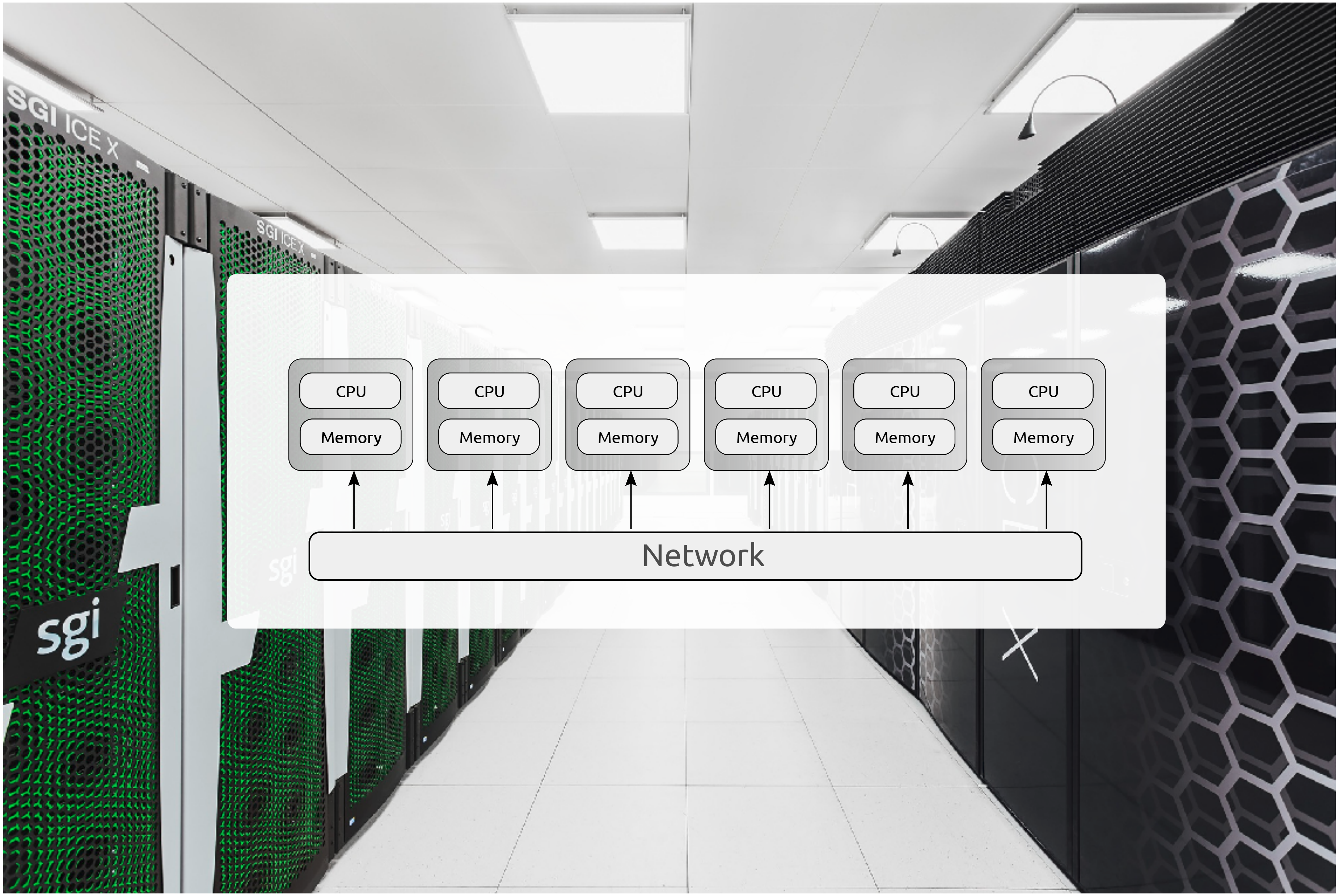
# State-Space Reduction of Non-deterministically Synchronizing Systems Applicable to Deadlock Detection in MPI

Stanislav Böhm[1], Ondřej Meca[1], Petr Jančar[2]

[1] IT4Innovations - National Supercomputing Center, Czech Republic
[2] FEI, Technical University of Ostrava, Czech Republic

SGI ICE X
sgi
ICE X


CPU
Memory
CPU
Memory
CPU
Memory
CPU
Memory
CPU
Memory
CPU
Memory
Network

## MPI - Message Passing Interface

```
MPI_Init(&argc, &argv);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int size;
MPI_Comm_size(MPI_COMM_WORLD, &size);
int value;
if (rank == 0) {
    value = 0;
    MPI_Send(&value, 1, MPI_INT,
             (rank + 1) % size,
             0, MPI_COMM_WORLD);
}
for (int t=0; t < 1000; t++) {
    MPI_Recv(&value, 1, MPI_INT,
                 MPI_ANY_SOURCE,
                 MPI_ANY_TAG,
                 MPI_COMM_WORLD,
                 MPI_STATUS_IGNORE);

    MPI_Send(&value, 1, MPI_INT,
                 (rank + 1) % size,
                 0, MPI_COMM_WORLD);
}
MPI_Finalize();
```

## State-Space Reduction of Non-deterministically Synchronizing Systems Applicable to Deadlock Detection in **MPI**

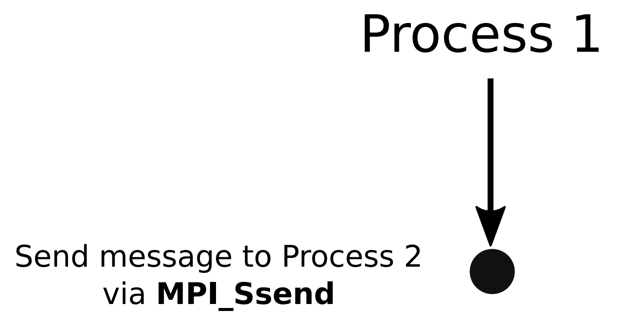
## State-Space Reduction of **Non-deterministically Synchronizing Systems** Applicable to Deadlock Detection in **MPI**

