Scheduling Scientific Experiments on the Rosetta/Philae Mission

G. Simonin - C. Artigues - E. Hebrard - P. Lopez

CP 2012

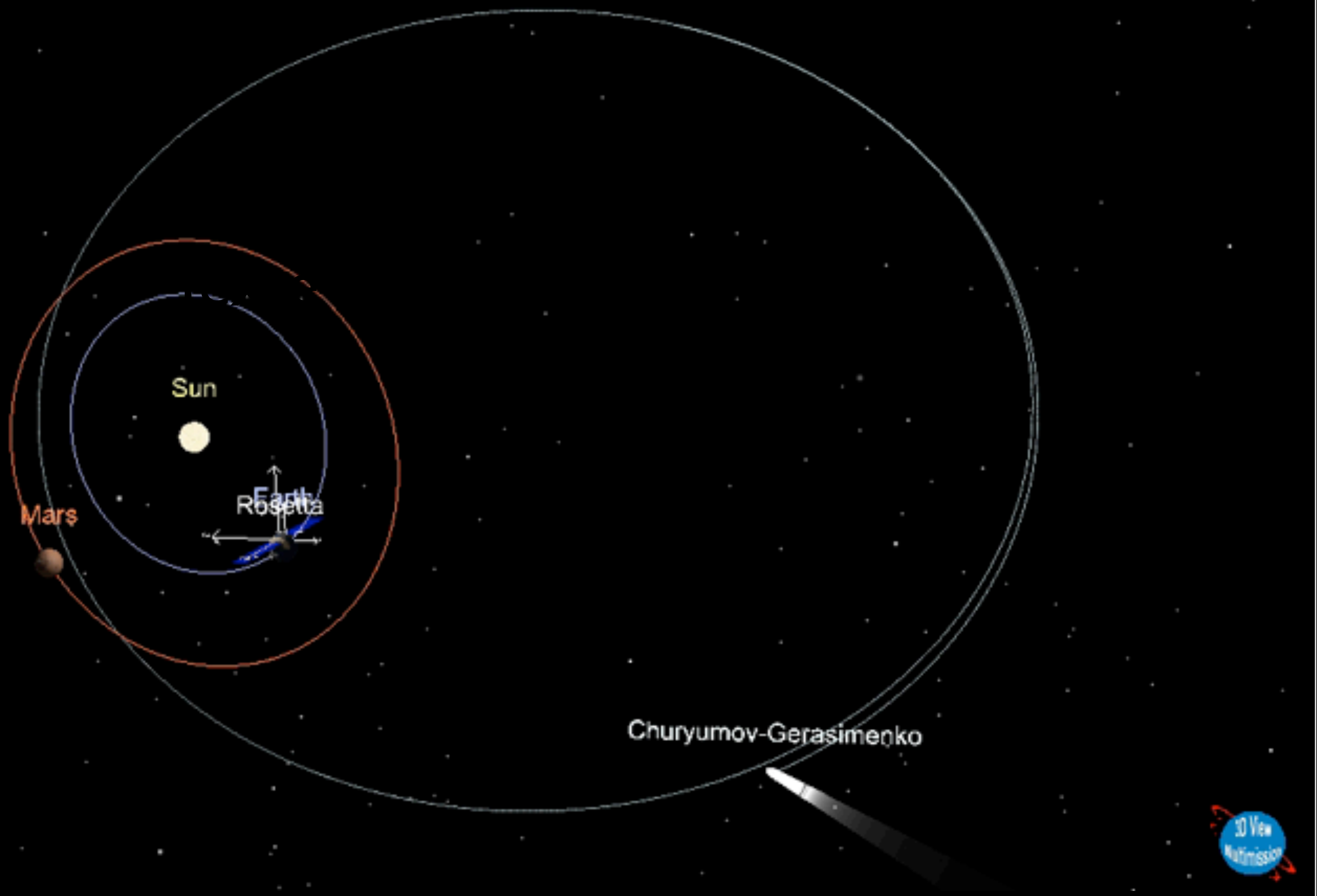
LAAS-CNRS
Toulouse, France

09/10/2012

Context

ROSETTA-PHILAE

Spacecraft Rosetta

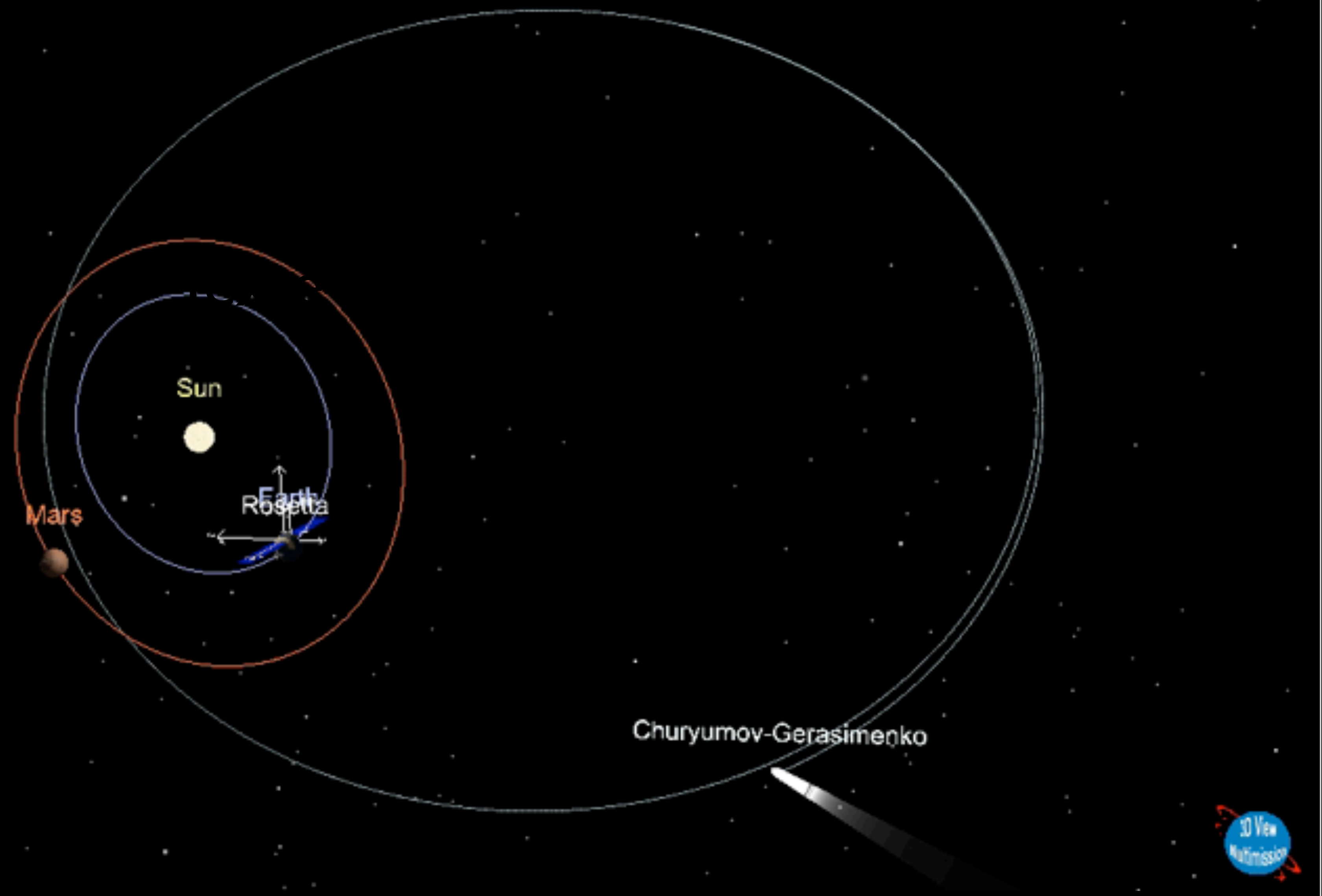


Context

ROSETTA-PHILAE

Spacecraft Rosetta

ESA/CNES mission
Sent by Ariane 5 on 2004



Context

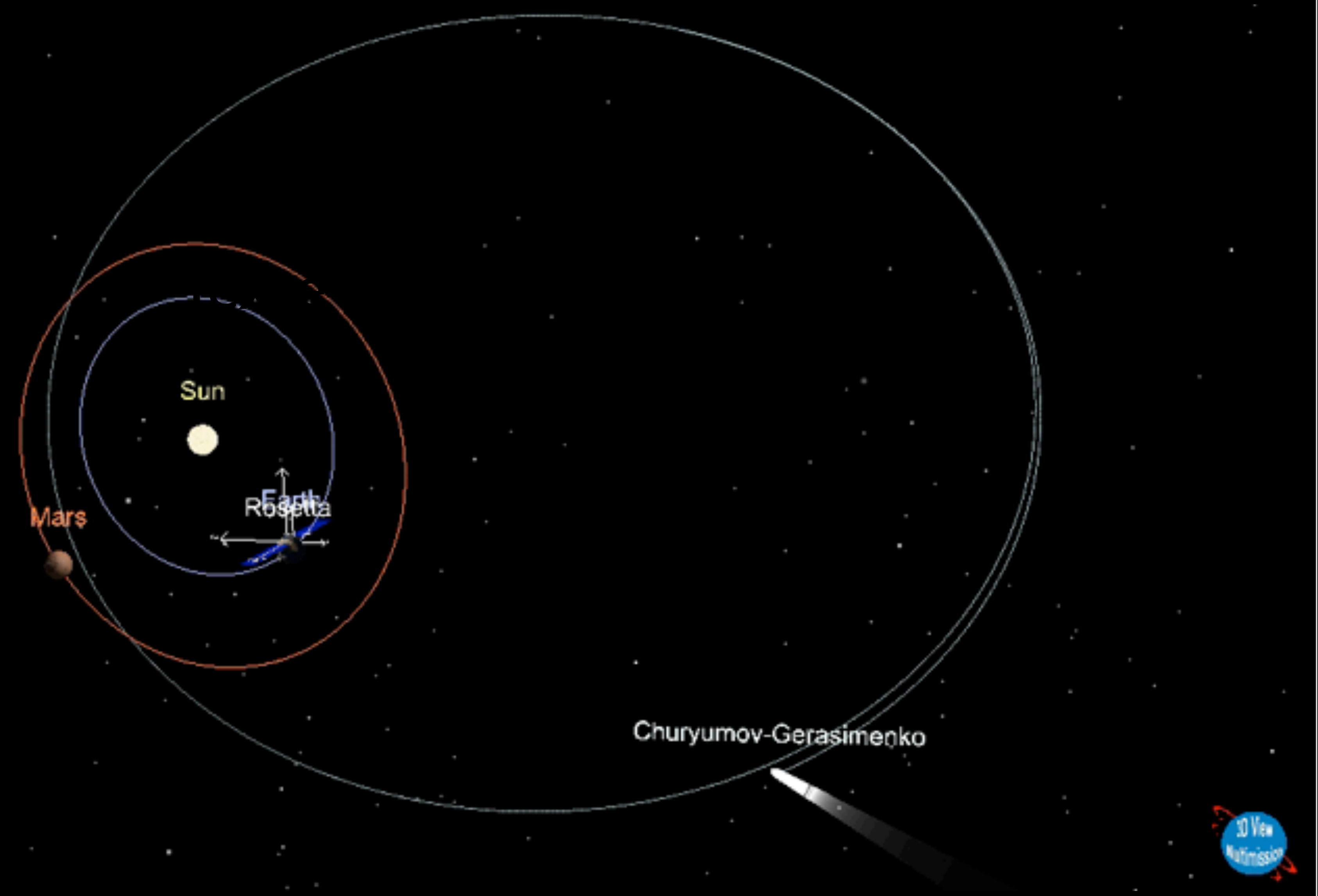
ROSETTA-PHILAE

Spacecraft Rosetta

Objective (2014):

Land on a comet

Analyze its surface



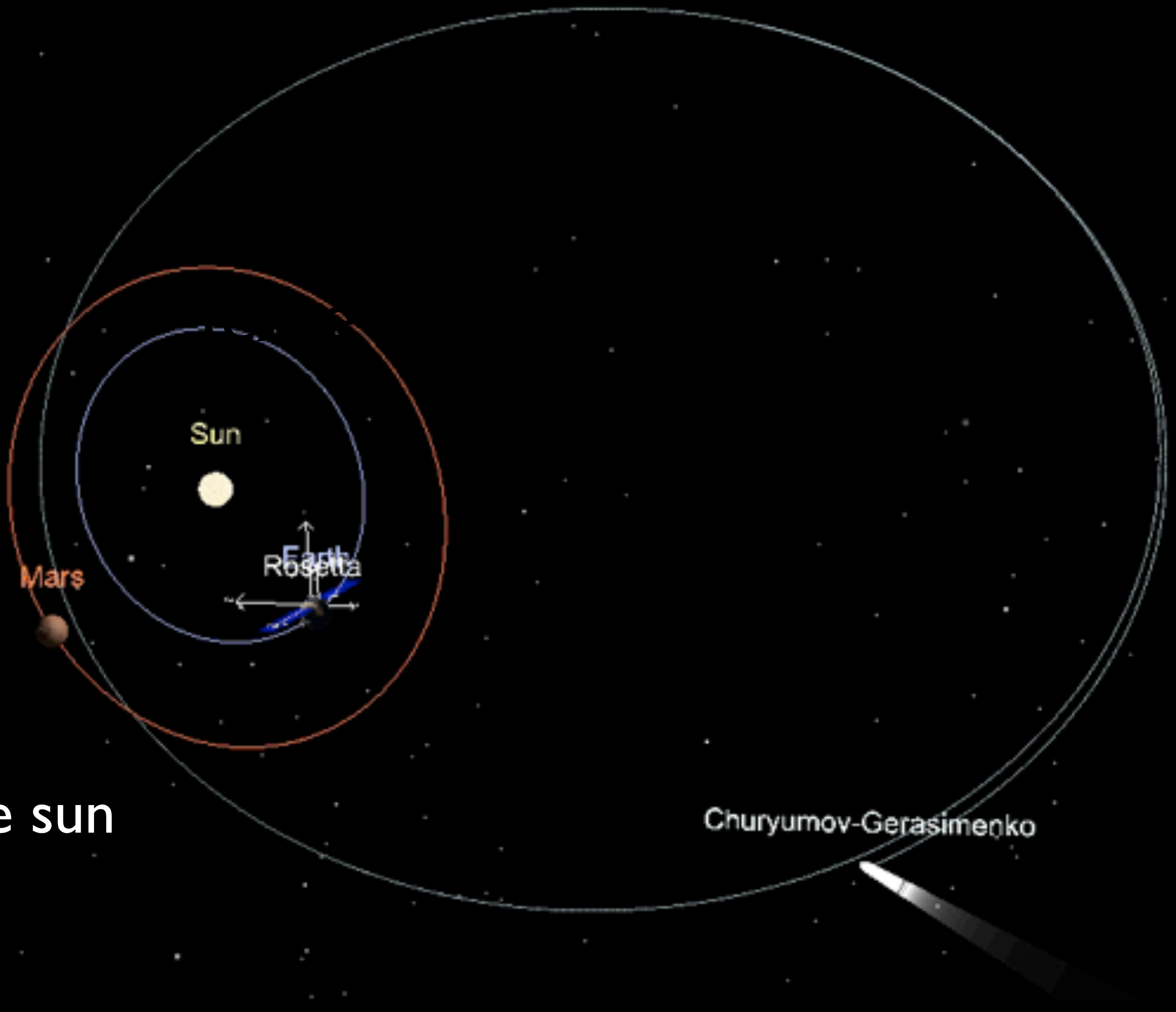
Context

ROSETTA-PHILAE

Spacecraft Rosetta

Trajectory:

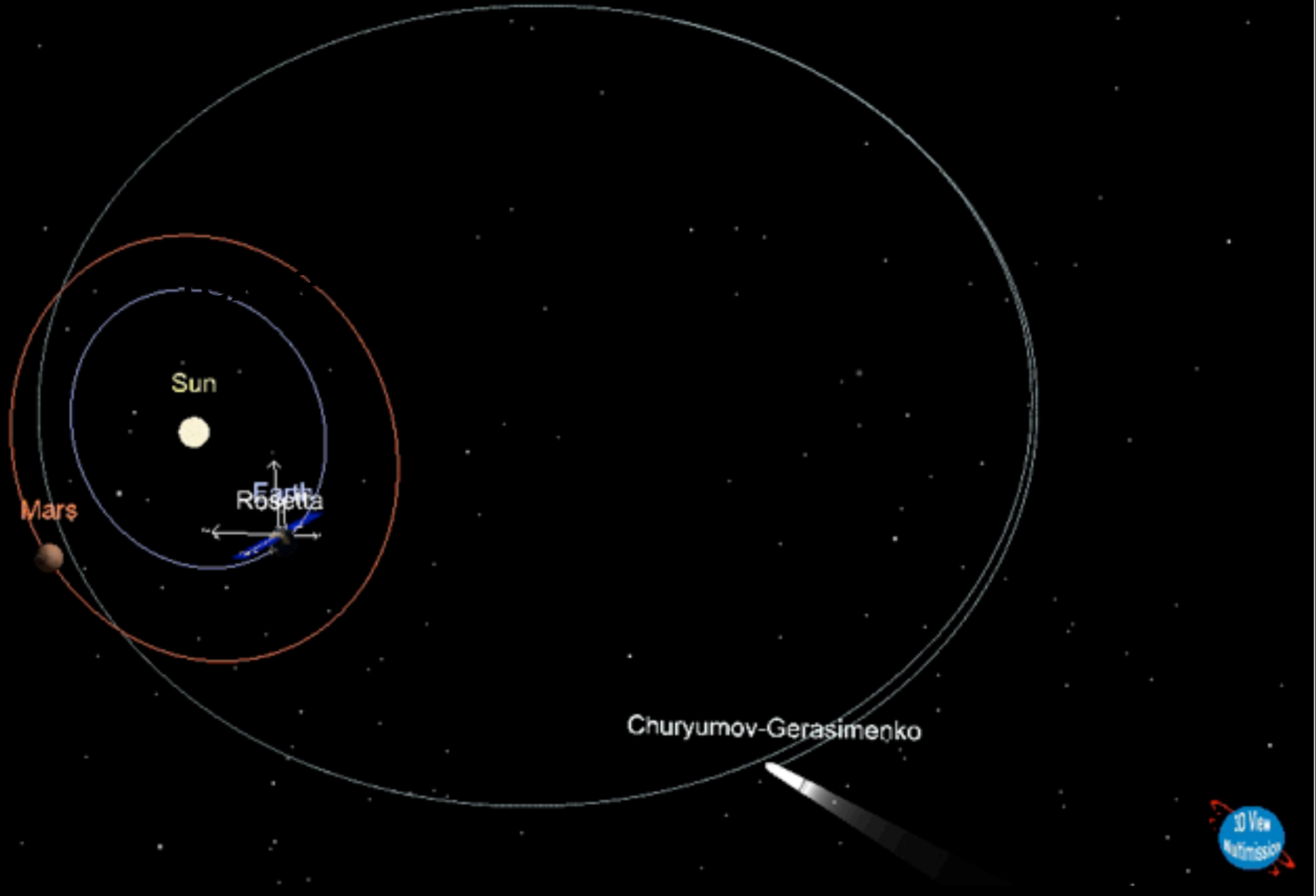
- 4 gravitational accelerations
- Intercept the comet far from the sun



Context

ROSETTA-PHILAE

Spacecraft Rosetta



The comet:

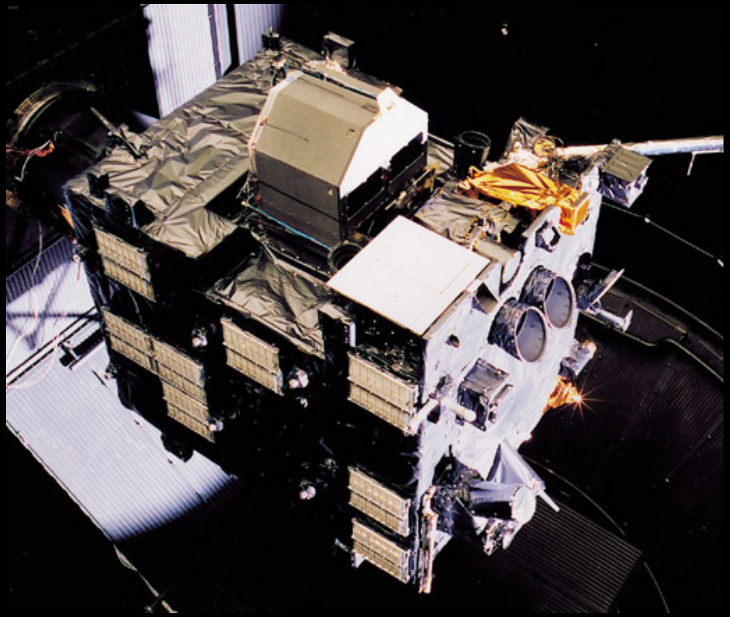
Churyumov-Gerasimenko
Mass of ice and snow powder
Limited knowledge
Get information from the past

