed recovery

Recovery is not that Simple

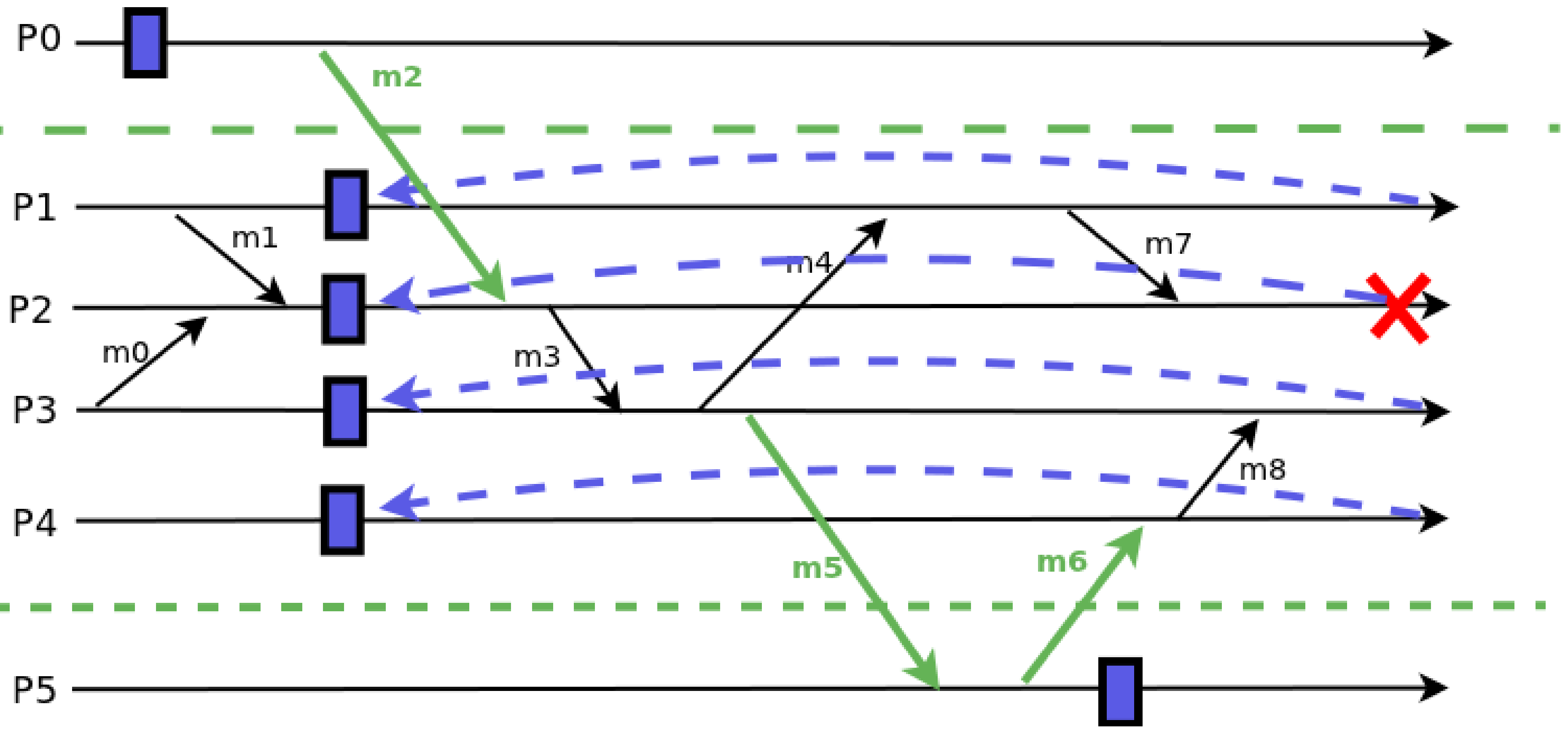


Recovery is not that Simple

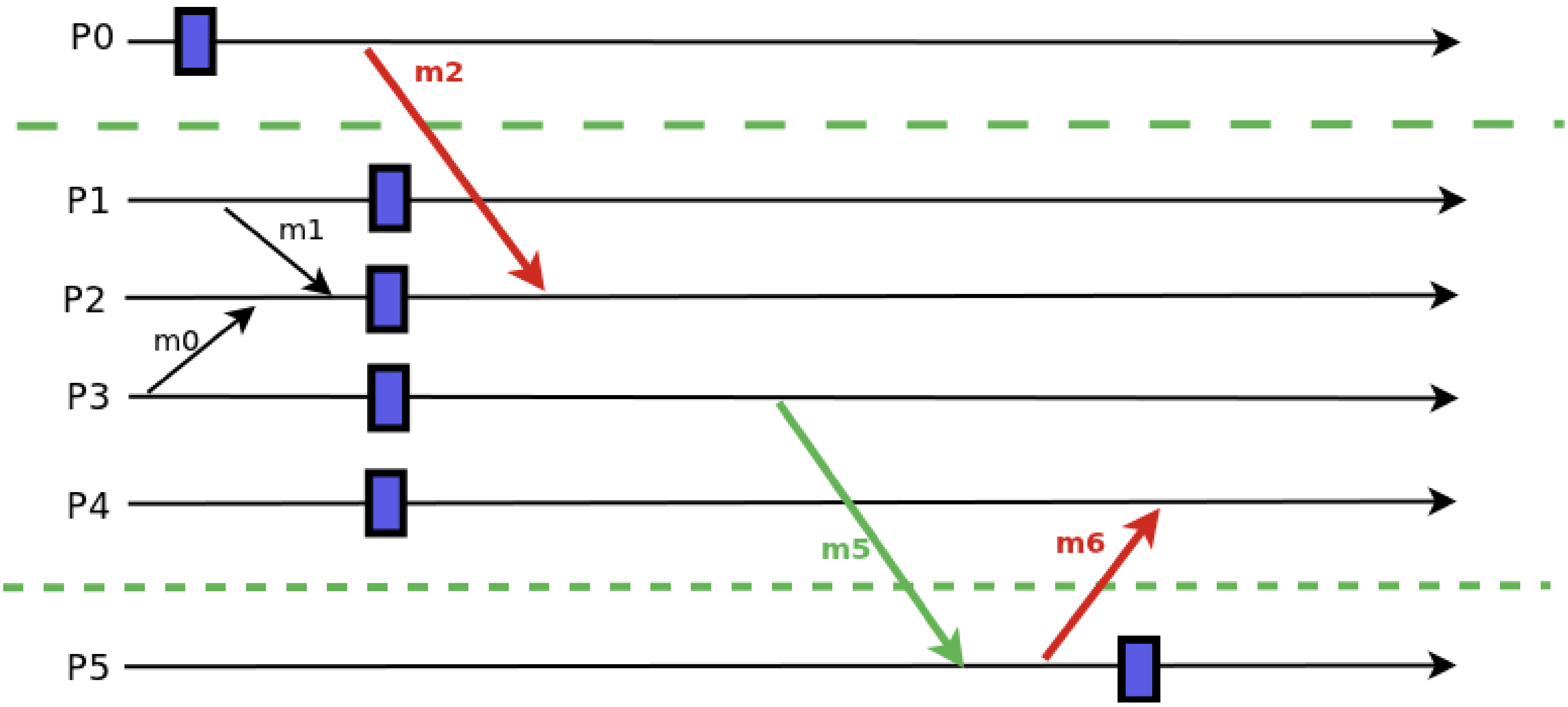


