

MUTUAL EXCLUSION

DISTRIBUTED COMPUTING

AKASH BANERJEE

M.S. NISHANTH

CONTENTS

- ▶ Abstract
- ▶ Algorithm Overview
- ▶ Algorithm
- ▶ Illustration
- ▶ Complexity Analysis
- ▶ Comparison
- ▶ Conclusion



ABSTRACT

We propose a mutual exclusion algorithm that uses concepts of request based, token based and quorum based algorithms to try and solve all the demerits of those algorithms, like the request based algorithms are simple and have low message complexity but inefficient as they require complete topologies. Also most of these techniques use broadcasting or flooding schemes which uses a lot of network traffic that could have been used for other important tasks otherwise.

ALGORITHM OVERVIEW

Since most distributed systems come in the form of clusters, we elect a subset of processes from each cluster to be the Leader.

We use coordinator based mutual exclusion within each cluster, and Raymond's token based mutual exclusions amongst the coordinators to achieve mutual exclusion.

Doing so we can have very low message complexity within each cluster, i.e, 3, and even when requests are across clusters we can have fast response time and synchronization by using Raymond's algorithm only amongst the handful of leaders.

Unlike Maekawa's algorithm where every set of leaders(Quorum Members) must have some overlapping we have no such restrictions in this scheme, thus choosing leaders is much easier and flexible, and also avoid its high message complexity of $O(6S)$.

ALGORITHM

When Process wants to enter CS:

1. Send **request** only to its leader.
2. Wait for **grant** from leader.
3. Once **grant** has been received enter CS.
4. Send **release** to leader after exiting CS.

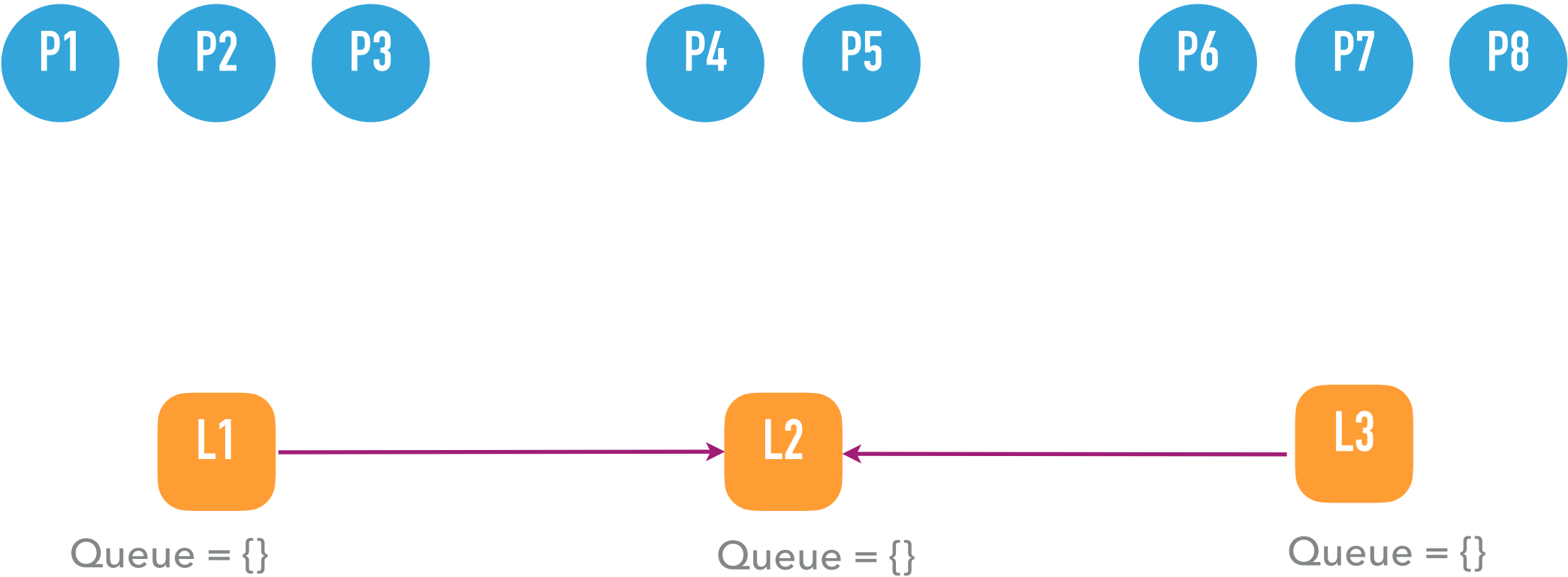
A process will never receive a request from another process.

ALGORITHM

When leader gets request:

1. Push request to queue.
2. While queue is not empty
 - ▶ Dequeue one process.
 - ▶ If process is a child then
 - ▶ If have key then send it.
 - ▶ Else acquire key following Raymond's algorithm and then send it.
 - ▶ If have key then send it to the process
 - ▶ Else acquire key following Raymond's algorithm and then send.