ROSETTA-PHILAE

Spacecraft Rosetta

ROSETTA-PHILAE

Spacecraft Rosetta

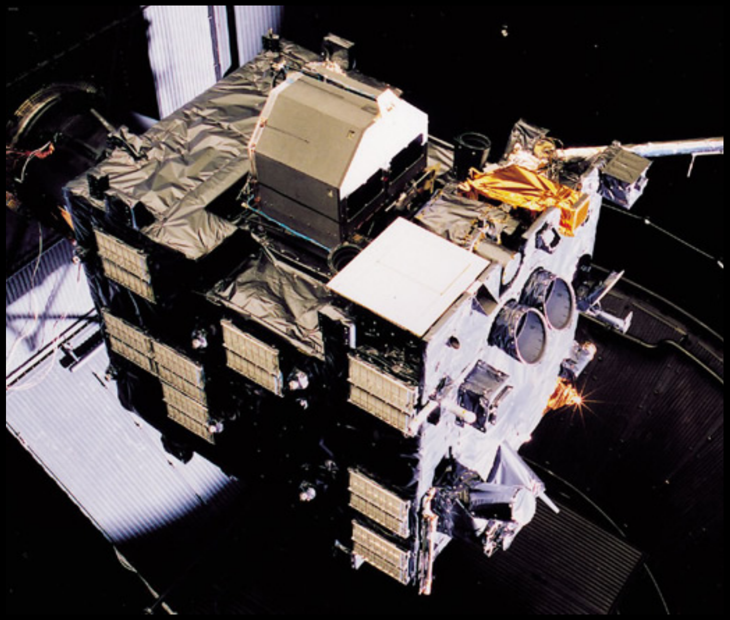


Lander Philae

Rosetta stays in orbit and deploys Philae

ROSETTA-PHILAE

Spacecraft Rosetta

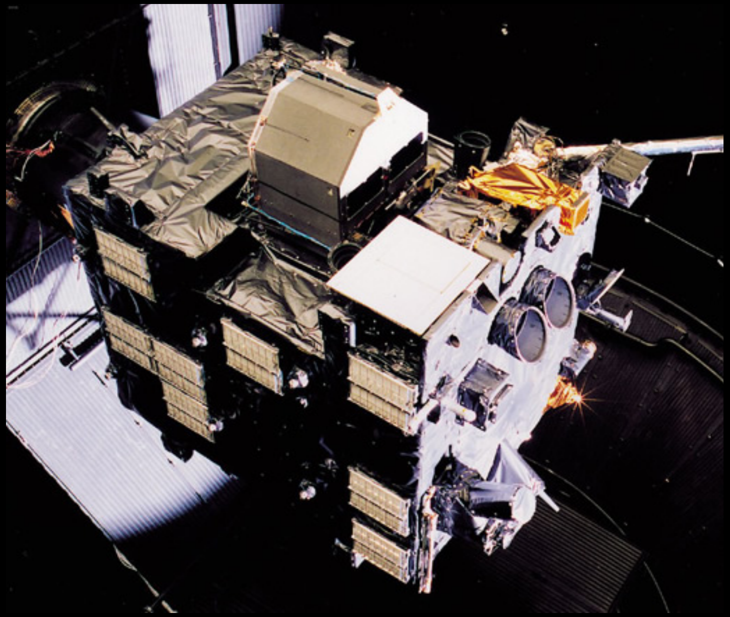


Lander Philae

Rosetta stays in orbit and deploys Philae
3 phases of experimentation:
SDL (Separation - Descent - Landing)
FSS (First Science Sequence)
LTS (Long Term Science)

ROSETTA-PHILAE

Spacecraft Rosetta



Lander Philae

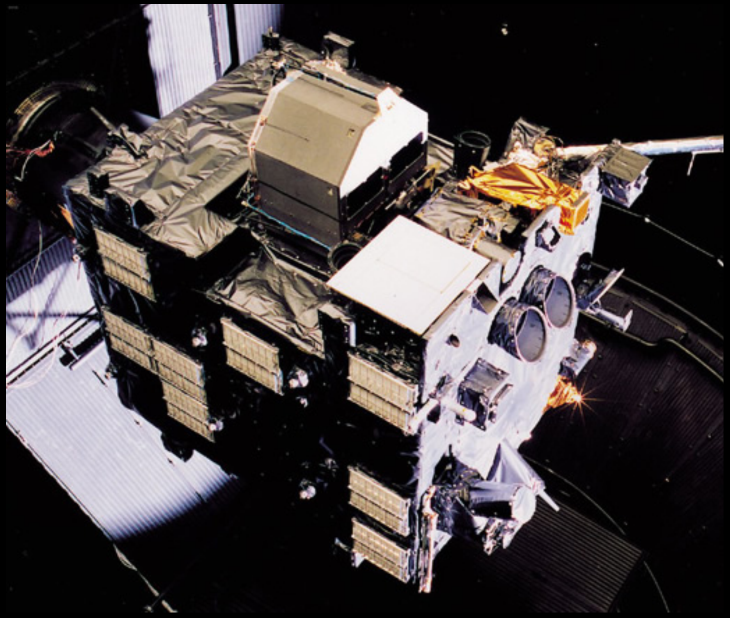
Instruments:
- Civa
- Sd2
- Consert
- Romap
- Ptolémé
- Cosac
- Apxs
- Sesame
- Mupus

Rosetta stays in orbit and deploys Philae

3 phases of experimentation:
- SDL (Separation - Descent - Landing)
- FSS (First Science Sequence)
- LTS (Long Term Science)

ROSETTA-PHILAE

Spacecraft Rosetta



Lander Philae

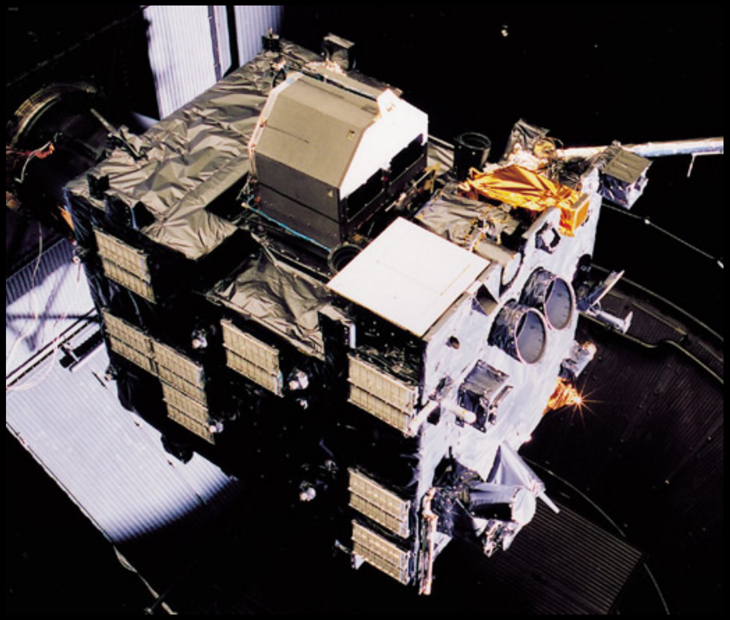
Instruments:

- Civa
- Sd2
- Consert
- Romap
- Ptolémé
- Cosac
- Apxs
- Sesame
- Mupus

Transfers Experiment memory to Mass Memory
Transfers Lander to Orbiter
Transfers Orbiter to Earth

ROSETTA-PHILAE

Spacecraft Rosetta



Lander Philae

Instruments:
- Civa
- Sd2
- Consert
- Romap
- Ptolémé
- Cosac
- Apxs
- Sesame
- Mupus

Transfers Experiment memory to Mass Memory
Transfers Lander to Orbiter
Transfers Orbiter to Earth

Constraints:
Energy
Limited memories
Limited transfers to Orbiter

Outline

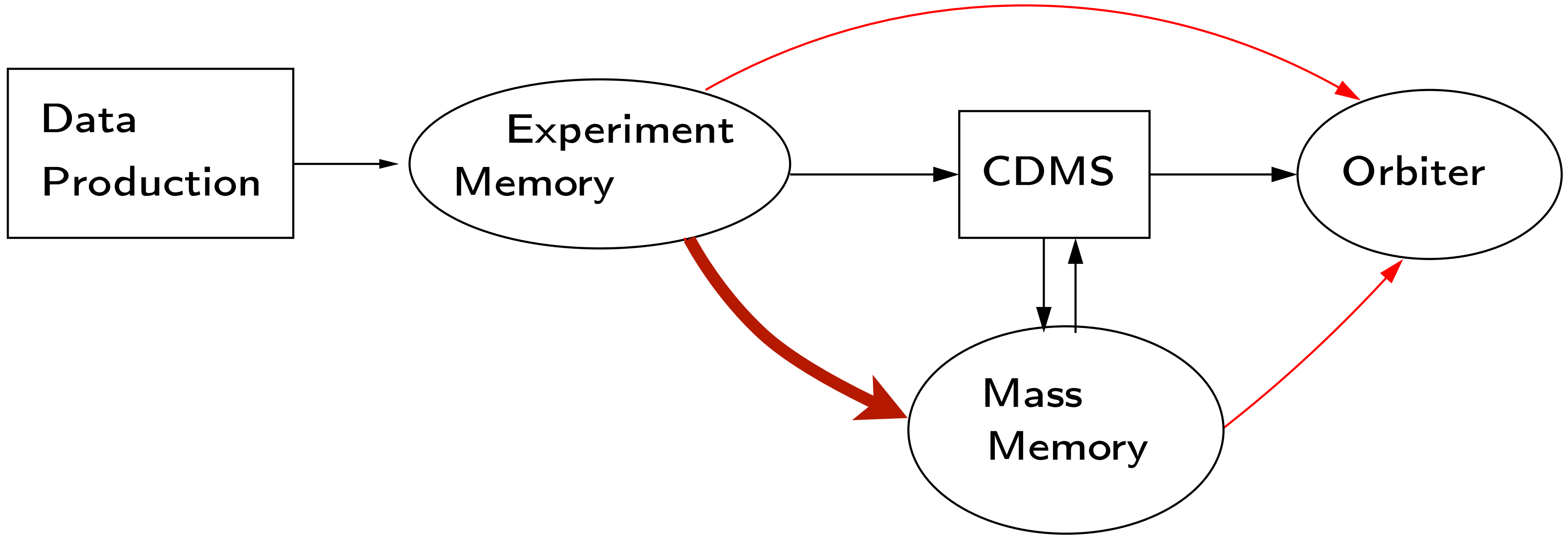
1 Problem
- Data Transfer

2 Mathematical modeling of transfers
- A Simplified model of Data Transfer

3 Algorithms
- Sweep Algorithm
- Filtering algorithms

4 Benchmarks

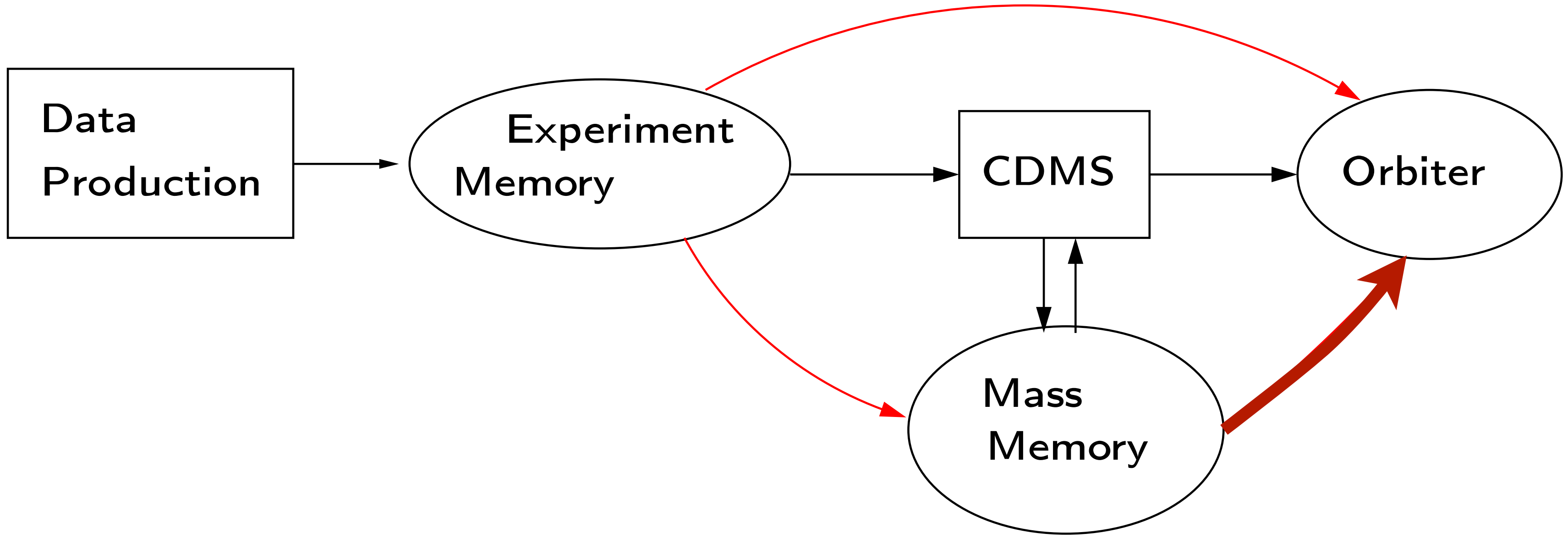
Data transfer problem



Data flow

- Experiments produce data and store it into their own memories
- The CDMS manages data transfers (following a fixed policy)
  - Transfer data from an experiment to the mass memory

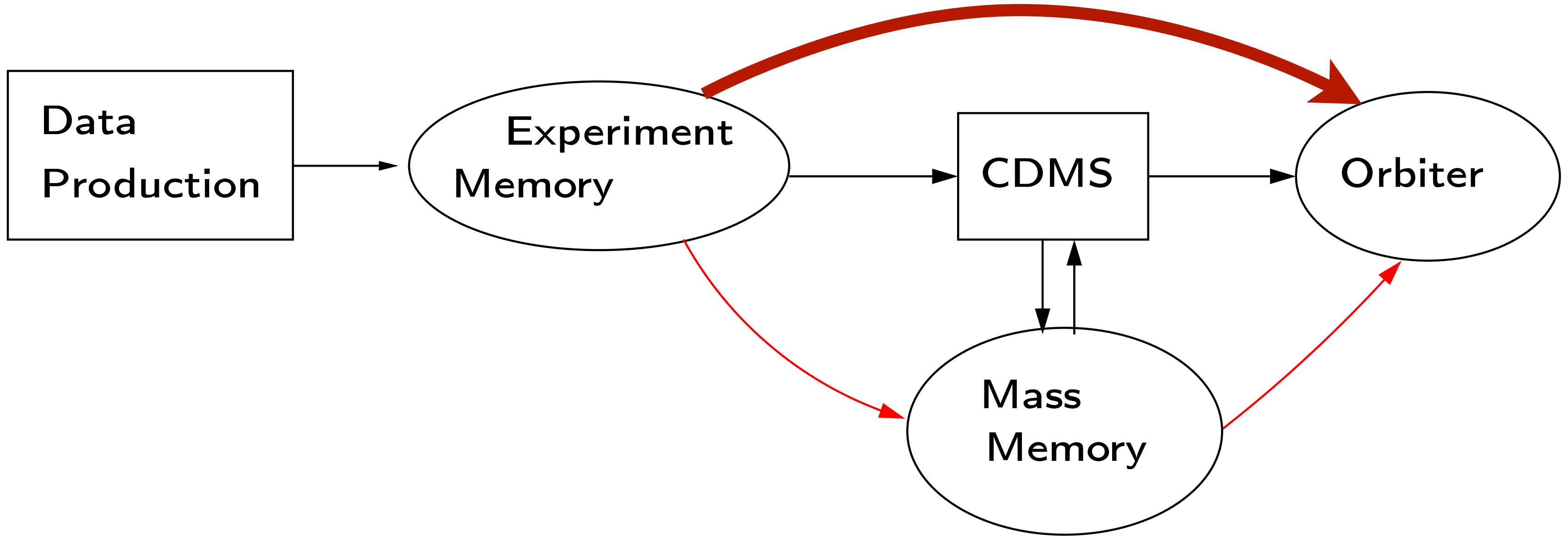
Data transfer problem



Data flow

- Experiments produce data and store it into their own memories
- The CDMS manages data transfers (following a fixed policy)
  - Transfer data from an experiment to the mass memory
  - Transfer data from the mass memory to the orbiter

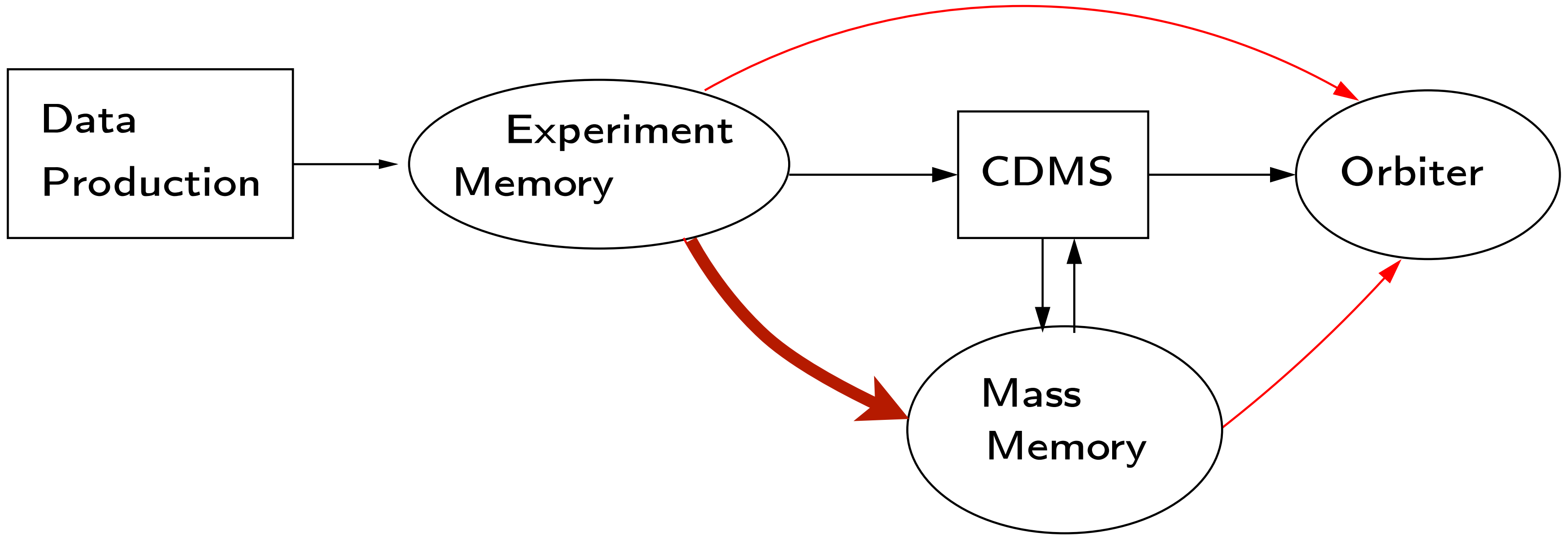
Data transfer problem



Data flow

- Experiments produce data and store it into their own memories
- The CDMS manages data transfers (following a fixed policy)
  - Transfer data from an experiment to the mass memory
  - Transfer data from the mass memory to the orbiter
  - Transfer data from an experiment directly to the orbiter