## MPI_Ssend

Synchronized send - waits for a matching receive

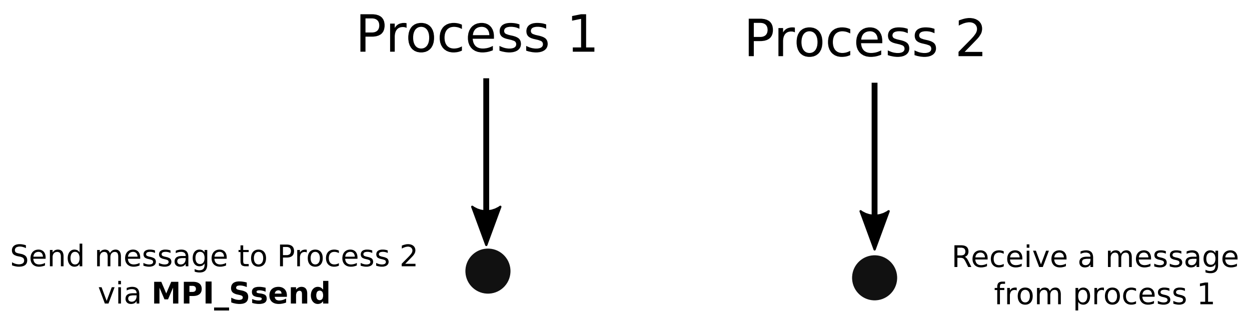
# **MPI_Ssend**

Synchronized send - waits for a matching receive



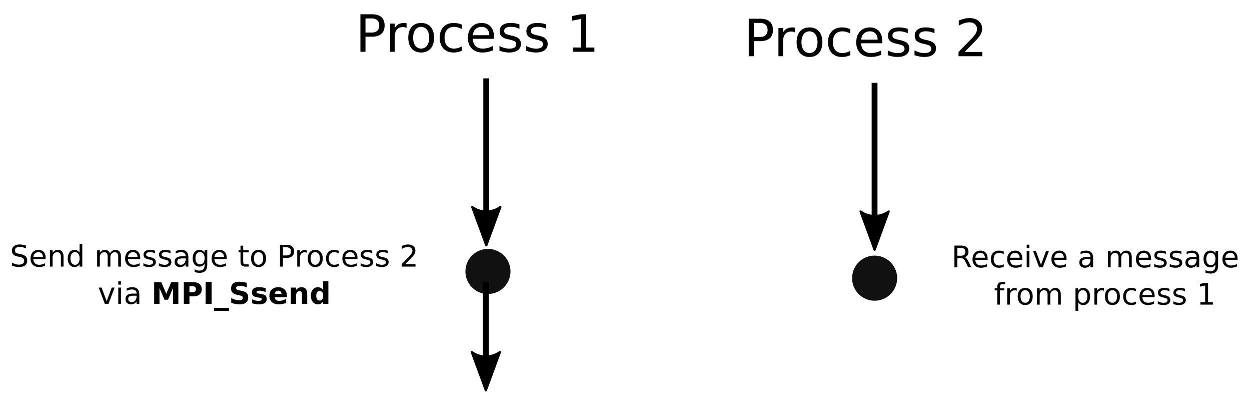
## **MPI_Ssend**

Synchronized send - waits for a matching receive


Process 1
Process 2
Send message to Process 2 via **MPI_Ssend**
Receive a message from process 1

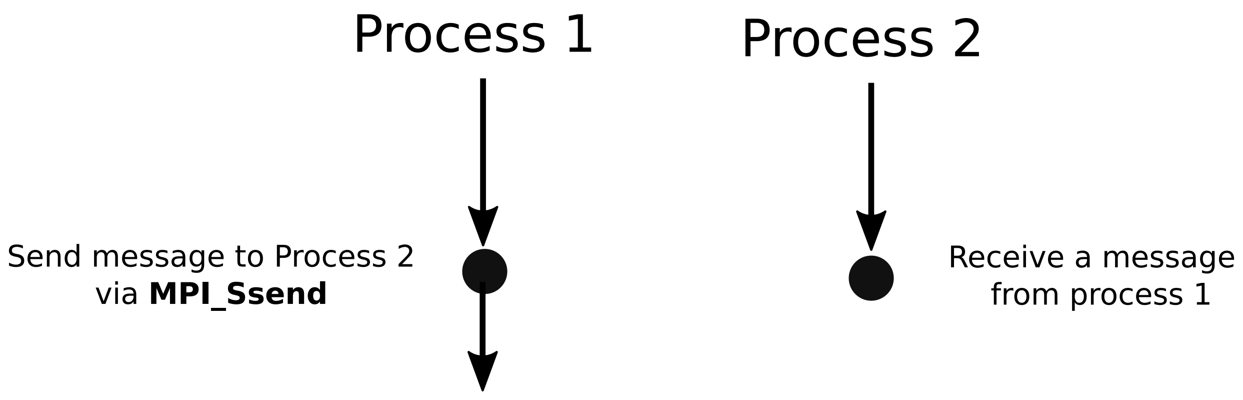
# MPI_Ssend

Synchronized send - waits for a matching receive


Process 1
Process 2
Send message to Process 2 via MPI_Ssend
Receive a message from process 1

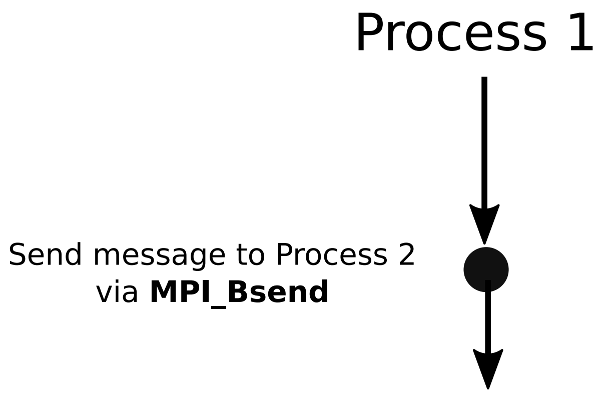
## **MPI_Ssend**

Synchronized send - waits for a matching receive


Process 1
Process 2
Send message to Process 2 via **MPI_Ssend**
Receive a message from process 1