- Inter-cluster messages are logged

Recovery is not that Simple

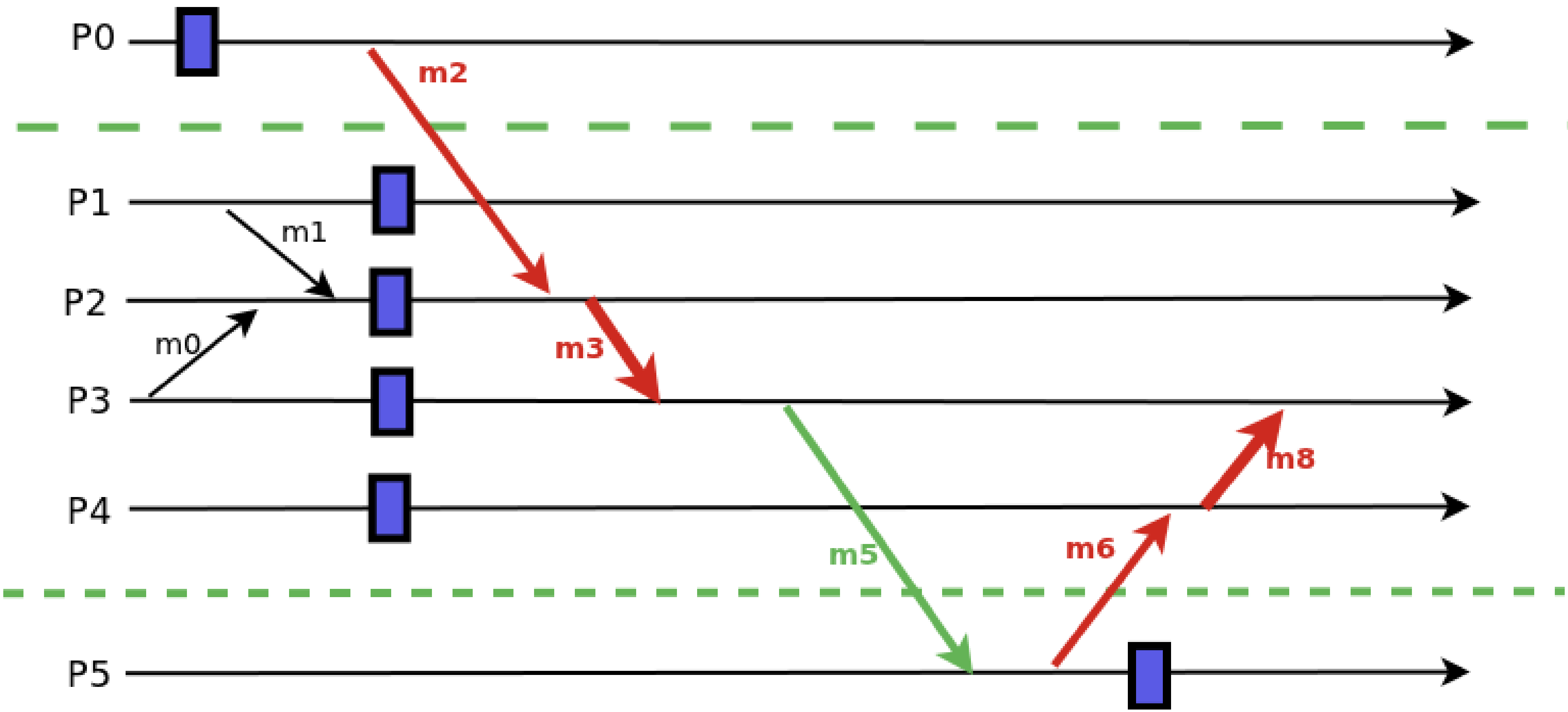


Recovery is not that Simple



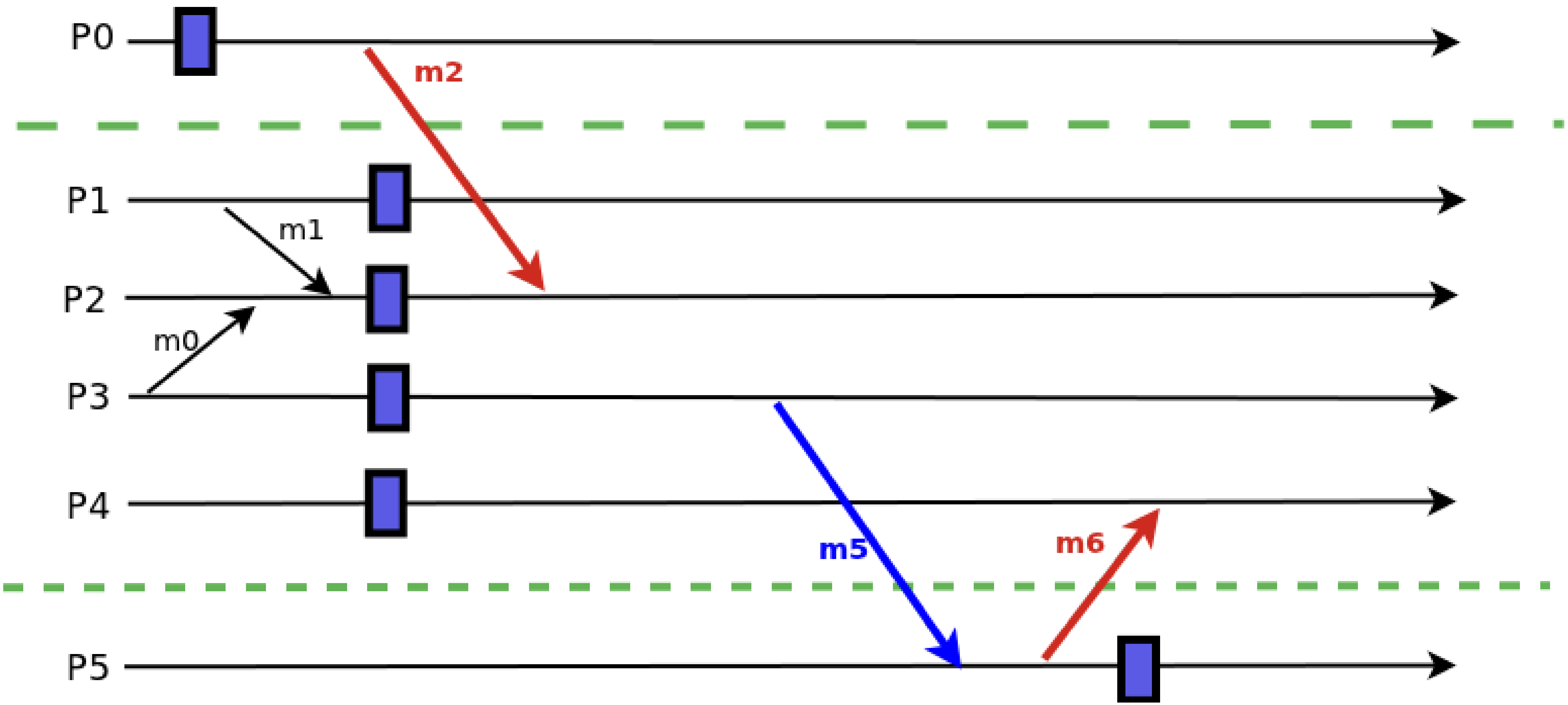