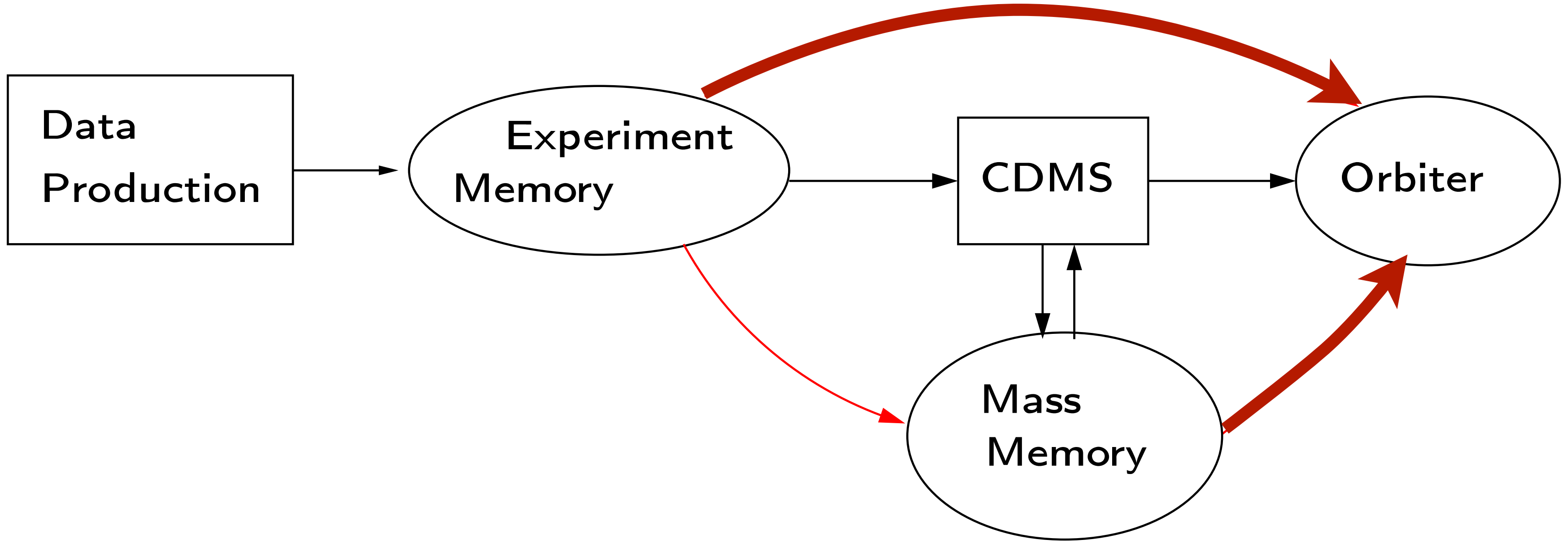
CDMS - Command and Data Management System



Transfers to the mass memory

- Repeat *as long as the mass-memory is not full*:
  - Scan experiments by decreasing priority
  - Stop at the first experiment holding some data
  - Transfer one block of data from this experiment to the mass memory

CDMS - Command and Data Management System



Transfers to the orbiter

- When Rosetta is *visible* (over the horizon):
  - If the mass memory is empty: do as above, but directly to the orbiter
  - Otherwise, do as above *and* transfer data from the mass memory to the orbiter

Data loss

- The CDMS policy may lead to *data loss*
  - When an experiment of low priority produces too much data too quickly
  - When the mass memory is full and the orbiter is not visible

Data loss

- The CDMS policy may lead to *data loss*
  - When an experiment of low priority produces too much data too quickly
  - When the mass memory is full and the orbiter is not visible

How to avoid data loss

- Other than assigning priorities, we have no control on data transfer
- Priorities are fixed for long periods

Data loss

- The CDMS policy may lead to *data loss*
  - When an experiment of low priority produces too much data too quickly
  - When the mass memory is full and the orbiter is not visible

How to avoid data loss

- Other than assigning priorities, we have no control on data transfer
- Priorities are fixed for long periods
- One needs to schedule data producing tasks so that those circumstances will not occur

Data loss

- The CDMS policy may lead to *data loss*
  - When an experiment of low priority produces too much data too quickly
  - When the mass memory is full and the orbiter is not visible

How to avoid data loss

- Other than assigning priorities, we have no control on data transfer
- Priorities are fixed for long periods
- One needs to schedule data producing tasks so that those circumstances will not occur
  - Global constraint on data producing tasks

Previous approach (CNES)

MOST

- Development of a model/solver tool named **MOST**
- Constraint Programming (with ILOG Scheduler/Solver)

Model for data transfer

- Modeled with reservoir resources (IloReservoir)
  - Production tasks fill the resources
  - *Predefined transfer tasks (of variable duration) empty the resources*
    - Many extra variables, poor propagation, precision loss

Outline

1 Problem

- Data Transfer

2 Mathematical modeling of transfers

- A Simplified model of Data Transfer

3 Algorithms

- Sweep Algorithm
- Filtering algorithms

4 Benchmarks

Constraint checking - Computing the transfers

Constraint checking

- We need to compute memory occupation through time
- Once data producing tasks are scheduled, it is possible to “unroll” the policy of the CDMS
    - Costly: we may need as many steps as blocks of data to transfer!

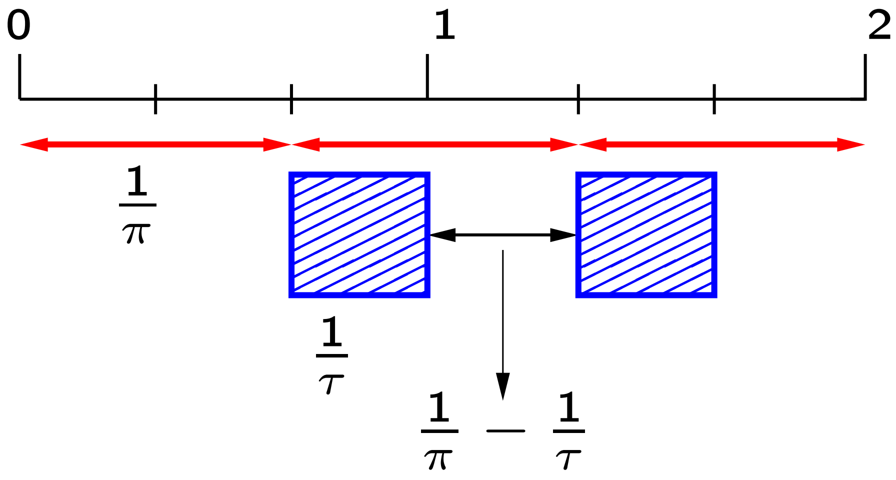
Constraint checking - Computing the transfers

Constraint checking

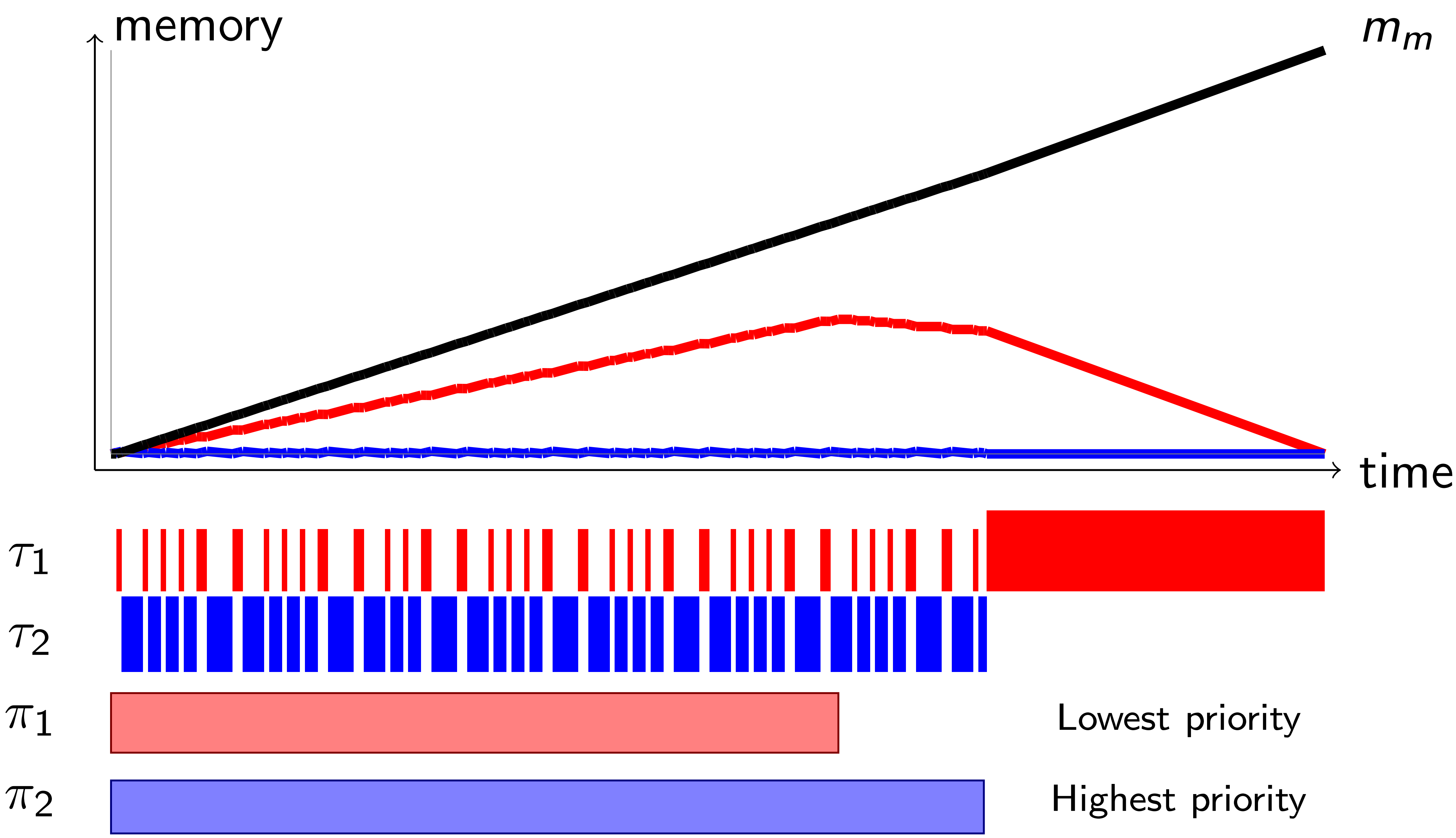
- We need to compute memory occupation through time
- Once data producing tasks are scheduled, it is possible to “unroll” the policy of the CDMS
  - Costly: we may need as many steps as blocks of data to transfer!
  - Let π be the production rate and τ the transfer rate
  - If $\pi < \tau$ there are gaps that can be used for other experiments

Production $\pi = 1,5\ blk/s$

Transfer $\tau = 3\ blk/s$



Example with two experiments

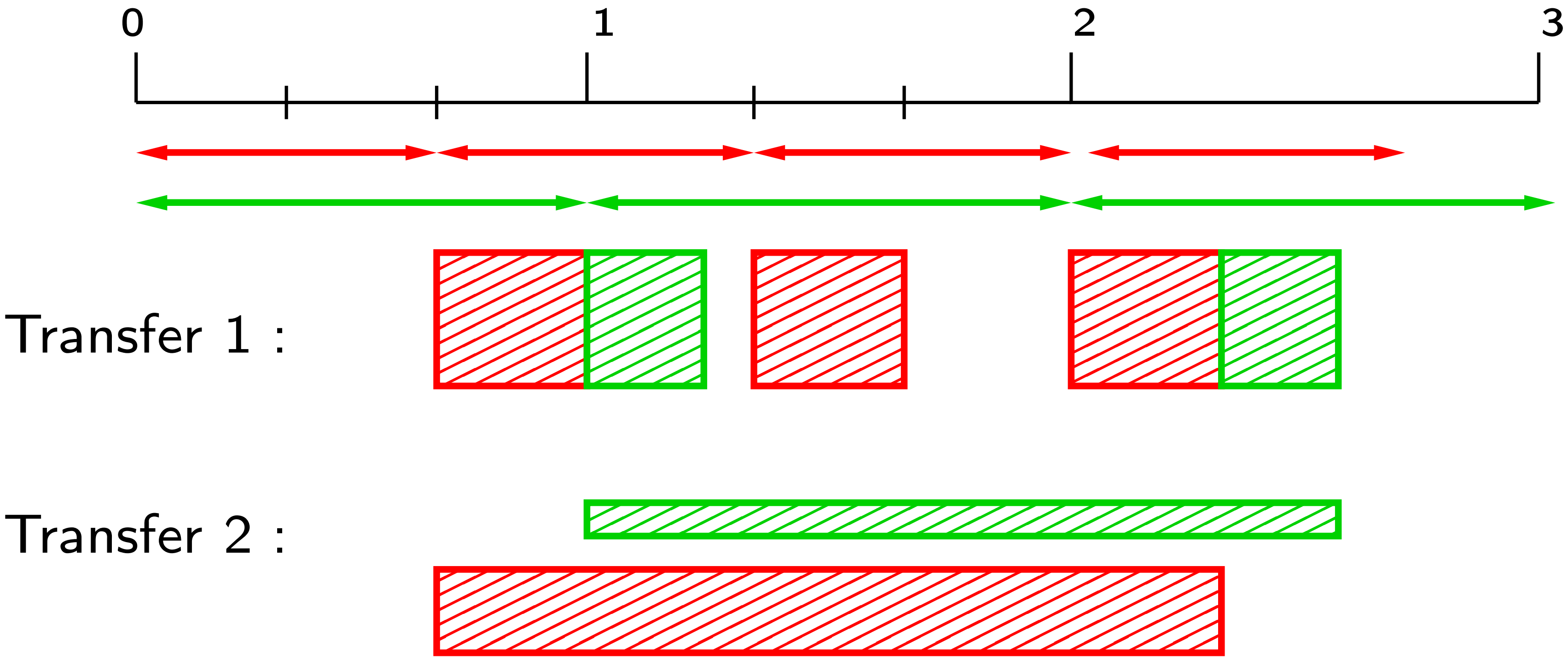


Modification

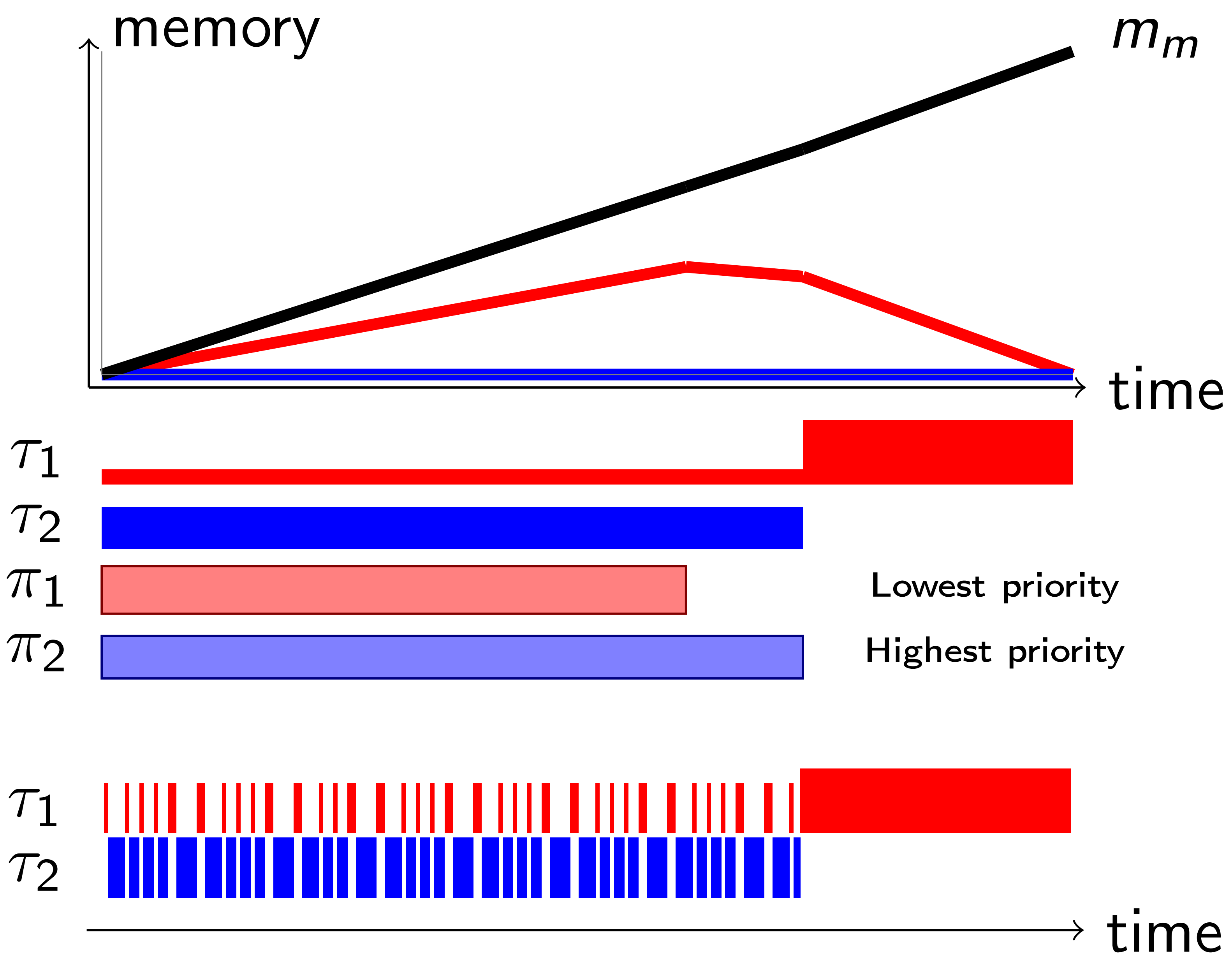
- Approximate switches between experiments by the partition of a *bandwidth*
- The bandwidth is allocated proportionally to the usage of the bus

Modification

- Approximate switches between experiments by the partition of a *bandwidth*
- The bandwidth is allocated proportionally to the usage of the bus
  - The area (transfer time *x* rate) is equivalent
  - We create only one transfer



Same example, using the “bandwidth” view



Outline

1 Problem
- Data Transfer

2 Mathematical modeling of transfers
- A Simplified model of Data Transfer

3 Algorithms
- Sweep Algorithm
- Filtering algorithms

4 Benchmarks

For the transfers

- A sweep algorithm to check the constraint
  - Given a schedule of production, the transfers are computed in $O(n \log n)$

Principle of the sweep algorithm for computing transfer

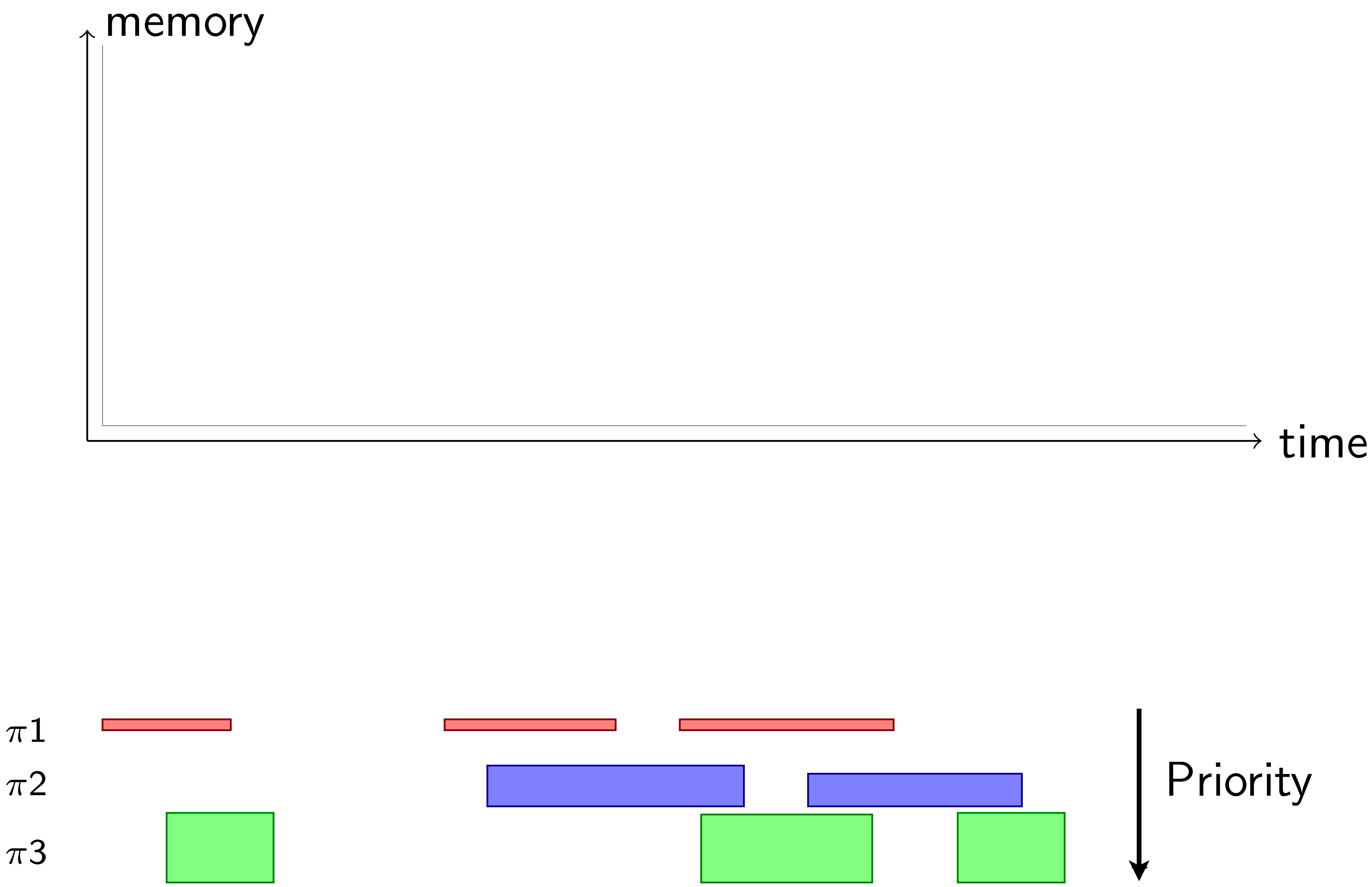
- We consider a list of events in a chronological order
    - start/end of an experiment
    - start/end of data production
    - start/end of visibility