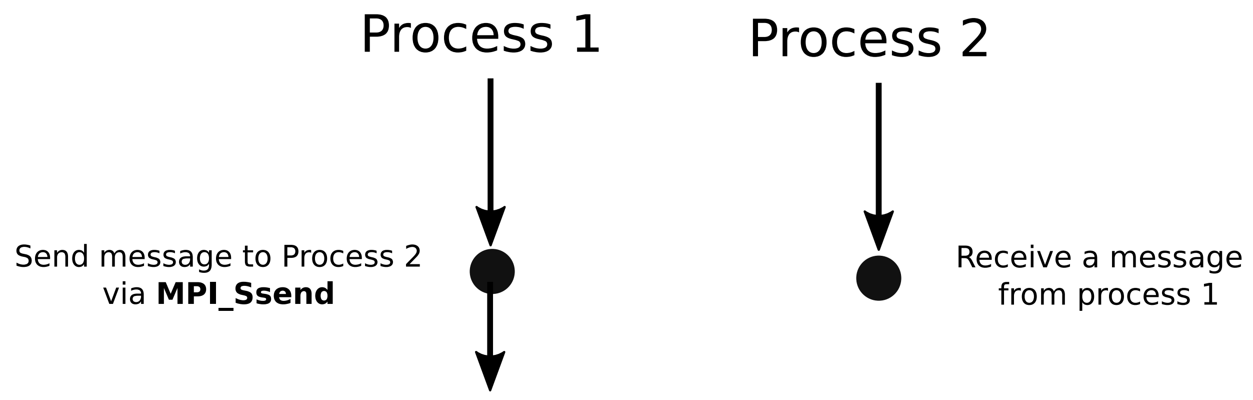
## **MPI_Bsend**

Buffered send - do not wait for receive



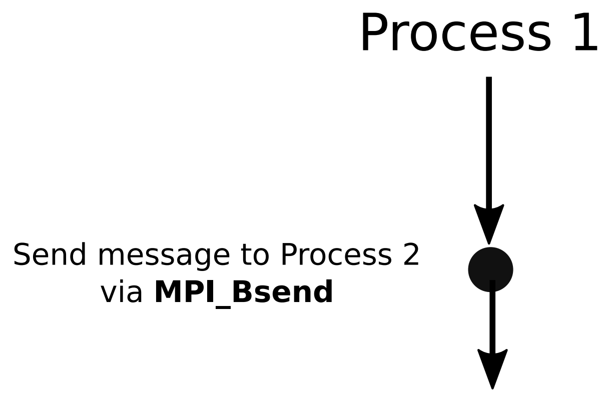
## MPI_Ssend

Synchronized send - waits for a matching receive



## MPI_Bsend

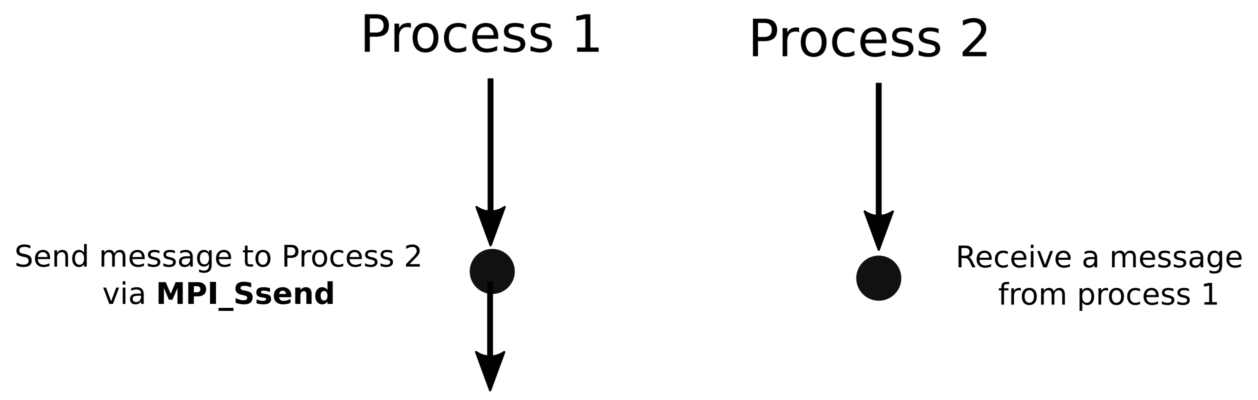
Buffered send - do not wait for receive



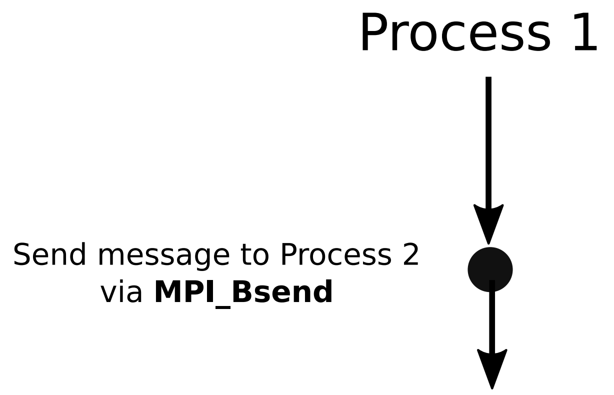
## MPI_Send

Standard send

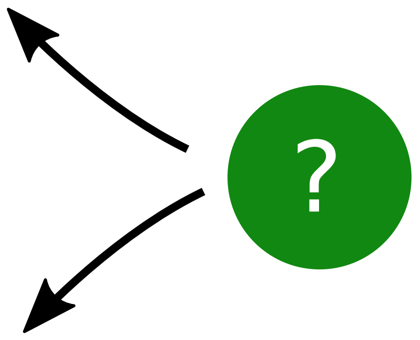
## MPI_Ssend
Synchronized send - waits for a matching receive


Process 1
Process 2
Send message to Process 2 via **MPI_Ssend**
Receive a message from process 1

## MPI_Bsend
Buffered send - do not wait for receive



## MPI_Send
Standard send



Nondeterministic choice

rendezvous / eager

| Process 1 | Process 2 |
|---|---|
| MPI_Send(to=2)<br>MPI_Recv(from=2) | MPI_Send(to=1)<br>MPI_Recv(from=1) |