- Messages m2 and m6 can be replayed

Recovery is not that Simple



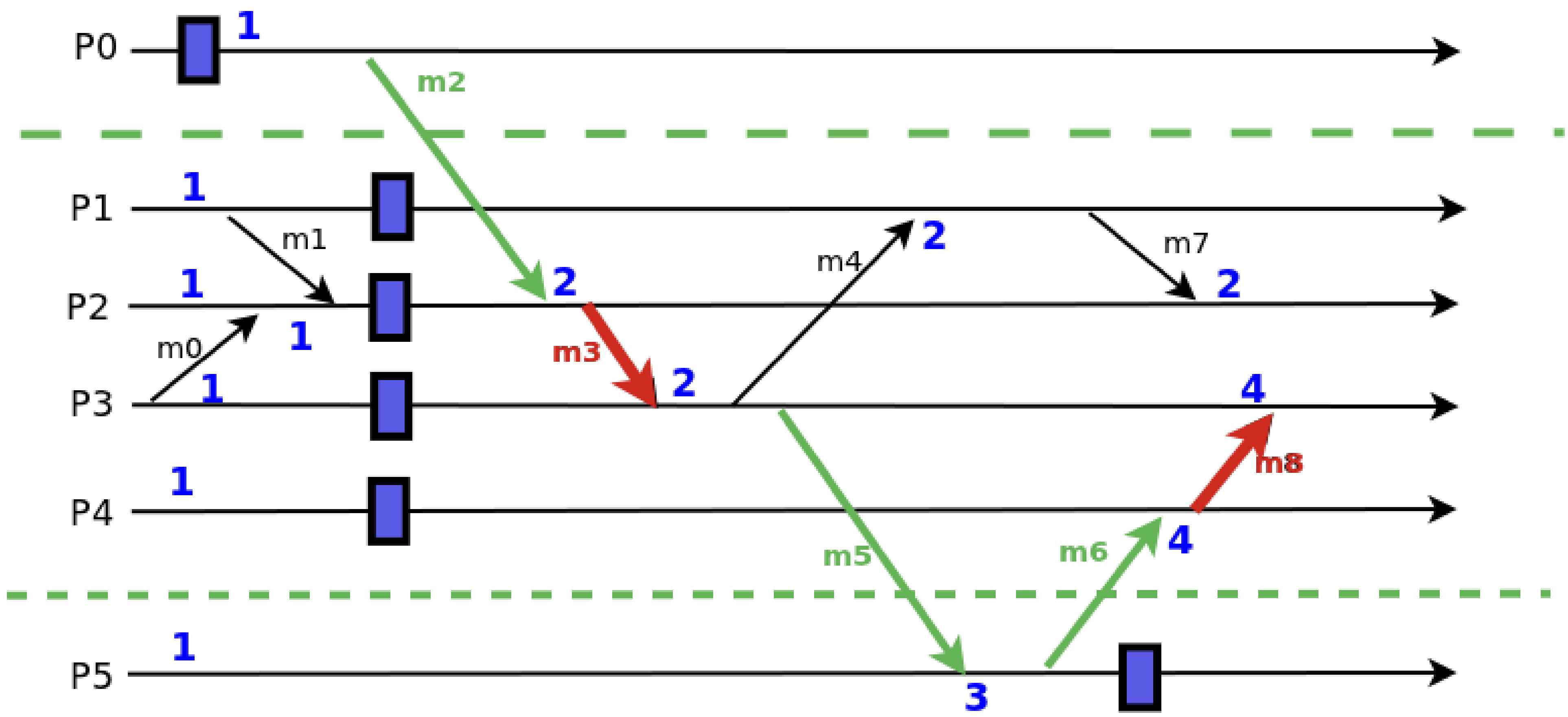
- Messages m3 and m8 can be replayed
  - Causal dependency between them
  - What if p3 is using ANY_SOURCE (anonymous reception) ?

Recovery is not that Simple



- The problem comes from m5.
  - When m6 is replayed it depends on an orphan message.

Recovery based on Phase Numbers



- Phase numbers are used to order messages replay
  - Similar to Lamport clocks
  - Incremented on inter-cluster messages