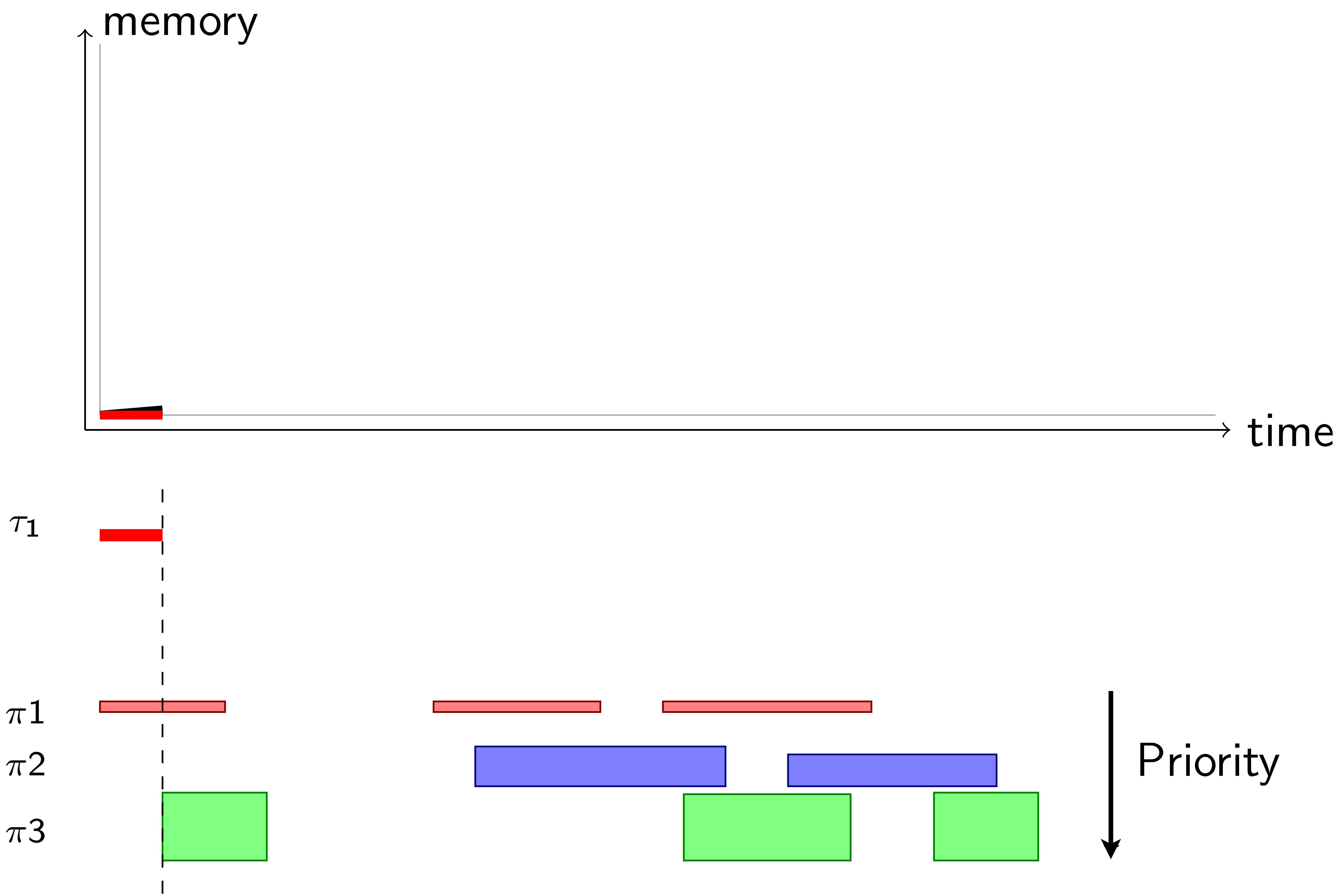
Principle of the sweep algorithm for computing transfer

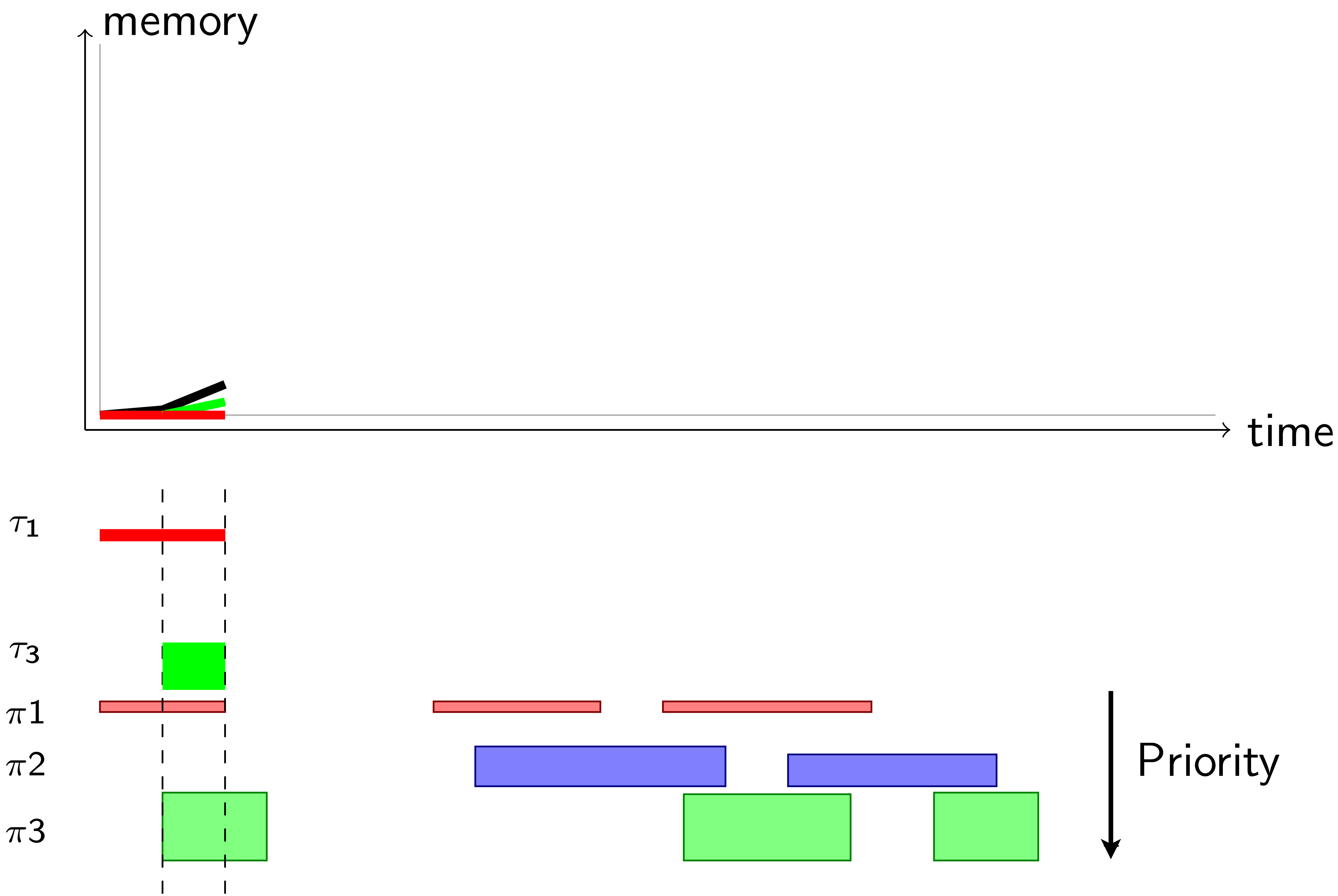
- We consider a list of events in a chronological order
    - start/end of an experiment
    - start/end of data production
    - start/end of visibility
- At each step, we apply the policy rules of the CDMS (priority) in order to determine the current transfers
    - Nothing will change until the next event
    - If the current transfer empty a memory until the next event, then we create the event "Memory emptied"
    - If the mass memory is full before the next event, we create the event "Full mass memory"

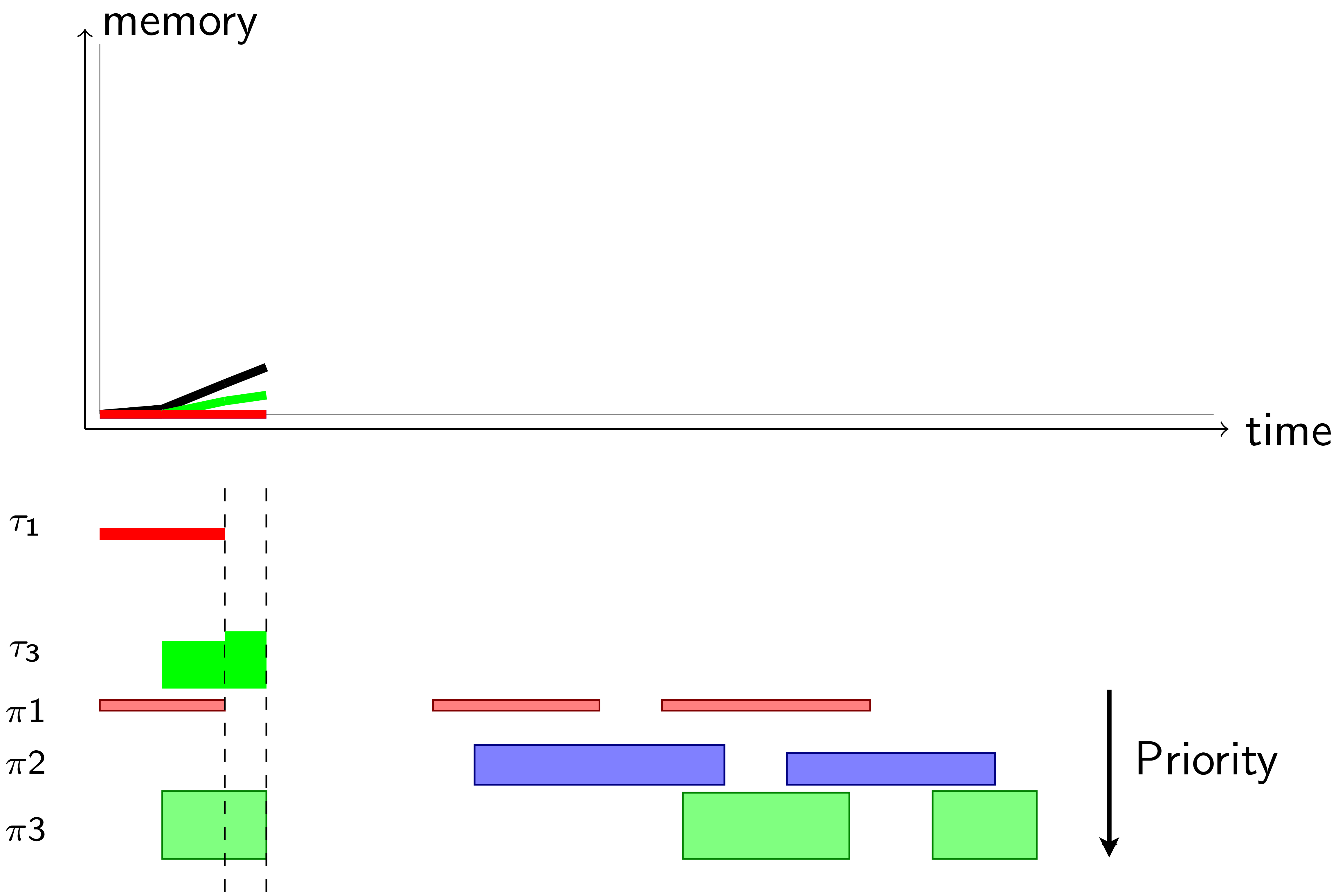
Principle of the sweep algorithm for computing transfer

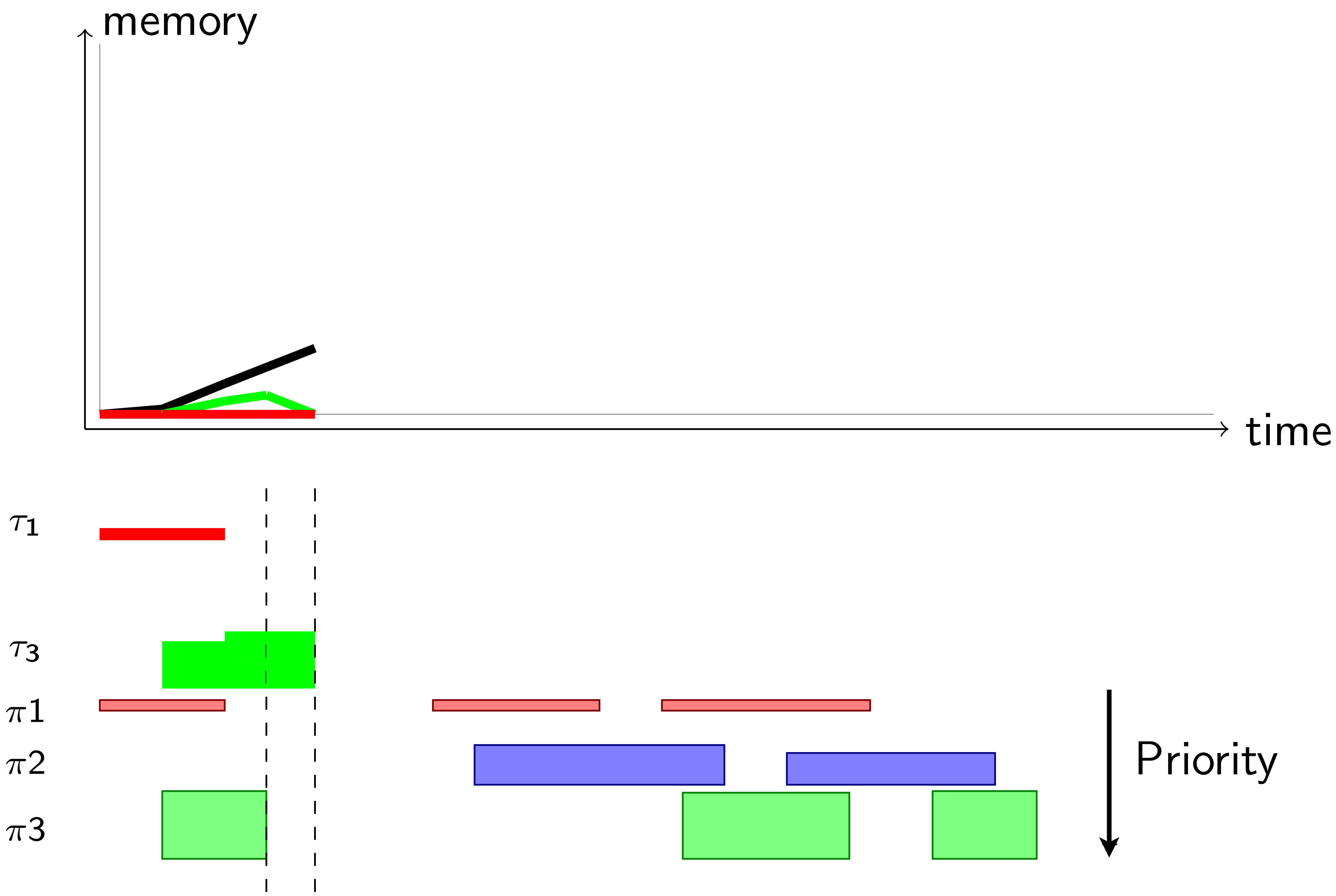
- We consider a list of events in a chronological order
    - start/end of an experiment
    - start/end of data production
    - start/end of visibility
- At each step, we apply the policy rules of the CDMS (priority) in order to determine the current transfers
    - Nothing will change until the next event
    - If the current transfer empty a memory until the next event, then we create the event "Memory emptied"
    - If the mass memory is full before the next event, we create the event "Full mass memory"
- If at any time the capacity of an experiment memory is exceeded, then data is lost and the constraint is violated for this schedule.