*Naive approach:*

Analyze all **MPI_Send**s as randezvous

~~*Naive approach:*~~

~~Analyze all **MPI_Send**s as randezvous~~

Not complete

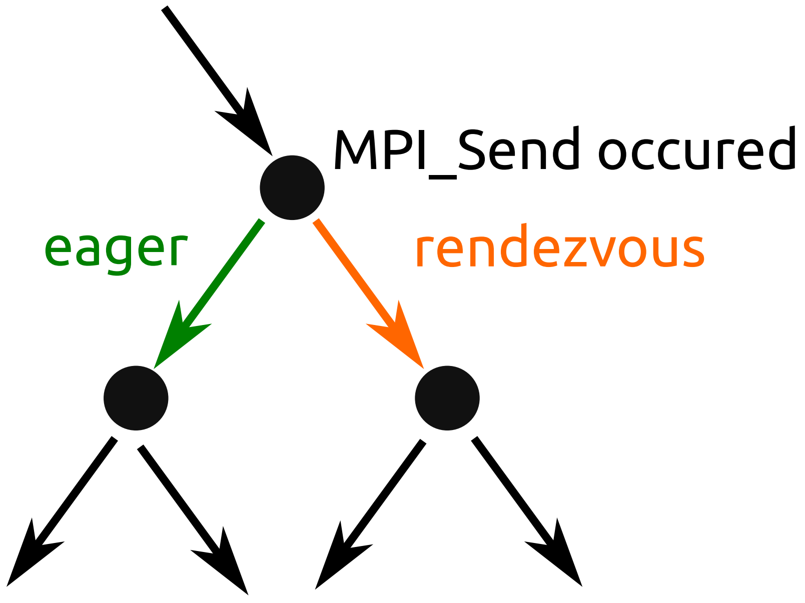
~~*Naive approach:*~~

~~Analyze all **MPI_Send**s as rendezvous~~

Not complete

*Approach 2:*

Encode **MPI_Send** choices into state space



## Observation

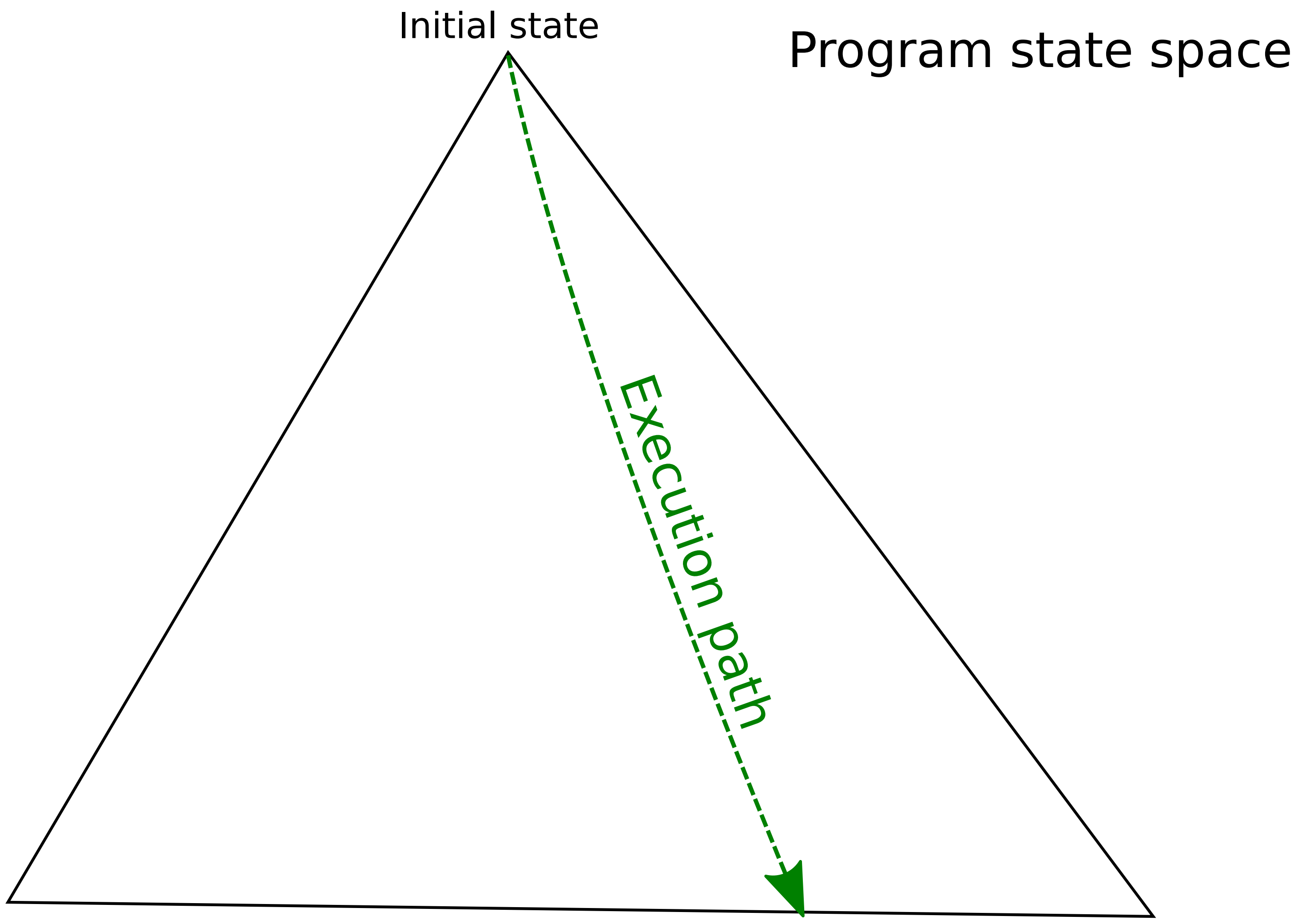
Choices inside MPI_Sends: no new behavior (except deadlocks)

## New approach

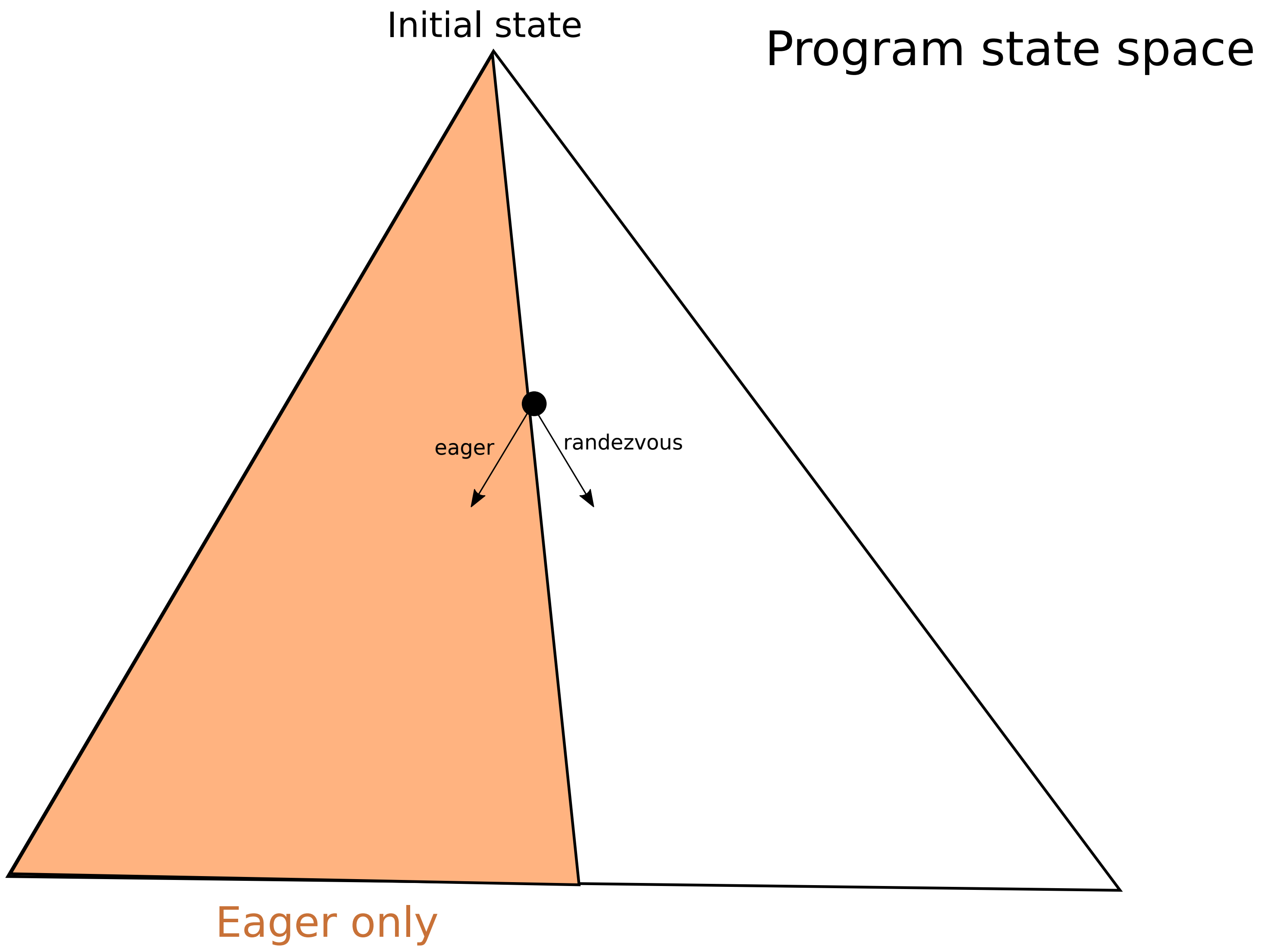
(1) Build a state space while considering only *eager* MPI_Send