Details on the Recovery Management

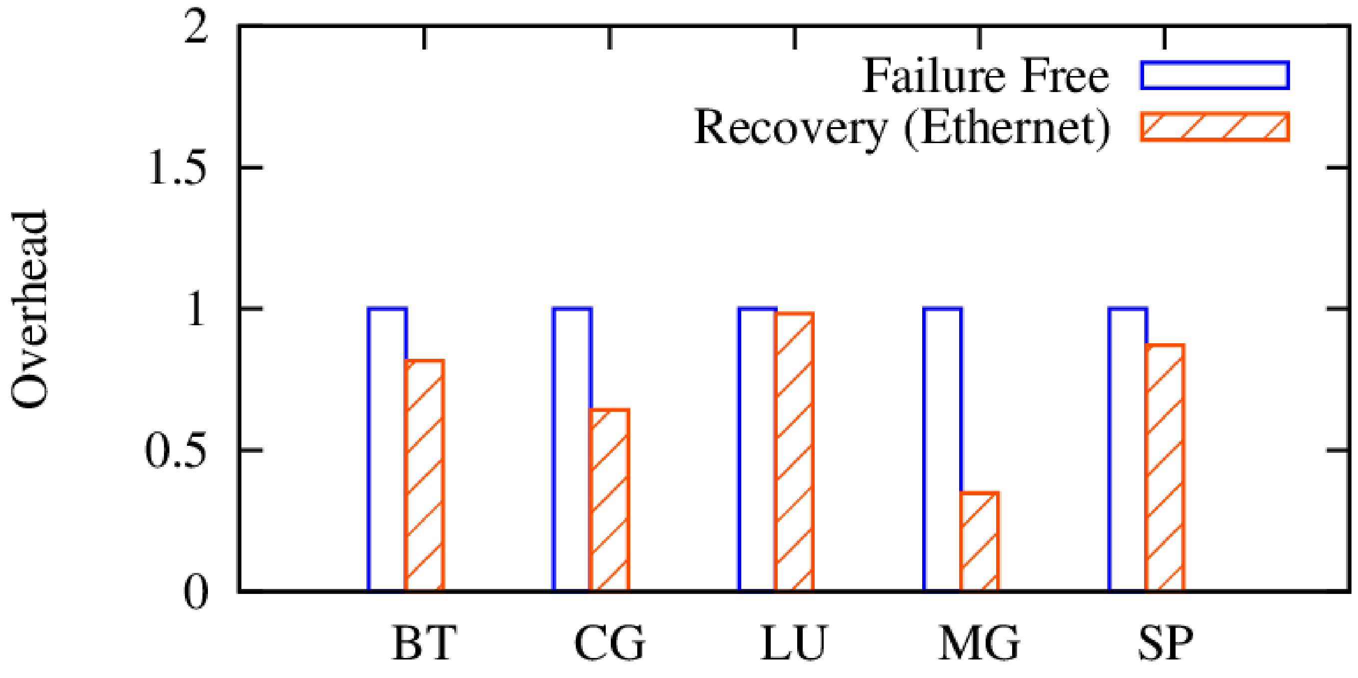
- Based on a recovery process
  - External MPI Process
    - Started when a failure occurs.
  - Orchestrates logged messages replay during recovery
    - Waits for notifications for all orphan messages in one phase
    - Allows the replay of all logged messages in the next phase
  - Needs a “global knowledge” of the application state
    - List of logged messages to replay
    - List of orphan processes
    - Hard to parallelize