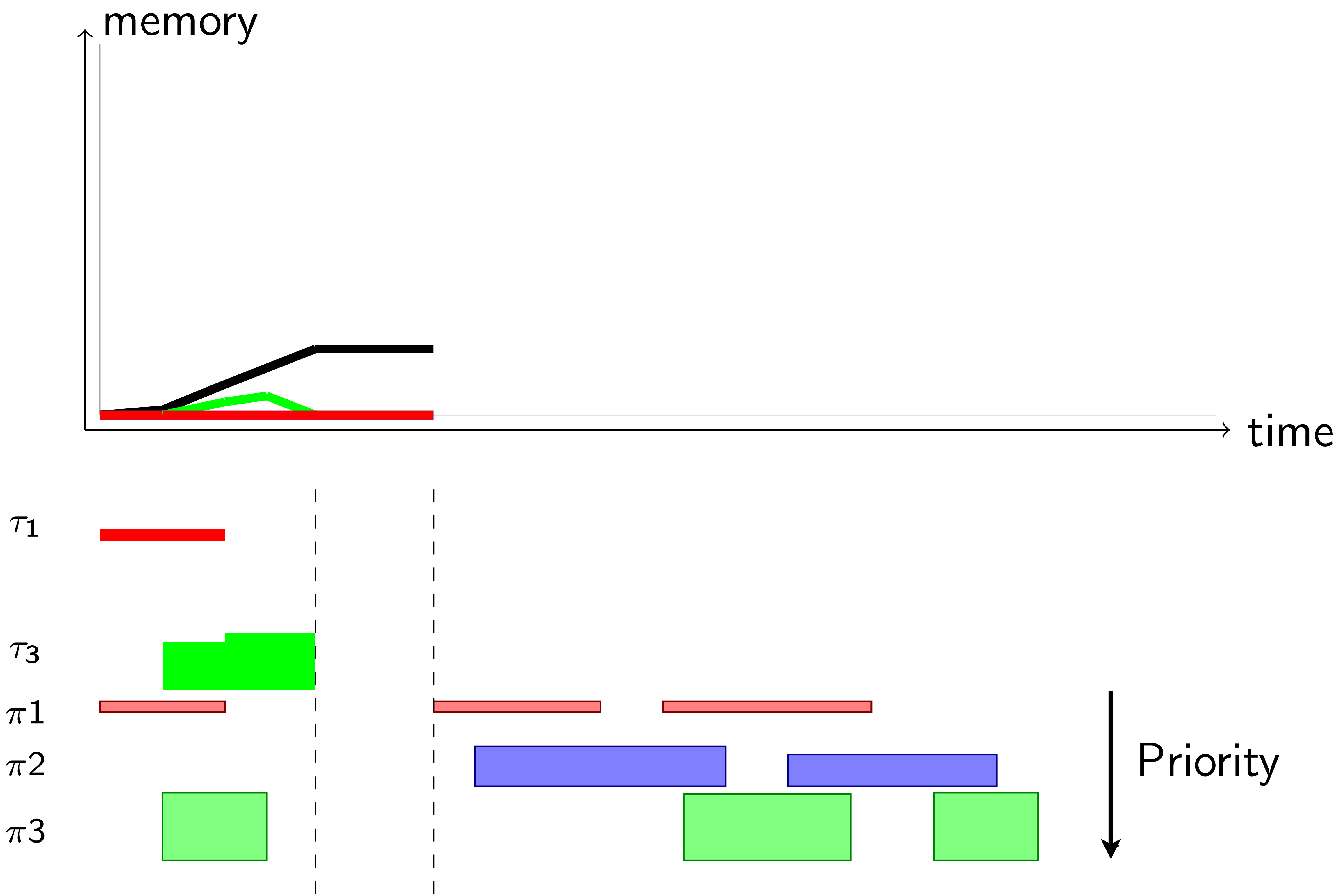

memory
time
$\pi 1$
$\pi 2$
$\pi 3$
Priority

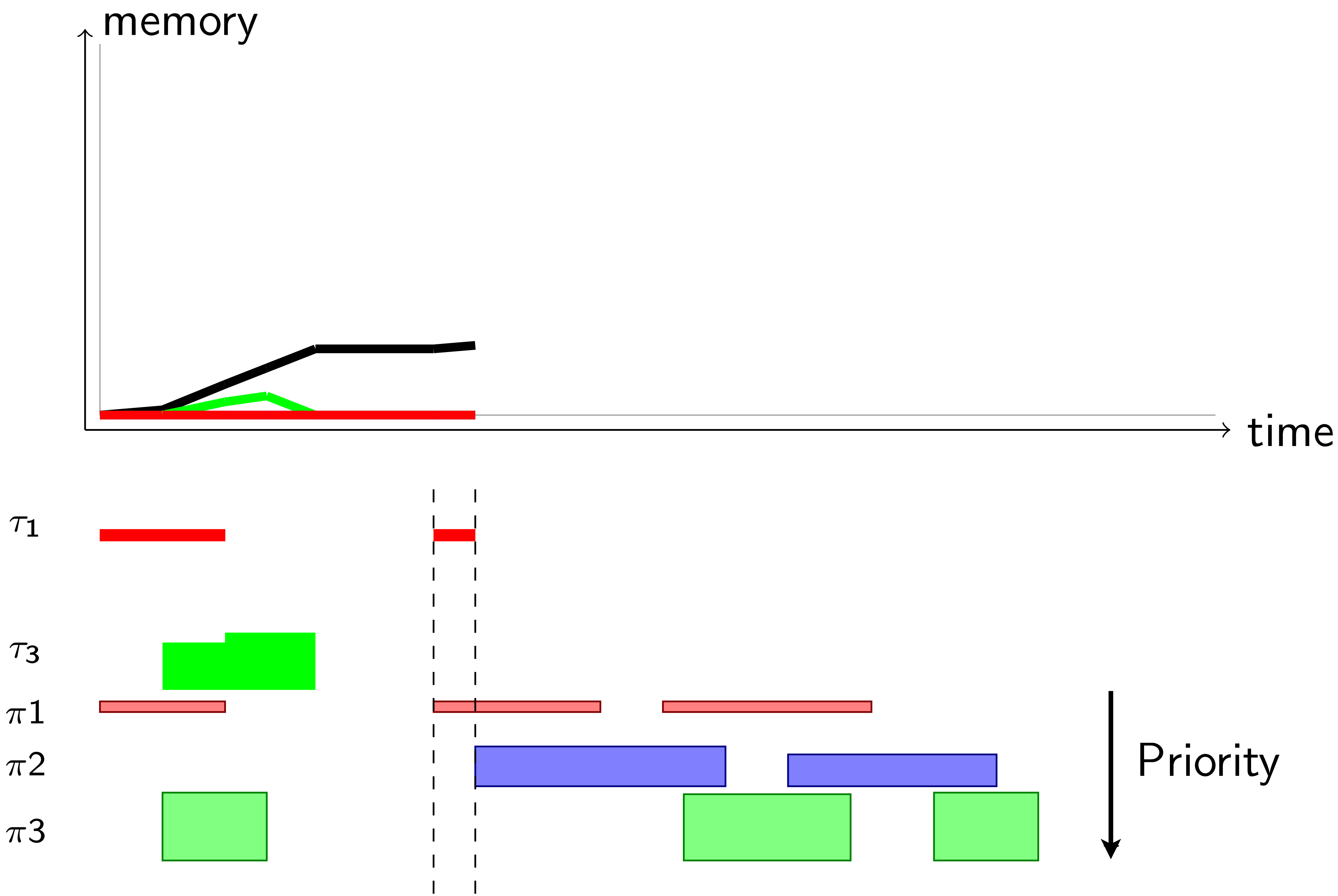

memory
time
τ_1
$\pi 1$
$\pi 2$
$\pi 3$
Priority

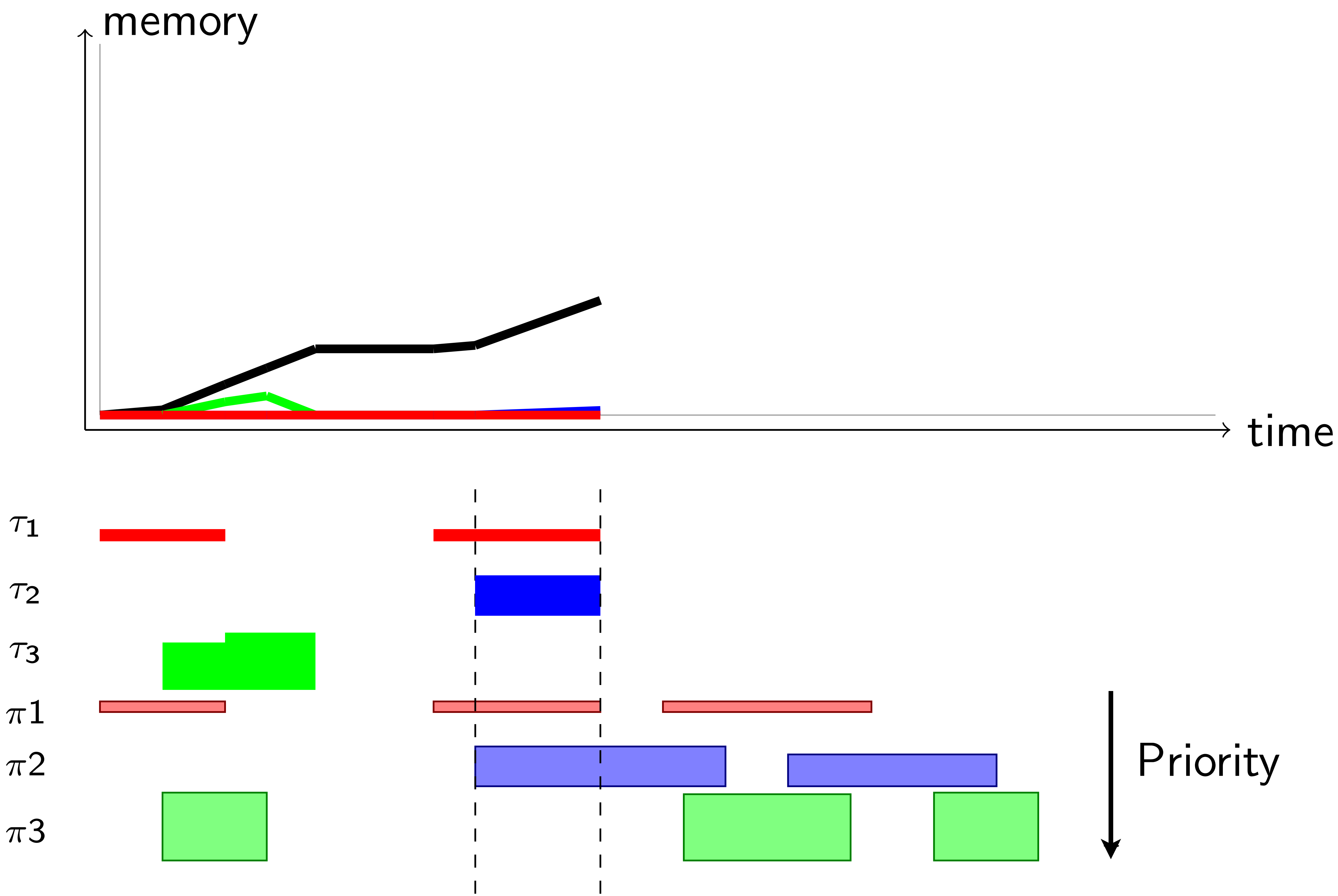

memory
time
τ_1
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

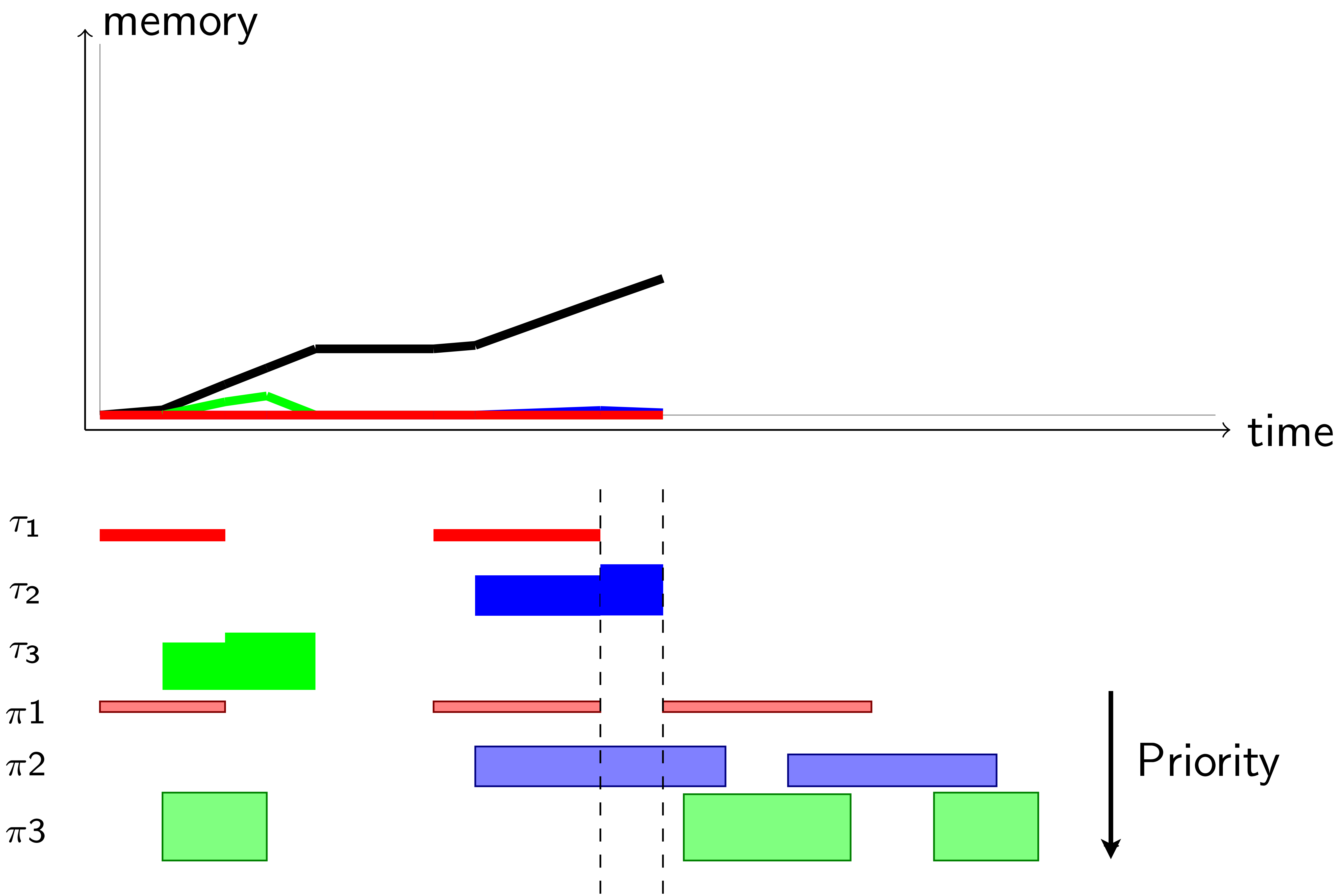

memory
time
τ_1
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

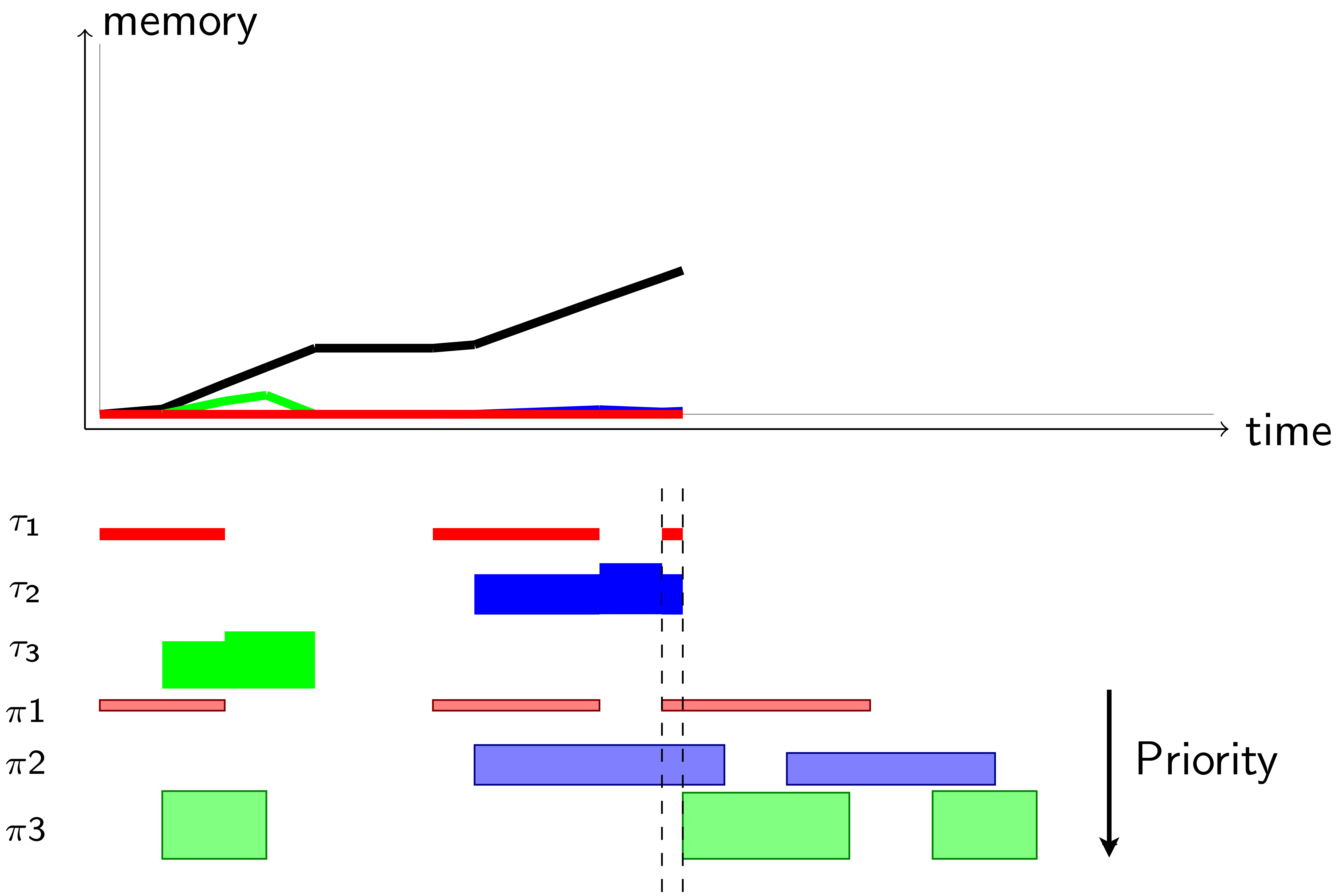

memory
time
τ_1
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

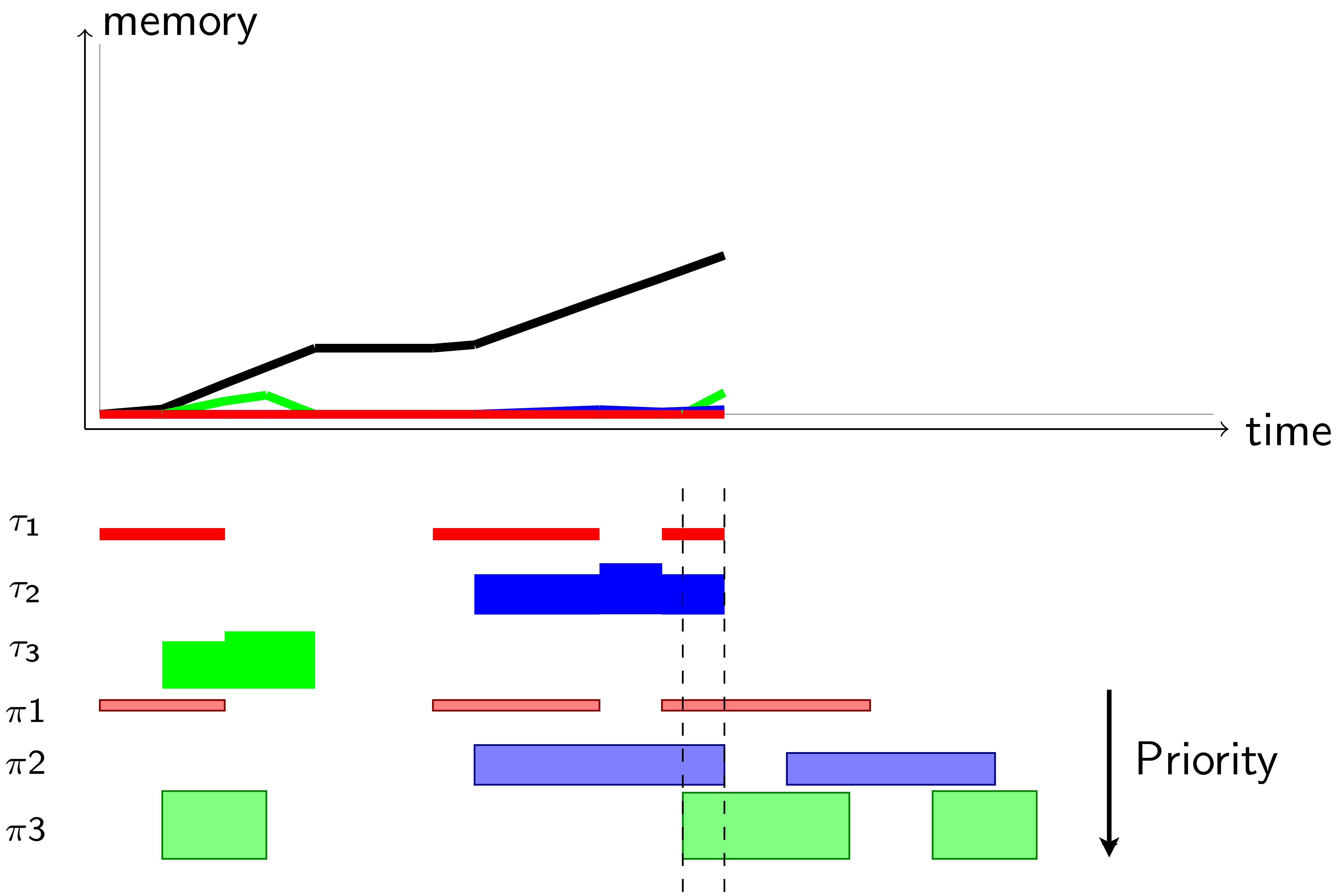

memory
time
τ_1
τ_3
$\pi1$
$\pi2$
$\pi3$
Priority

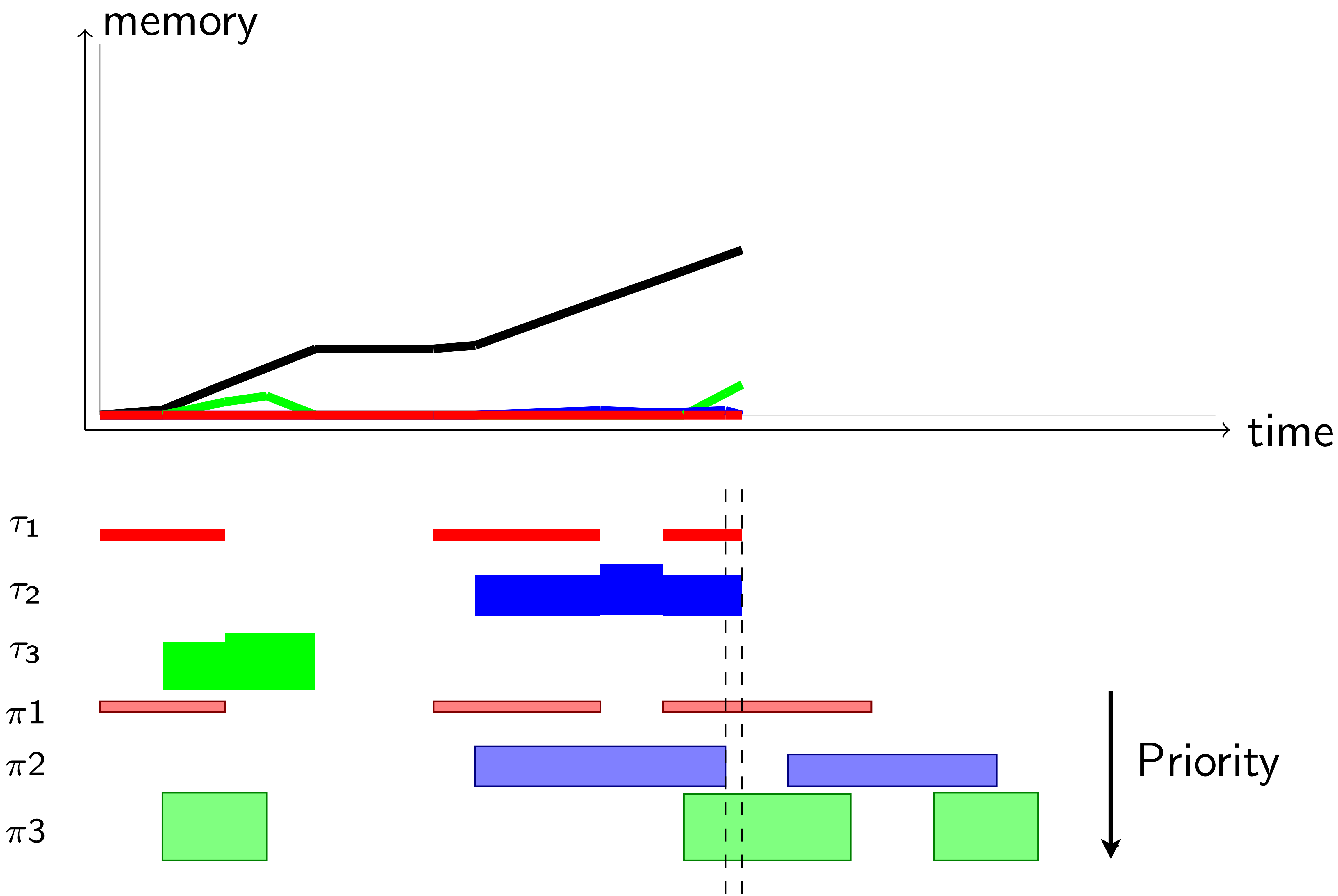

memory
time
τ_1
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

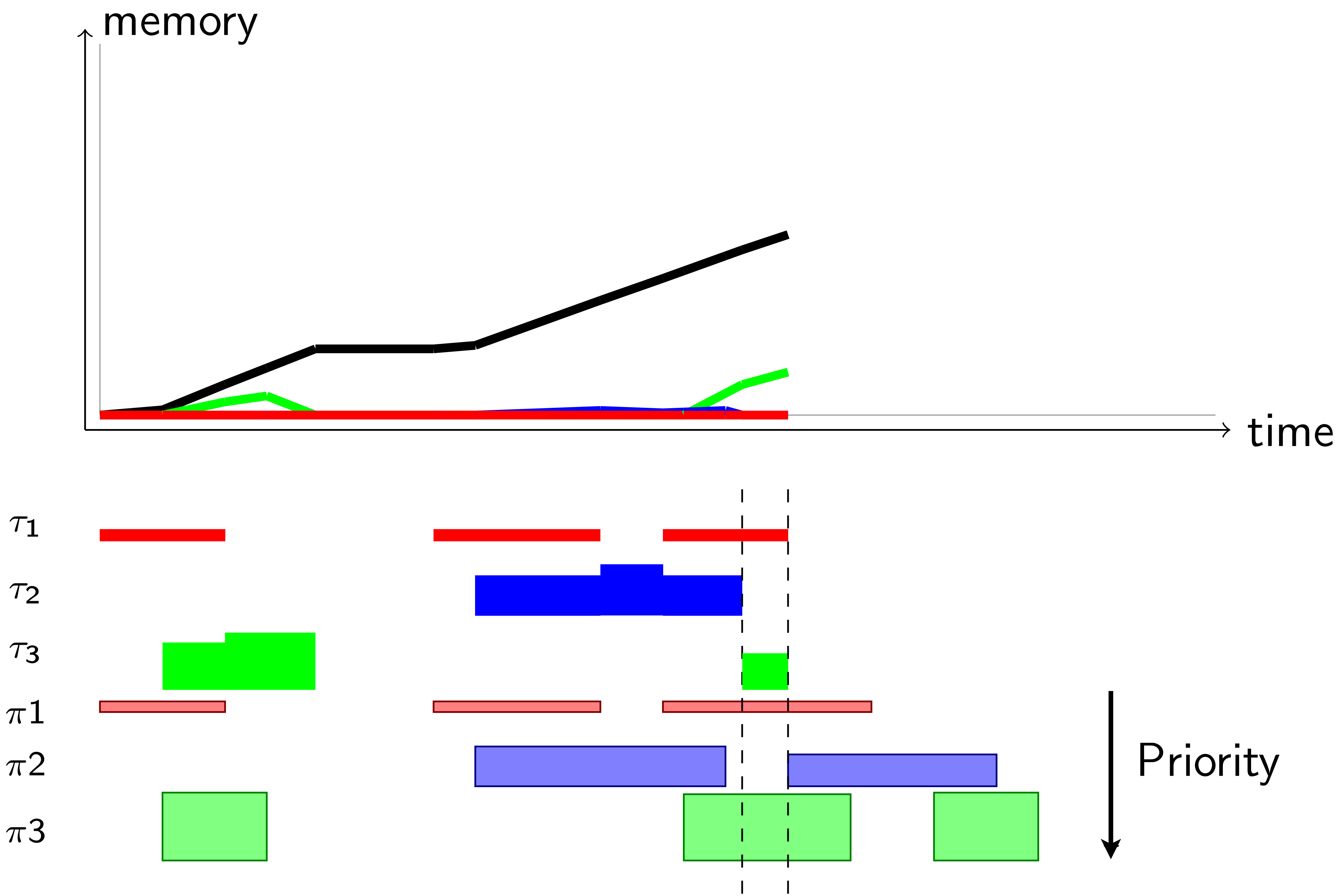

memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

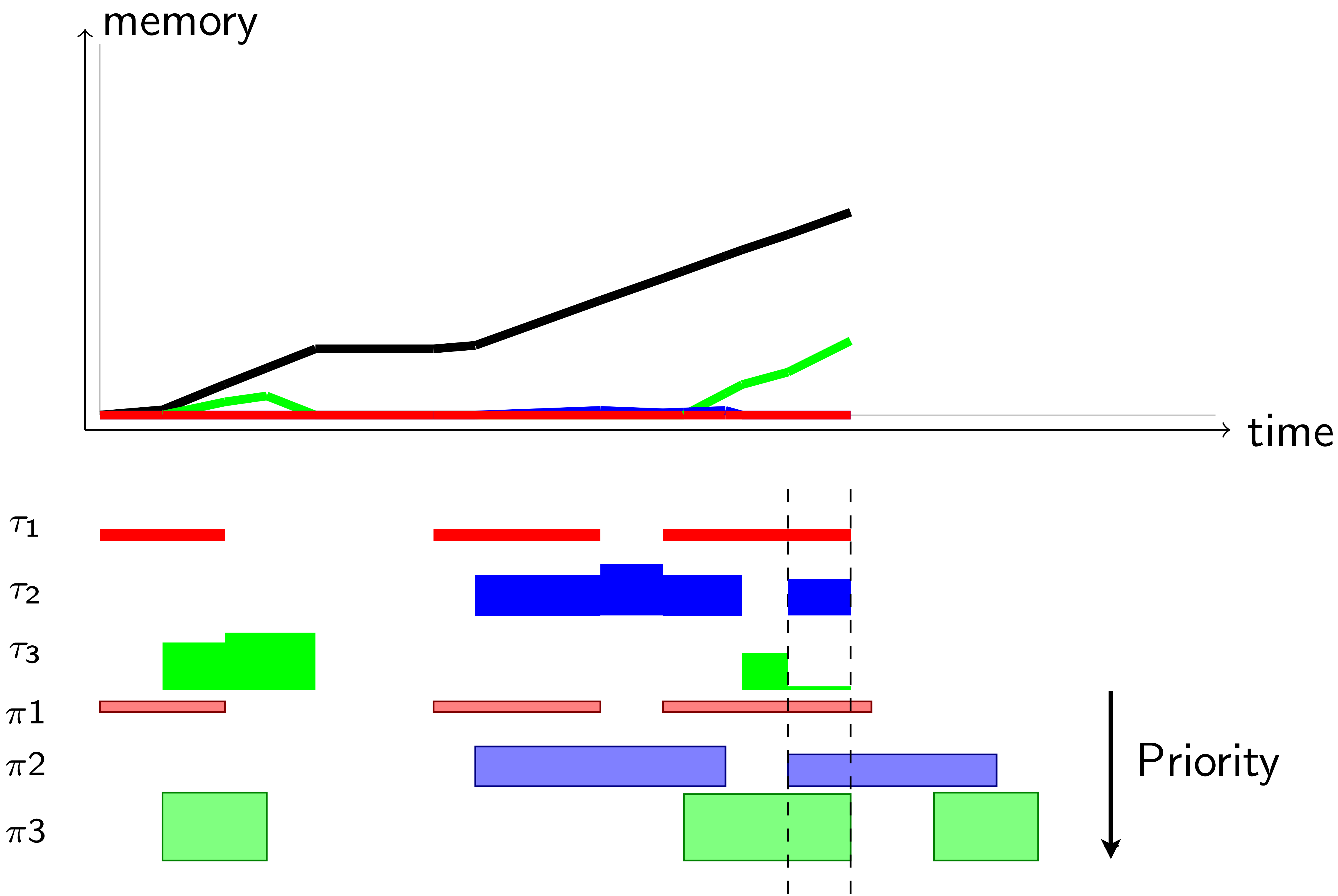

memory
time