(2) Find missing deadlocks

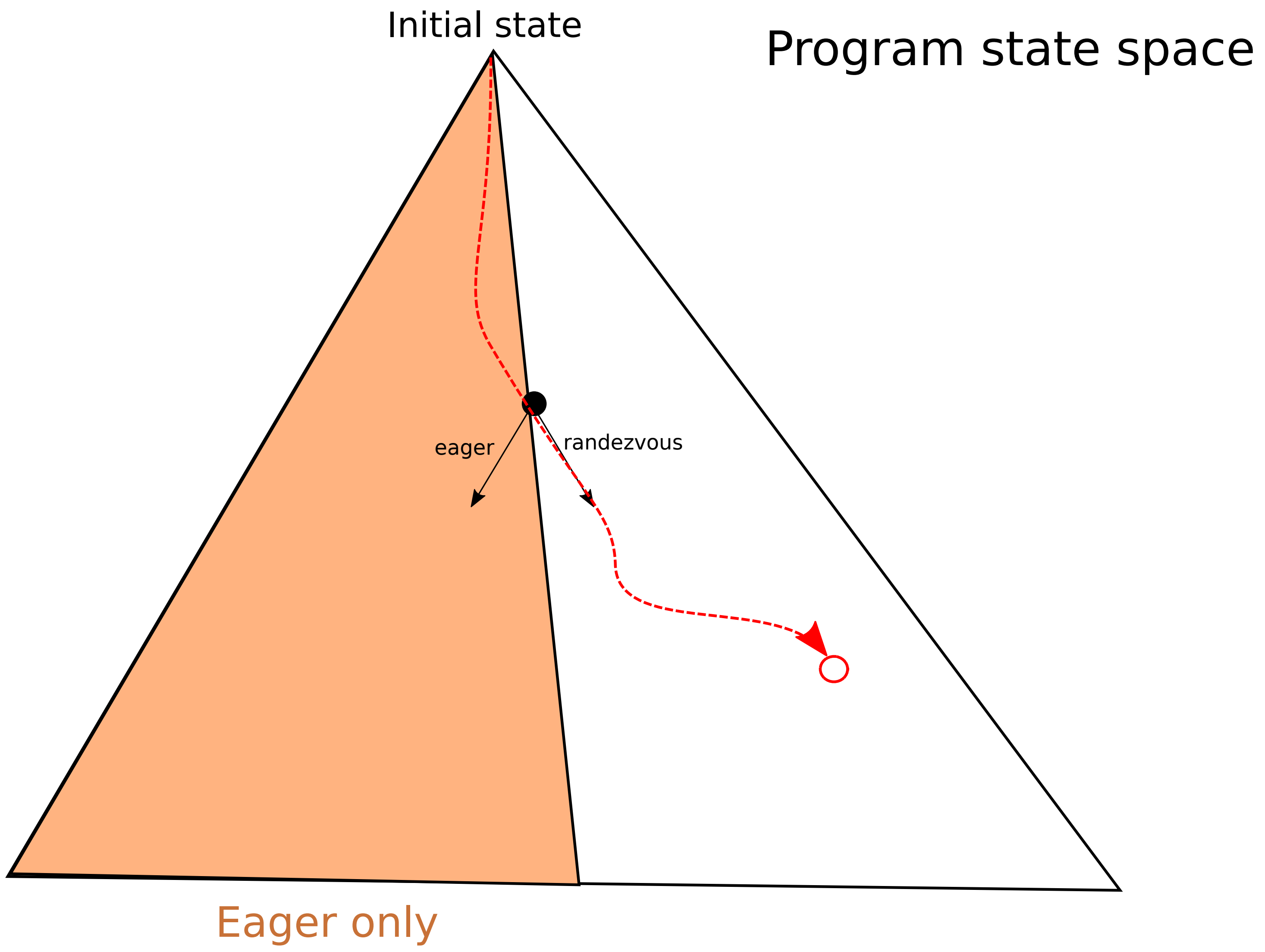
## New approach



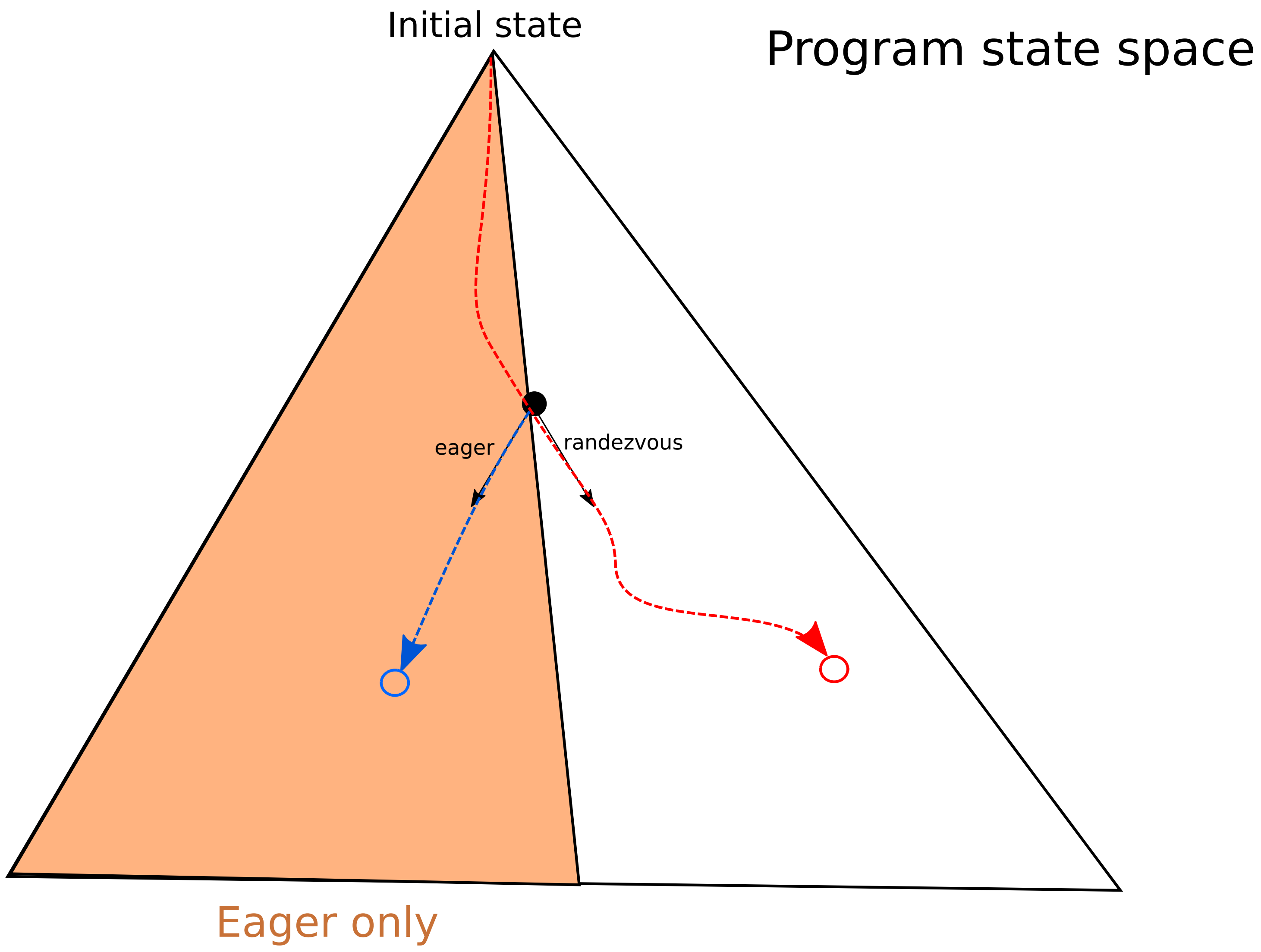