Experimental Setup

- Testbed: Nancy Grid5000 cluster
  - 33 nodes equipped with 2 Intel Xeon L5420 processors (4 cores), 16 GB of Memory
  - Infiniband-20G Network interface
  - Ethernet Network interface

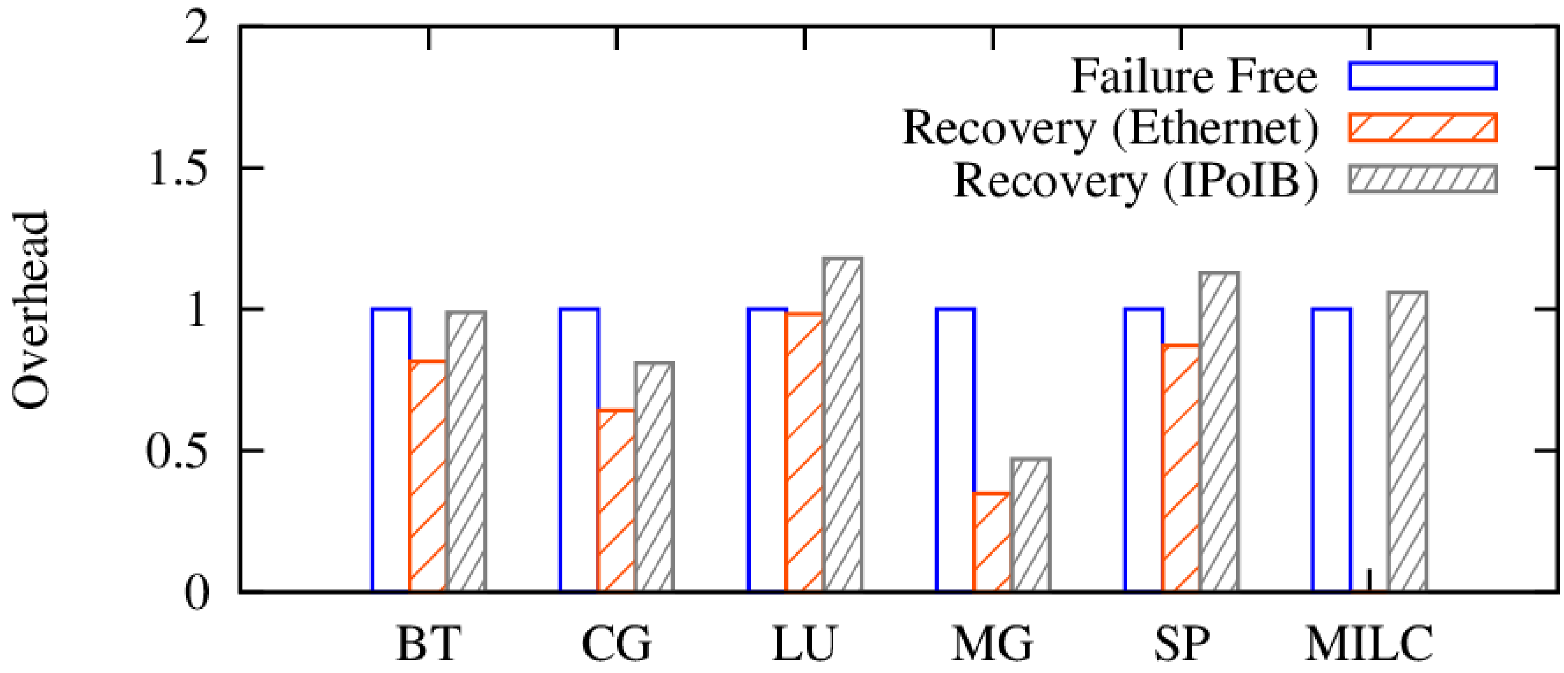
- Test description:
  - The application is run once failure free to generate the logs.
  - Application is restarted from the beginning
    - The cluster including process 0 is executed.
    - Inter-cluster messages are replayed from the logs.