τ_1
τ_2
τ_3
$\pi1$
$\pi2$
$\pi3$
Priority

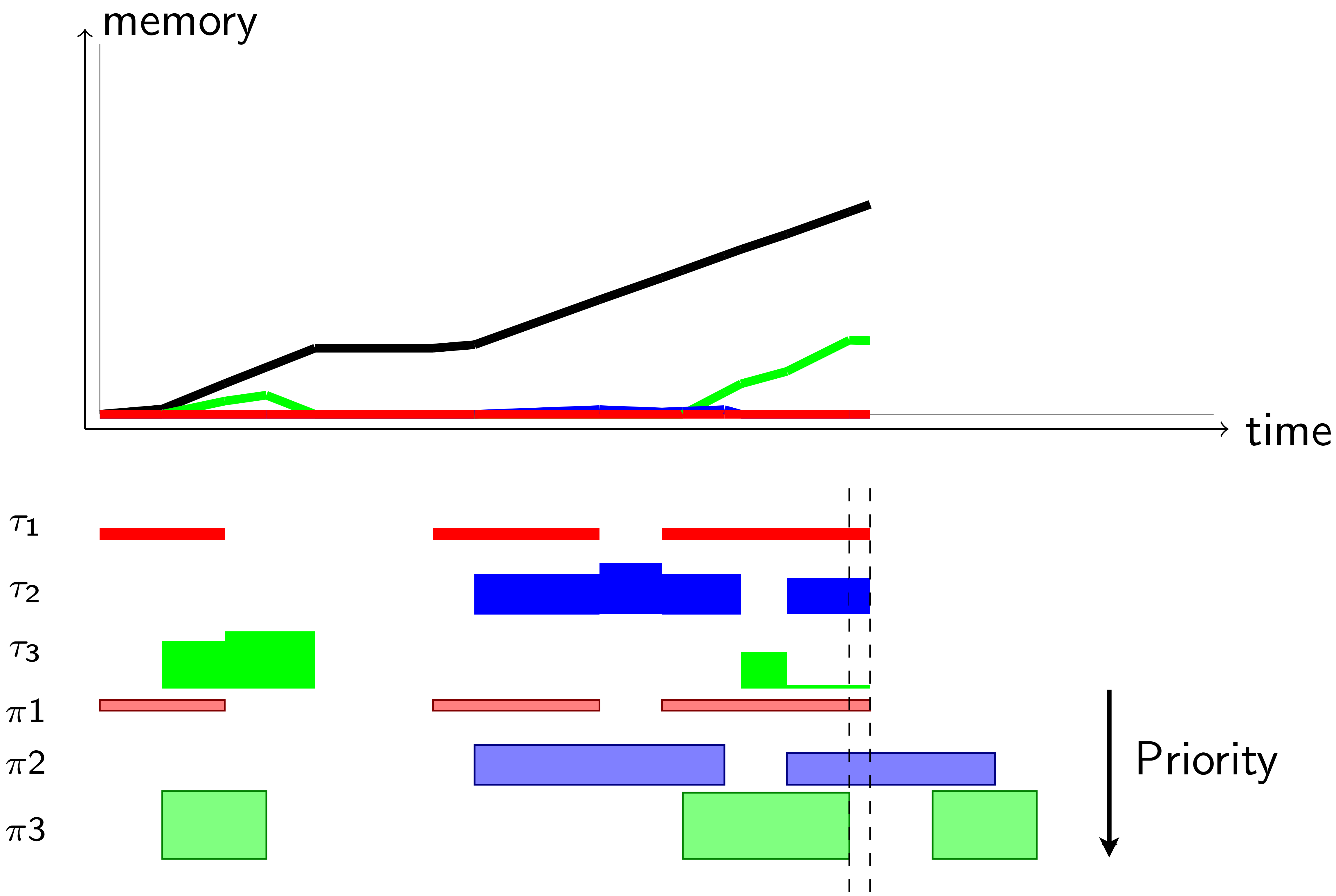

memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

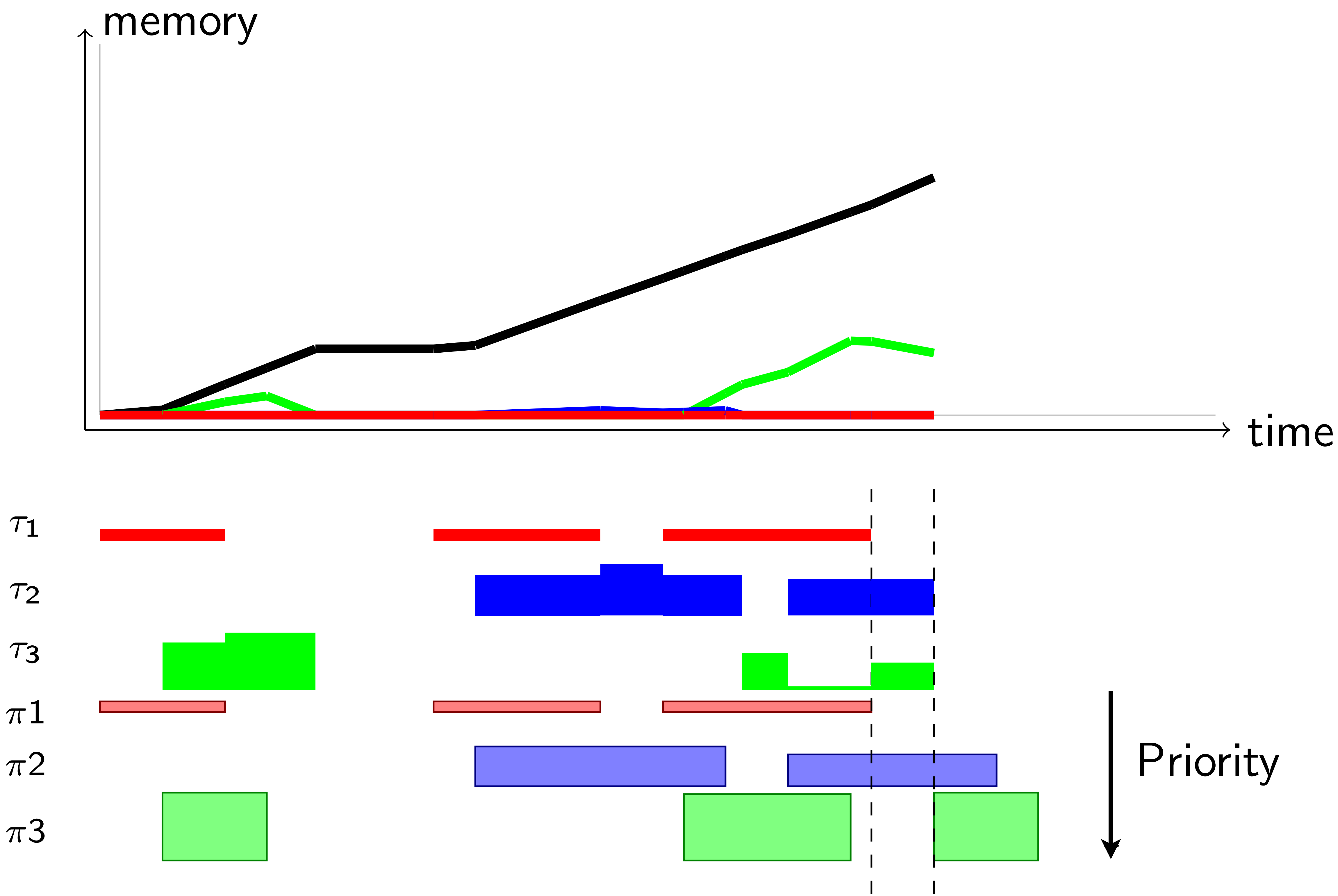

memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

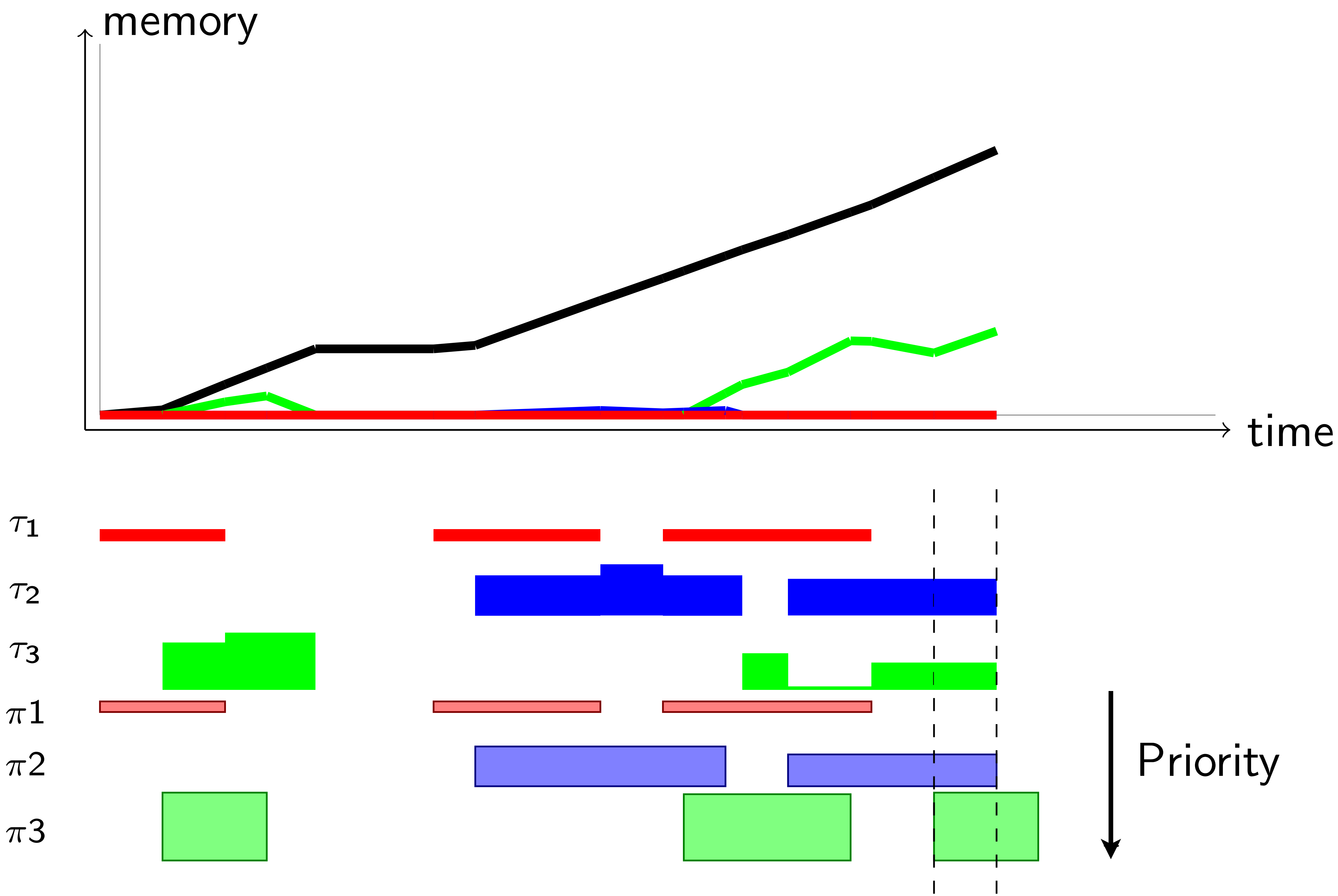

memory
time
τ_1
τ_2
τ_3
$\pi1$
$\pi2$
$\pi3$
Priority

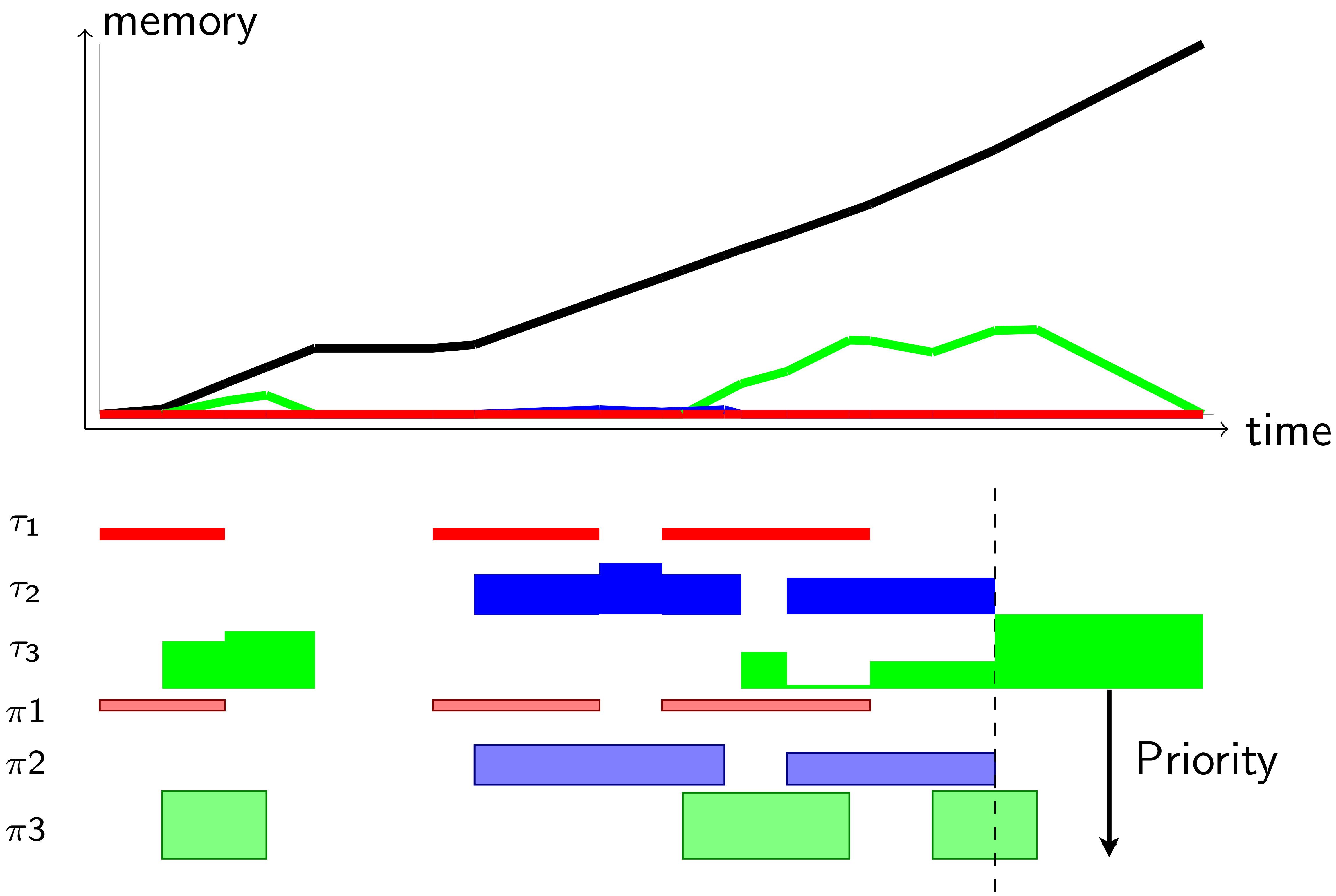

memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority


memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority


memory
time
τ_1
τ_2
τ_3
$\pi1$
$\pi2$
$\pi3$
Priority

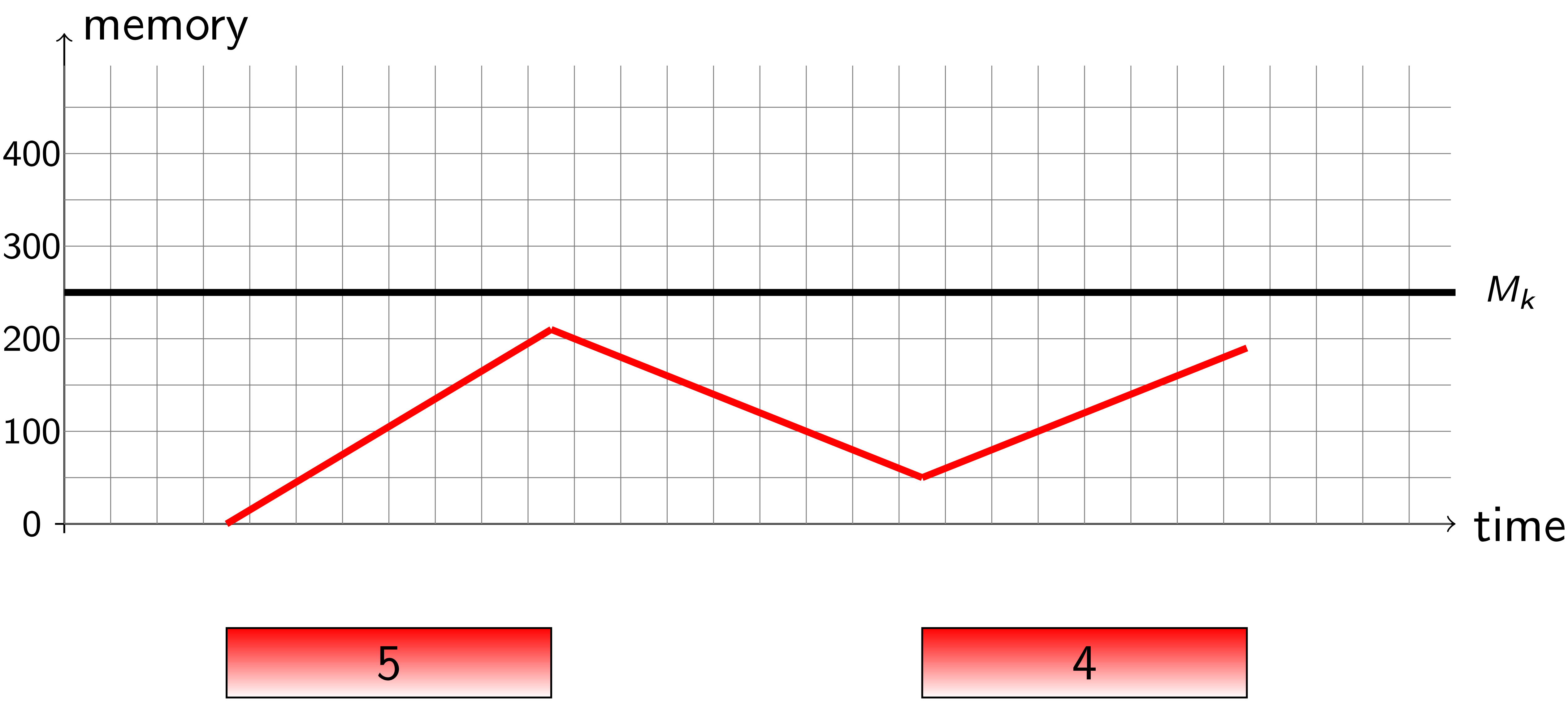

memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority


memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority


memory
time
τ_1
τ_2
τ_3
$\pi 1$
$\pi 2$
$\pi 3$
Priority

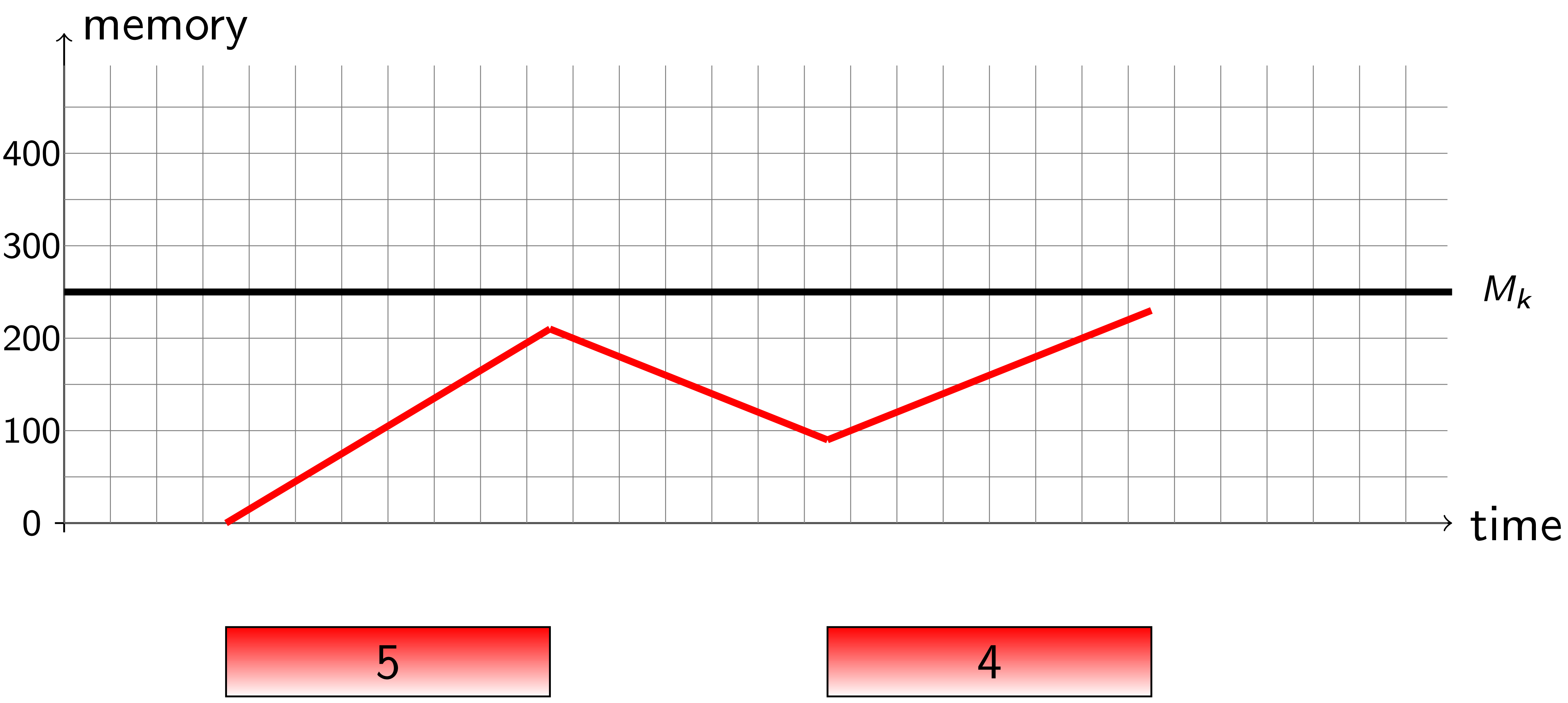
First filtering algorithm