## New approach



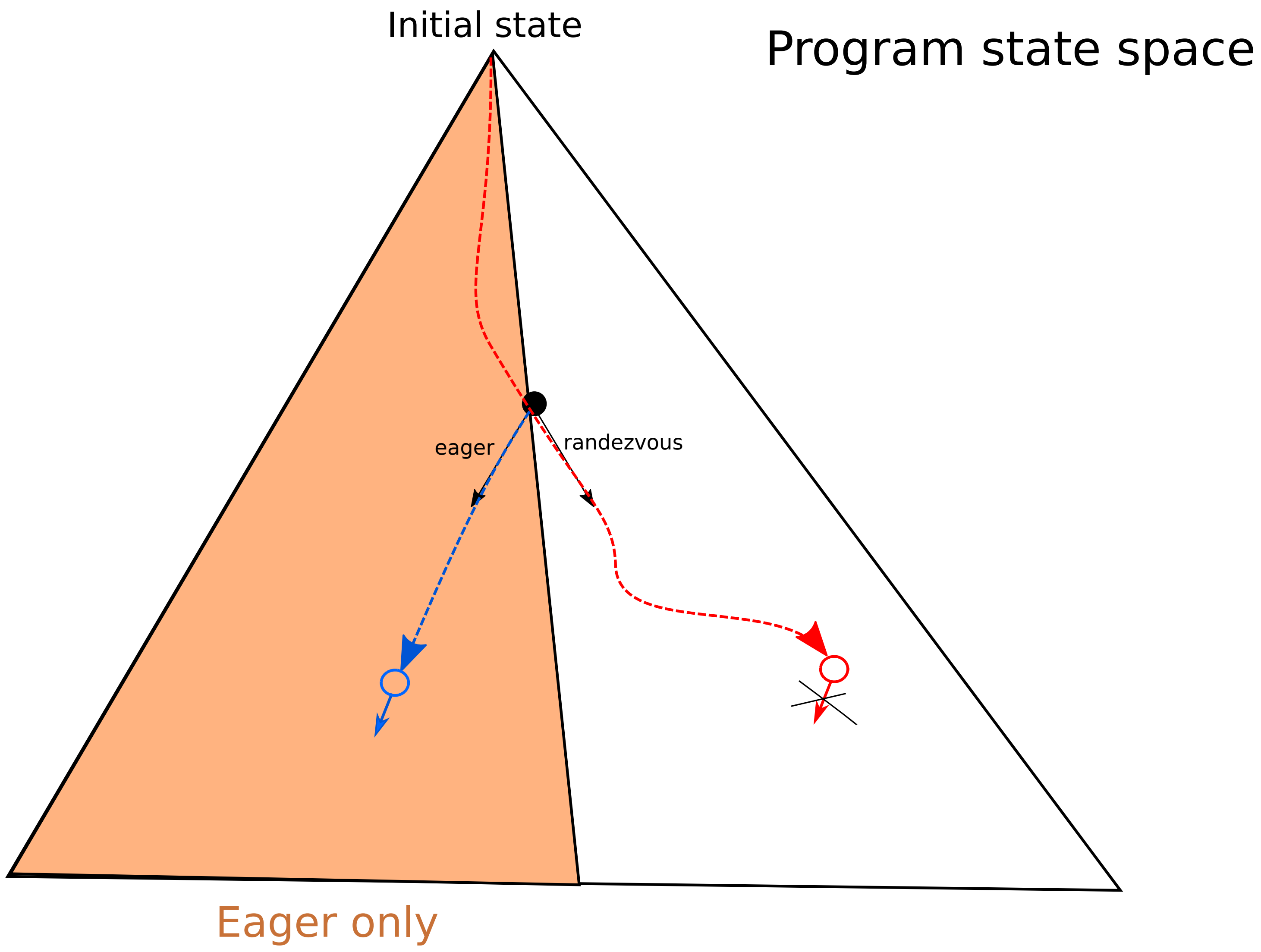
## New approach



# New approach



# New approach

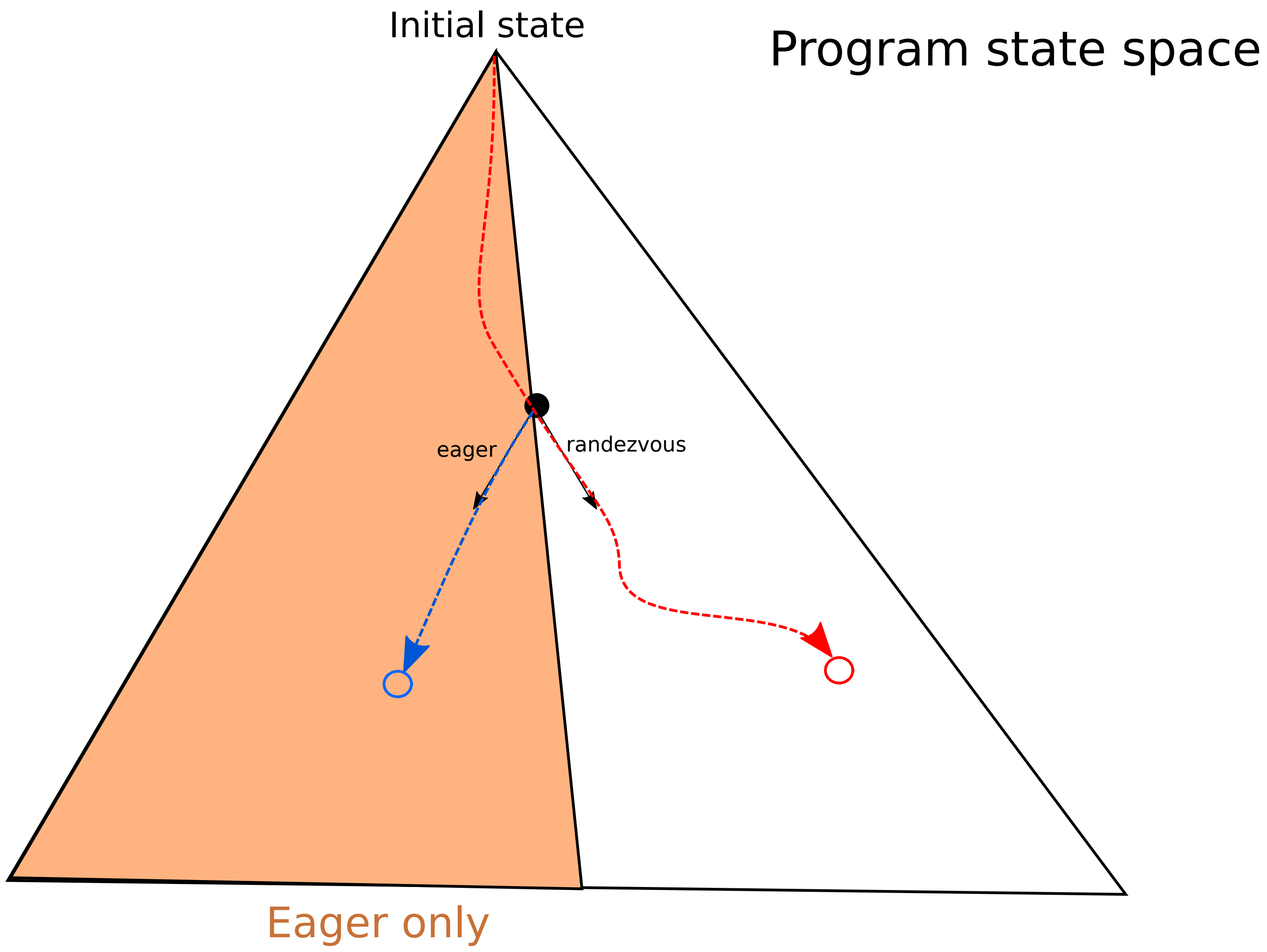