Evaluation of Recovery (TCP)
NAS - Class D - 256 processes



- Reasons for performance improvement: inter-cluster communications
  - Recovering processes send notifications instead of real messages
  - Messages are ready to be received

Evaluation of Recovery (IP over IB)
NAS - Class D - 256 processes

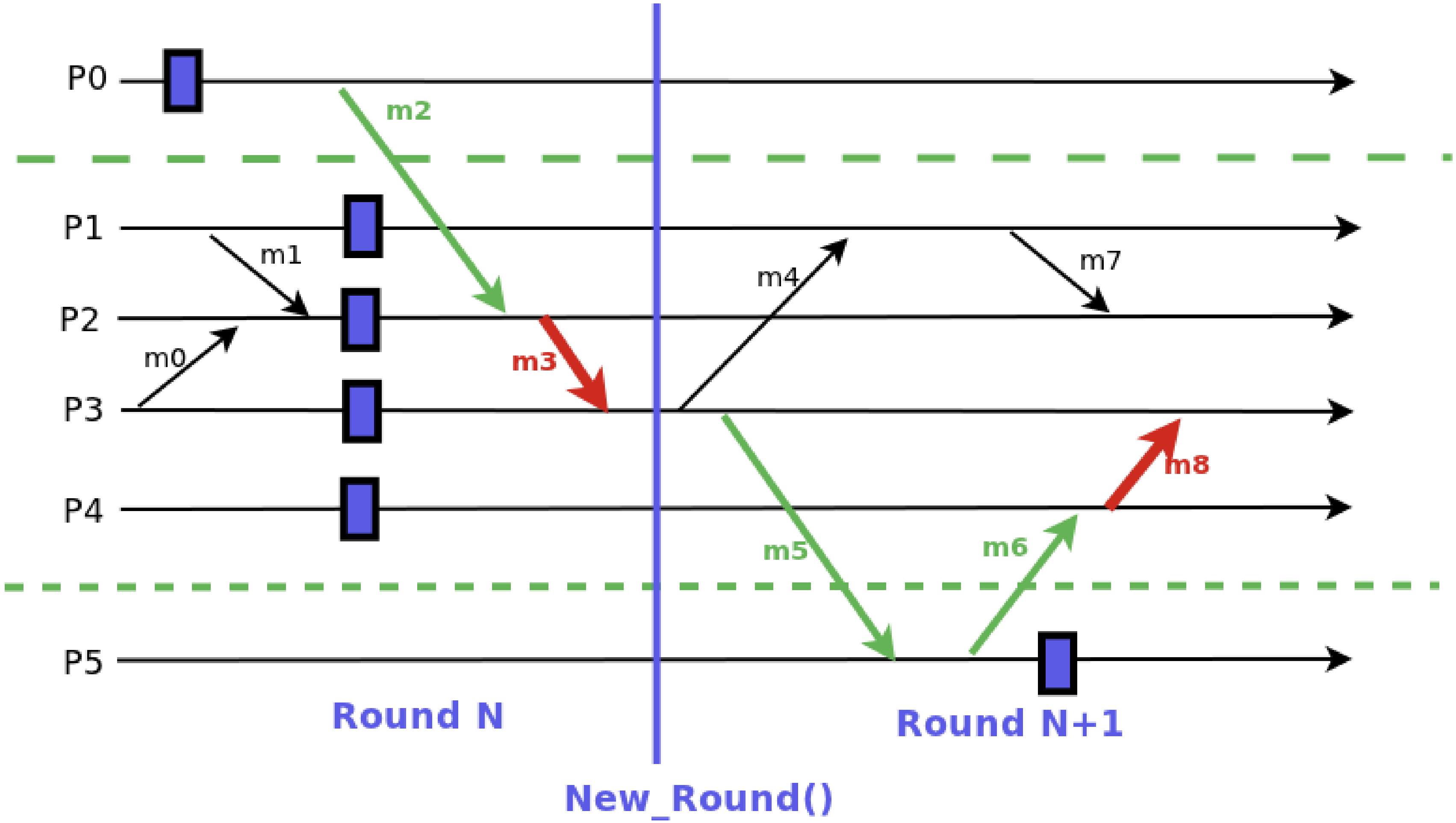


- Lower speed up
  - Overhead for some applications
- The recovery process becomes the bottleneck

Send-Deterministic Aware Applications

- Goal: fully distributed recovery
- Comments:
  - Problems come from anonymous receptions (ANY_SOURCE).
  - No problems during a failure free execution.
  - The programmer knows:
    - Its program is composed of implicit rounds (explicit or implicit barriers)
    - Which messages can be received in each round.
- Proposition:
  - Provide a way for the programmer to make the rounds explicit:
    - New_Round()

Send-Deterministic Aware Applications



- Messages m3 and m8 can be ordered based on their round.
  - No need for a recovery process.

Conclusion

- HydEE: rollback-recovery protocol for large scale MPI applications
  - Failure containment
    - Combines coordinated checkpointing and message logging
    - No information saved on stable storage (except checkpoints)
  - Implemented in MPICH2
  - Good performance in failure free execution
  - Good performance in recovery execution
    - Scalability issue: centralized recovery process
    - Send-deterministic aware applications
      - Executions rounds

Future work

- Prototype implementation
  - Coordinated checkpointing
  - Distributed recovery
  - Partial restart
- Improving data management
  - Topology-aware checkpointing
  - Asynchronous sender-based message logging
- Integration with application-level checkpointing

On Distributed Recovery for Send-Deterministic-Aware MPI Applications

Thomas Ropars, Amina Guermouche, Marc Snir and Franck Cappello

informatics mathematics

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Example of Send-Deterministic Pattern

```
for(i=0; i<nb_recv; i++){
    MPI_Irecv(T[i], ..., i, ...);
}
for(i=0; i<nb_send; i++){
    MPI_Send(..., i, ...);
}
for(i=0; i<nb_recv; i++){
    MPI_Waitany(...);
}
```

Clustering Configuration

	Nb Clusters	Size of cluster to restart	Avg % of ps to restart	Logging (GB)
BT	5	63	21.78%	143/791 (18.09%)
CG	16	16	6.25%	440/2318 (18.98%)
FT	2	129	50%	431/860 (50.19%)
LU	8	32	12.5%	44/337 (13.26%)
MG	4	64	25%	13/66 (19.63%)
SP	6	32	18.56%	289/1446 (20.04%)