- We study a list of production tasks Ω from an experiment
- Objective: Compute the minimum duration necessary to transfer data without saturating the experiment's memory



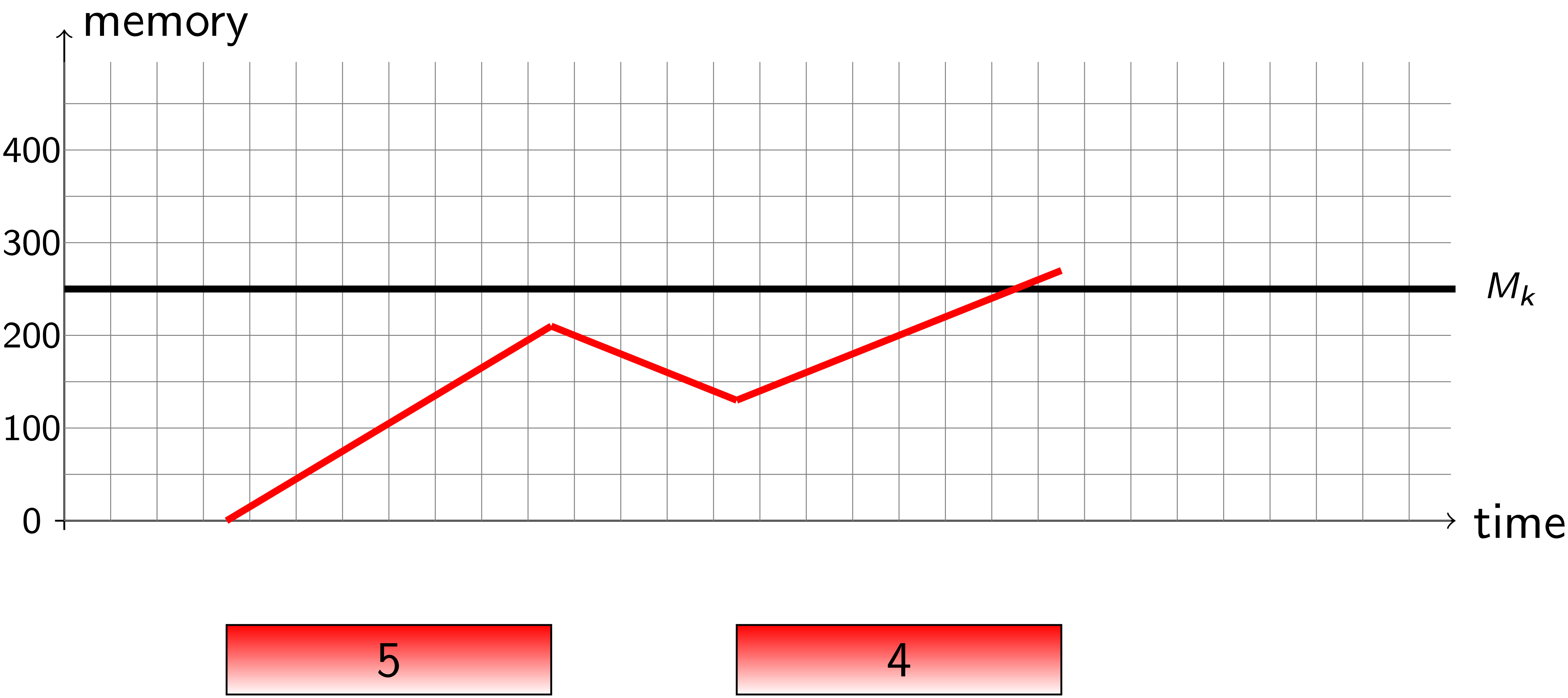
First filtering algorithm

- We study a list of production tasks Ω from an experiment
- Objective: Compute the minimum duration necessary to transfer data without saturating the experiment's memory



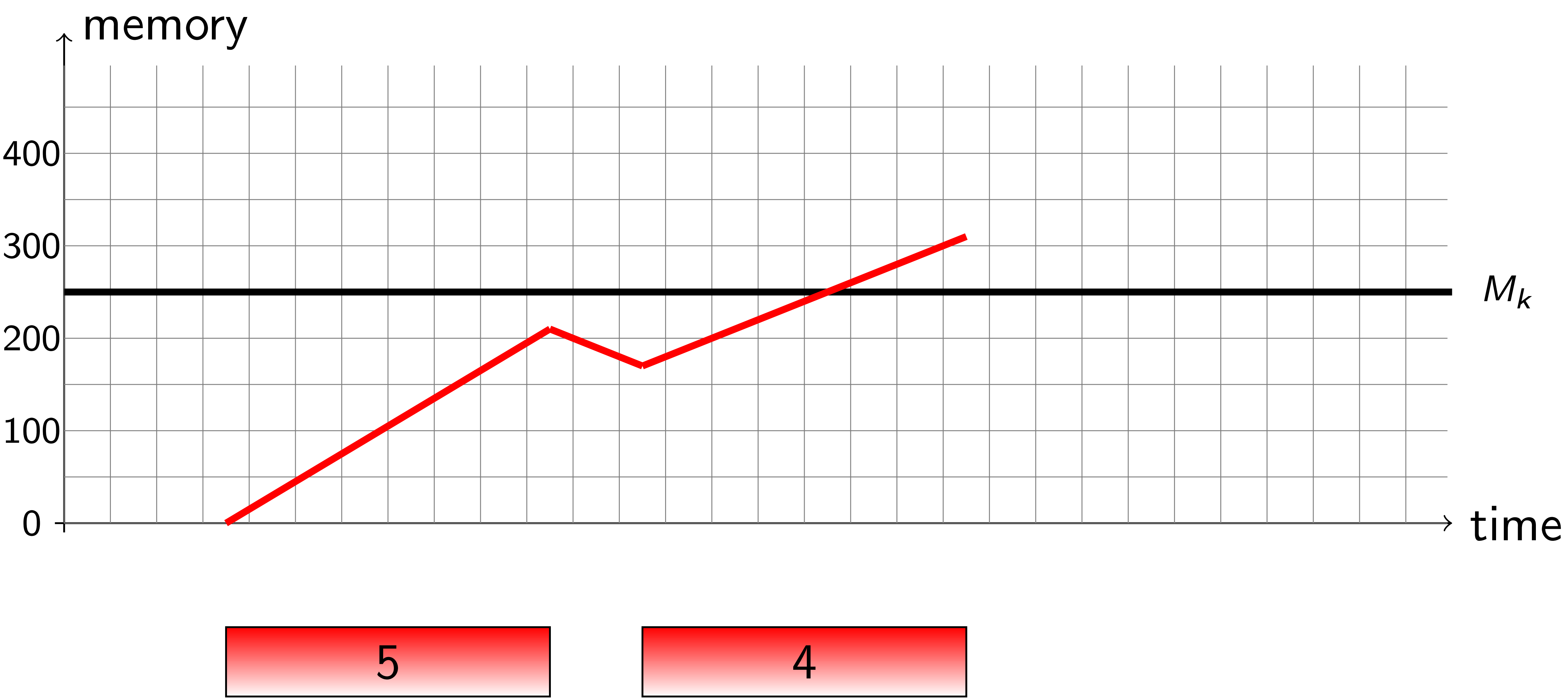
First filtering algorithm

- We study a list of production tasks Ω from an experiment
- Objective: Compute the minimum duration necessary to transfer data without saturating the experiment's memory



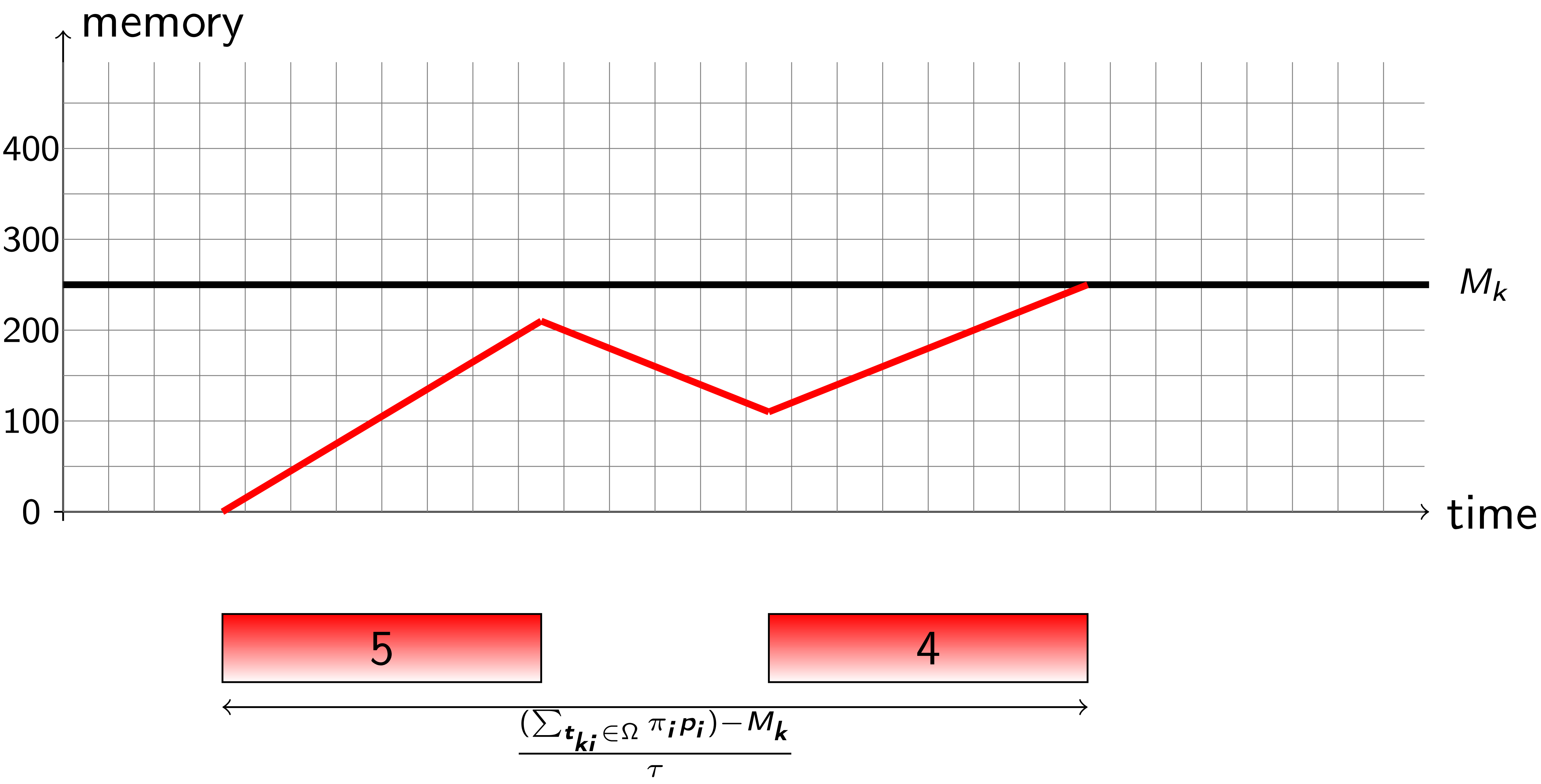
First filtering algorithm

- We study a list of production tasks Ω from an experiment
- Objective: Compute the minimum duration necessary to transfer data without saturating the experiment's memory