## Approach 3

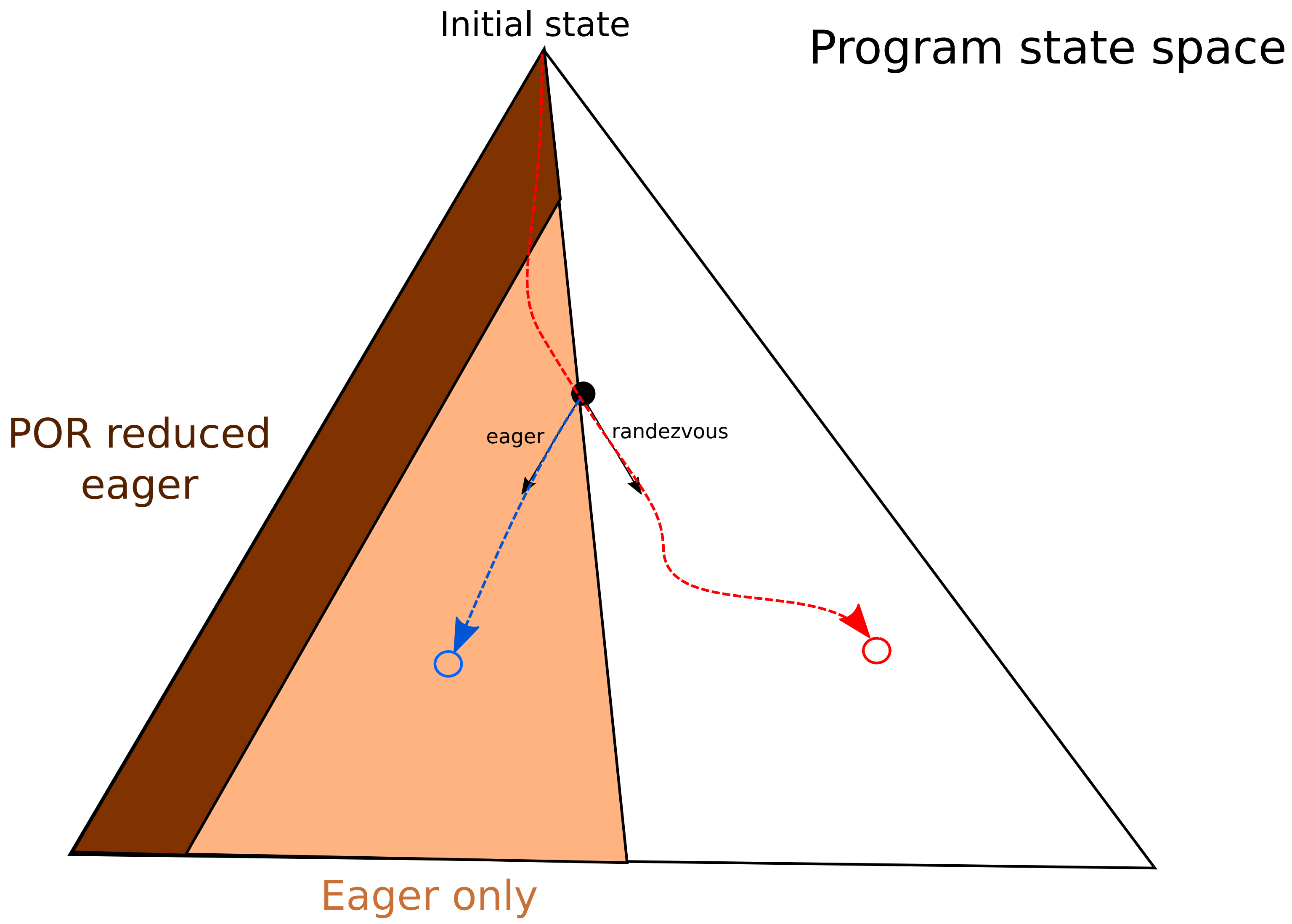
(1) Build a **parial-order reduced** state space while considering
 only *eager* MPI_Send

(2) Find missing deadlocks

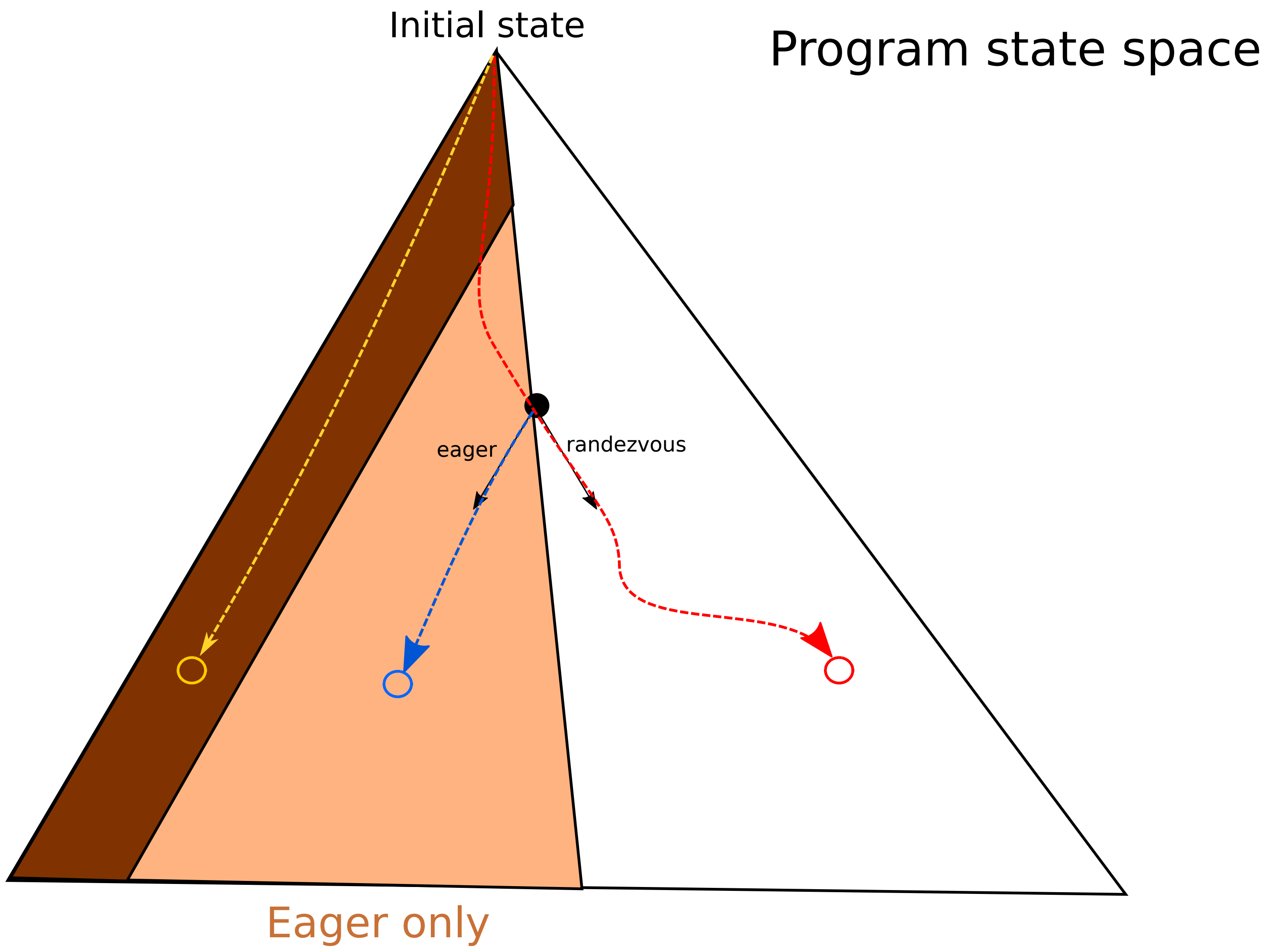
## Approach 3


Initial state
Program state space
eager
randezvous
Eager only

# Approach 3


Initial state
Program state space
POR reduced eager
eager
randezvous
Eager only

# Approach 3



Initial state
Program state space
eager
randezvous
Eager only

## Problem

### Input

**R** - a system obtained by applying any* POR method on a system **T**
any POR that preserves Mazurkiewicz Traces
**I** - the independent set used in reduction
**C** - set of 'candidates' that may nondetermistically synchronize

### Output

Is there a deadlock in **T** while considering synchronizations of **C** ?

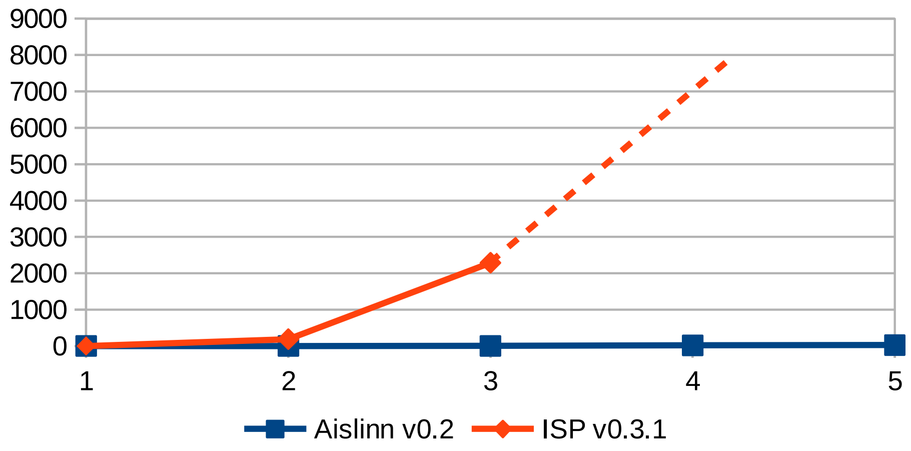
## Aislinn

http://verif.cs.vsb.cz/aislinn/

Benchmark: Workers

| # of verified processes | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| Aislinn v0.2 | 0.7 | 1.1 | 4.3 | 17.6 | 26.2 |
| ISP v0.3.1 | 2.1 | 191.9 | 2287.2 | >7200 | >7200 |

execution times in seconds



Approach 3 compared to Approach 2 (with optimizations)

**3x - 15x** speedup