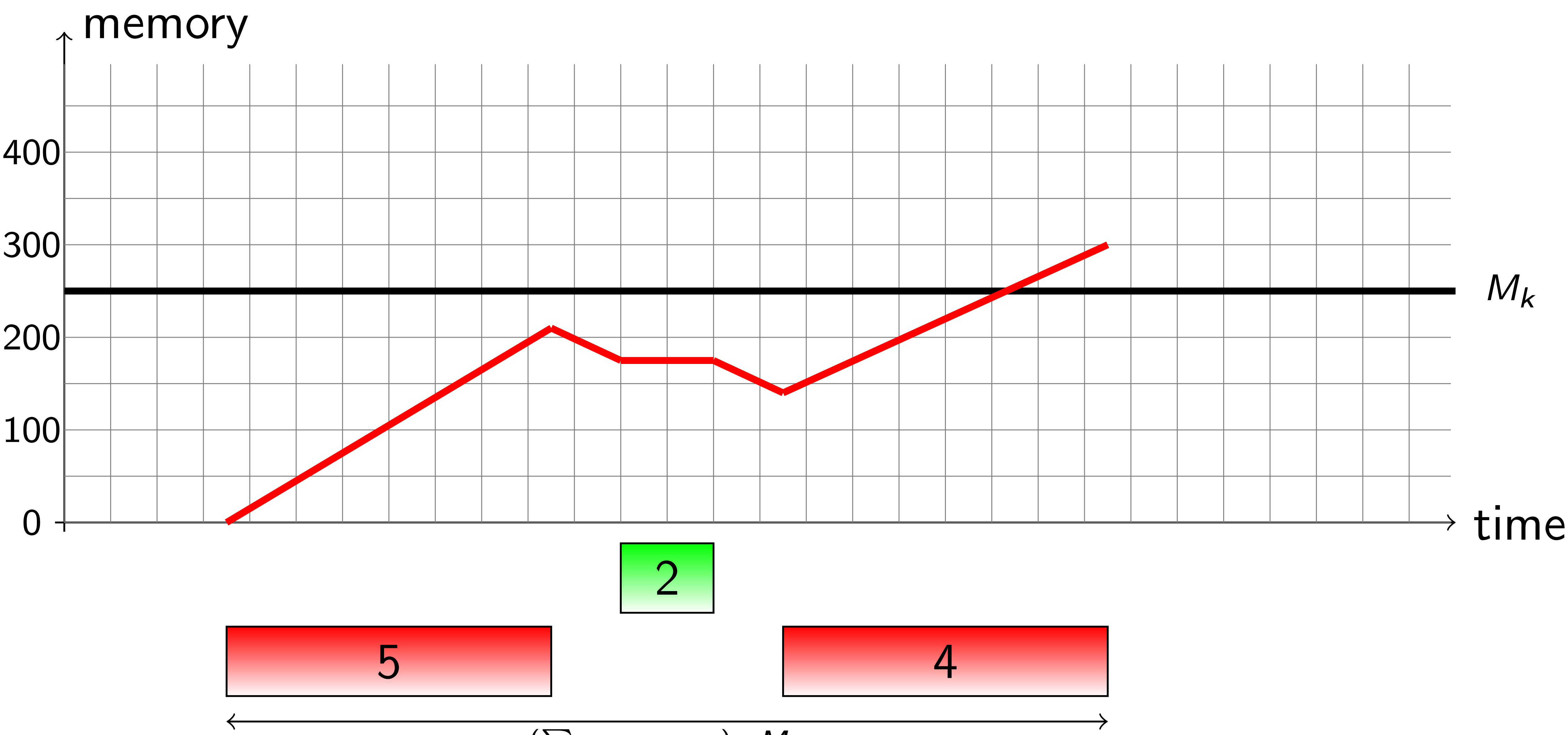
Rule

- Induce constraint: $end(t_n) - start(t_1) \geq \frac{(\sum_{t_{ki} \in \Omega} \pi_i p_i) - M_k}{\tau}$



Improvement

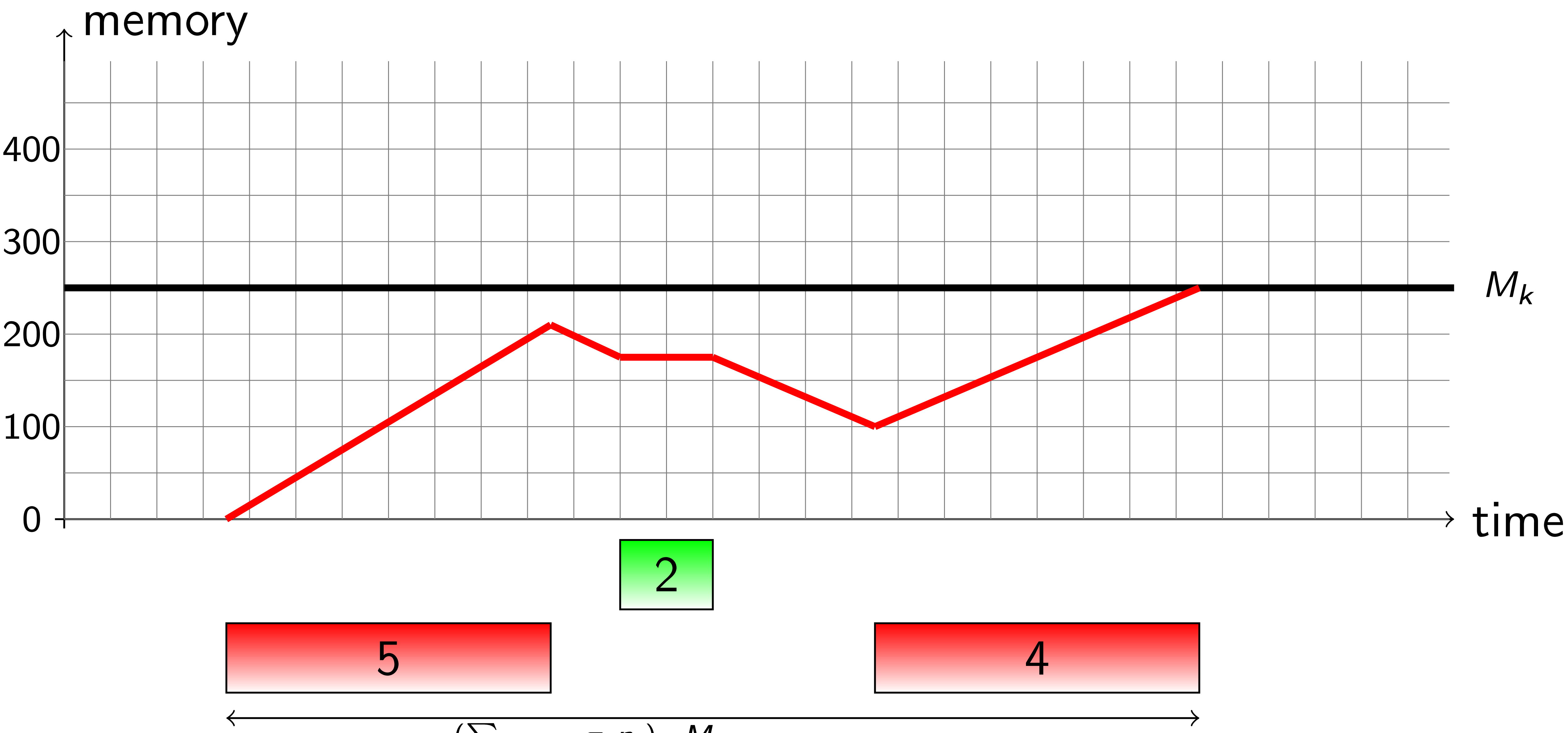
- We can improve the filtering if we take into account the other experiments with a higher priority.



$$\frac{(\sum_{t_{ki}\in\Omega}\pi_i p_i)-M_k}{\tau}$$

Improvement

- Constraint: $end(t_n) - start(t_1) \geq \frac{(\sum_{t_{ki}\in\Omega} \pi_i p_i) - M_k}{\tau} + \sum_{j=1}^{j<R(k)} T_j$

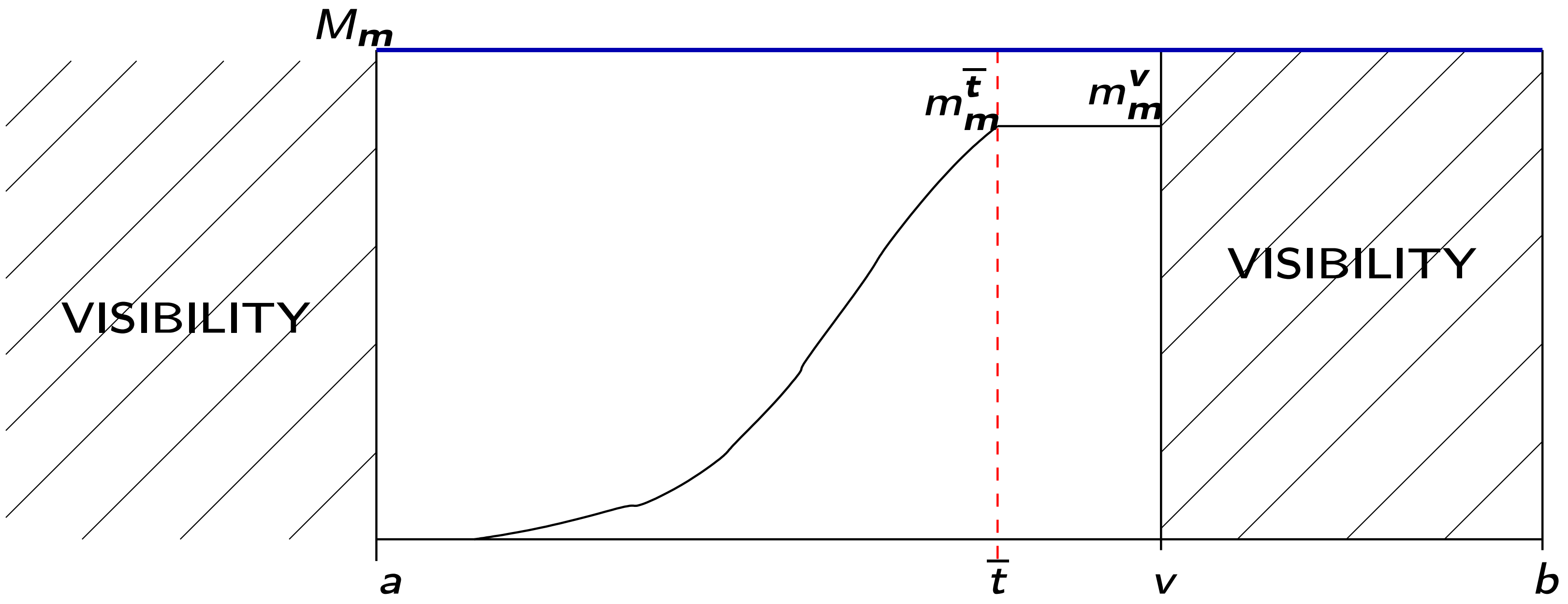


$$\frac{(\sum_{t_{ki}\in\Omega} \pi_i p_i) - M_k}{\tau} + \sum_{j=1}^{j<R(k)} T_j$$

Second filtering algorithm

Second rule

- Periods preceding visibility are critical for data loss
  - No transfer to the orbiter is possible
  - Data accumulates on the mass memory
  - At $\overline{t}$, the mass memory is saturated
- Data produced after $\overline{t}$ and before v must remain in exp memory

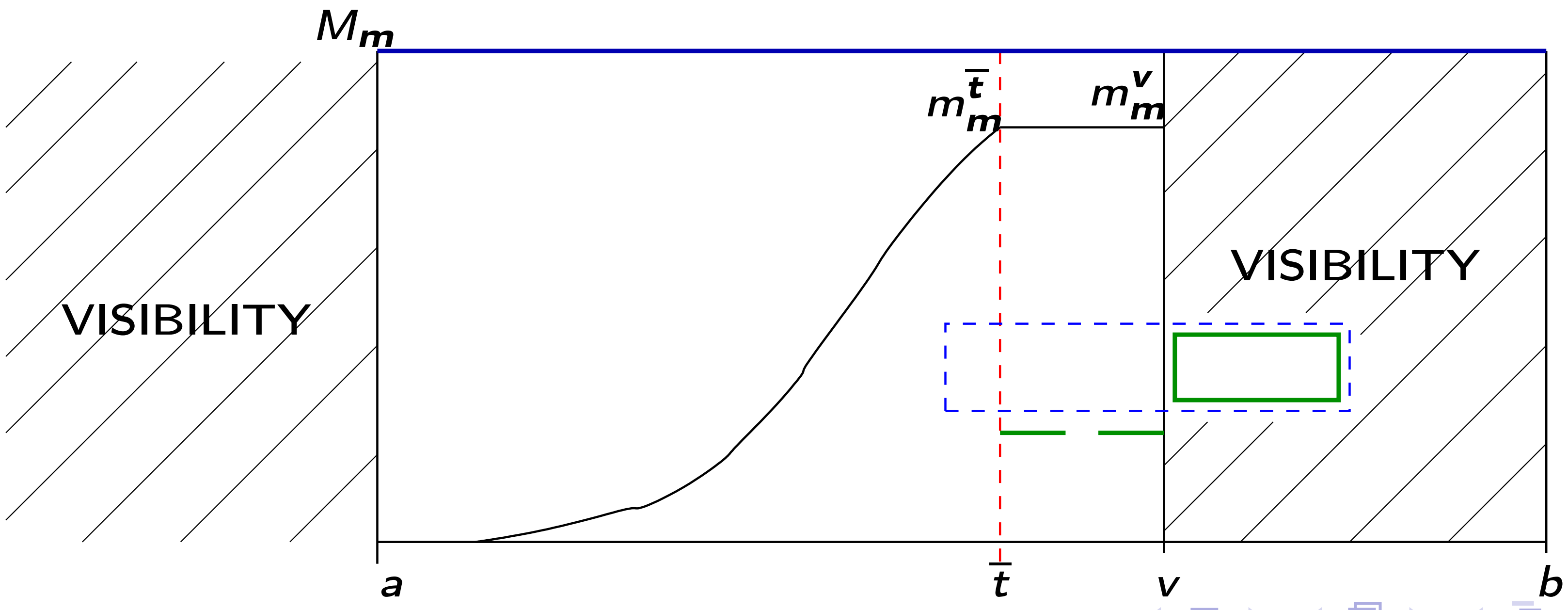


Second filtering algorithm

Second rule

- For each experiment Exp_k:
  - The data produced between $\bar{t}$ and v may exceed its capacity
    - Either it should be run earlier when it was possible to transfer
    - Either it should be run later, that is in visibility
  - It induces forbidden interval:

$$|t_{ki} \cap [\bar{t}, v]| \geq \frac{\delta_k - \gamma_k + (|t_{ki} \cap [\bar{t}, v]| * \pi_{ki})}{\pi_{ki}}$$



Outline

1 Problem
- Data Transfer

2 Mathematical modeling of transfers
- A Simplified model of Data Transfer

3 Algorithms
- Sweep Algorithm
- Filtering algorithms

4 Benchmarks

Benchmarks

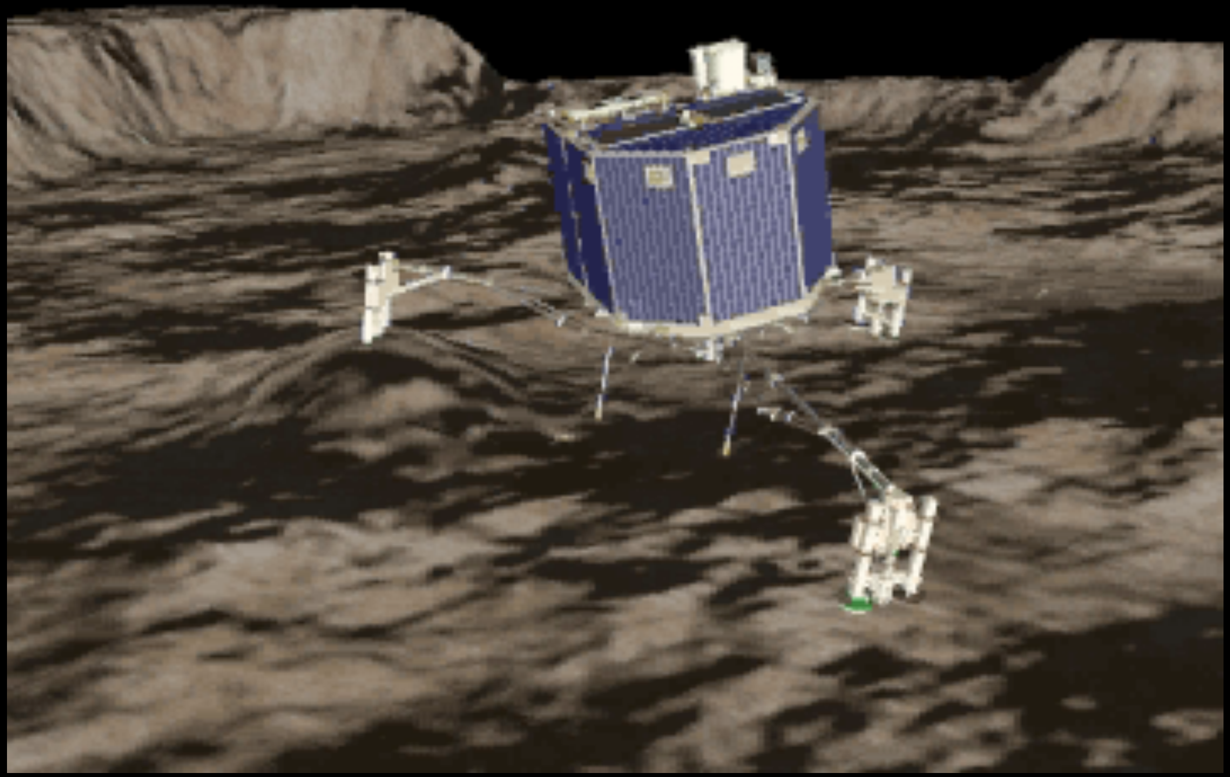
SCENARIO	PARAMETERS		MOST+Ilc-Reservoir		MOST +DataTransfer	
	M_m	Visi.	Init. time (s)	CPU time (s)	Init. time (s)	CPU time (s)
Consert	17456	Periodic	4.06	20.07	0.88	0.08
Consert/Romap	17456	Periodic	11.13	Time out	1.17	0.1
Consert/Romap	37456	Periodic	11.03	Time out	1.17	0.1
SD2/Ptolemy	17456	Periodic	26.71	41.72	3.37	0.09
SD2/Ptolemy	17456	Continuous	32.78	79.48	3.25	0.08
SD2/Cosac/Civa	37456	Periodic	50.20	181.91	2.75	0.14
SD2/Cosac/Civa	17456	Periodic	50.84	179.19	2.95	0.15
SD2/Cosac/Civa	17456	Continuous	25.12	91.08	1.82	0.10

Table: Old vs. new version of MOST on 8 standard scenarios

Resume

Transfer problem
  A better modeling
  A Faster transfers computing
  Need propagation

Results
  Improvement of computing times
  An exact and approximate modeling
  Two filtering algorithms



Thank for your attention