AN ILLUSTRATION

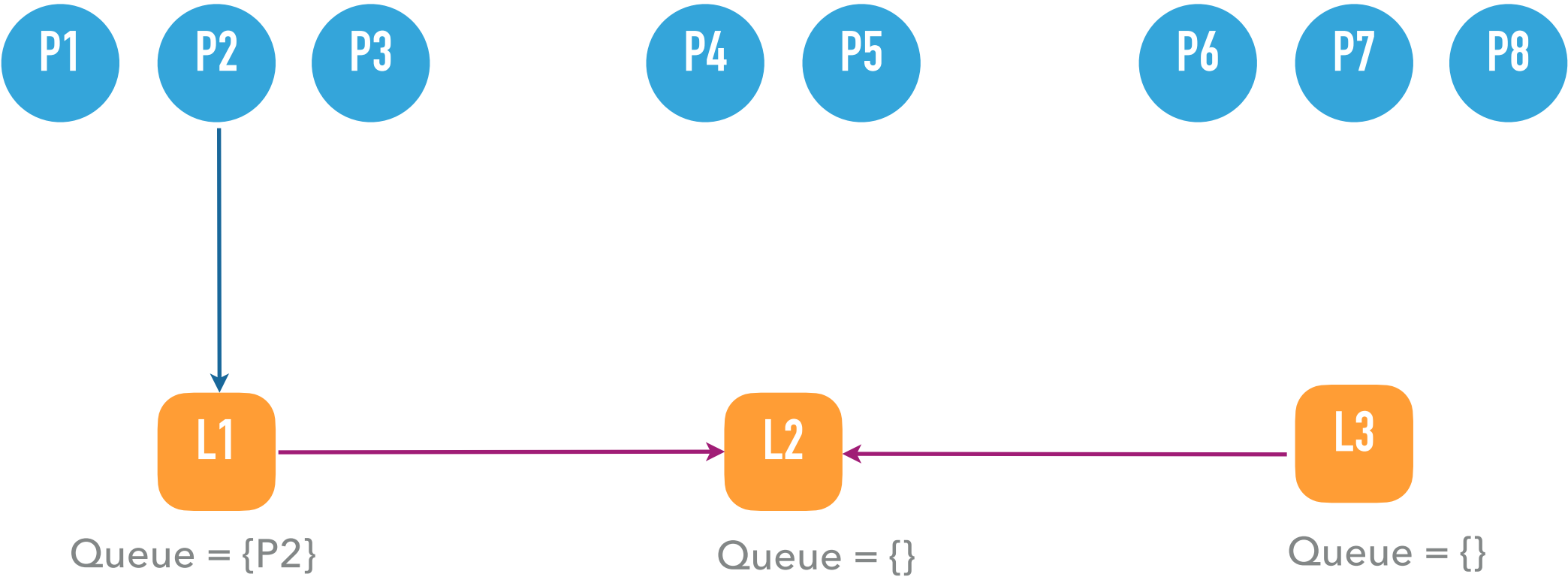


Leader for process P1, P2 & P3 is L1
Leader for process P4 & P5 is L2
Leader for process P6, P7 & P8 is L3

Legend:

- Process (blue circle)
- Leader (orange circle)
- Request (blue arrow)
- Greet (green arrow)
- Release (orange arrow)
- MST Edge (purple arrow)

AN ILLUSTRATION

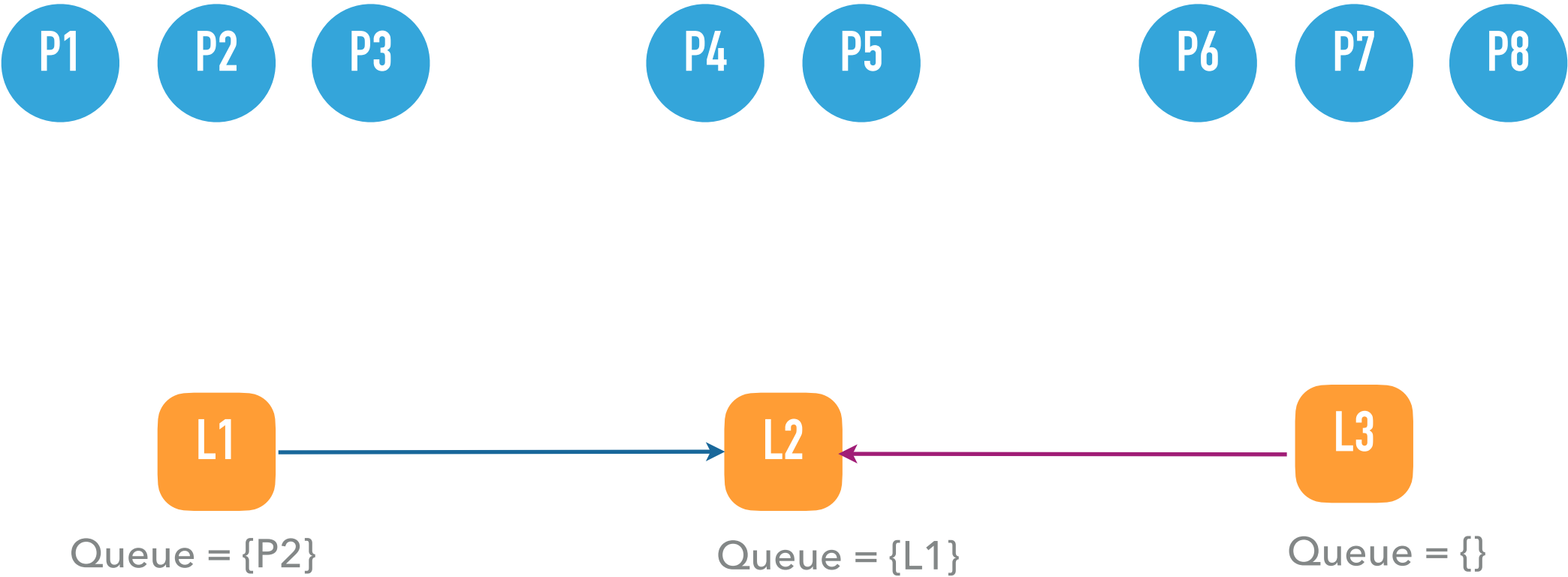


P2 sends a request to L1

Legend:

- Process (blue circle)
- Leader (orange circle)
- Grant (green arrow)
- Release (orange arrow)
- Request (blue arrow)
- MST Edge (purple arrow)

AN ILLUSTRATION

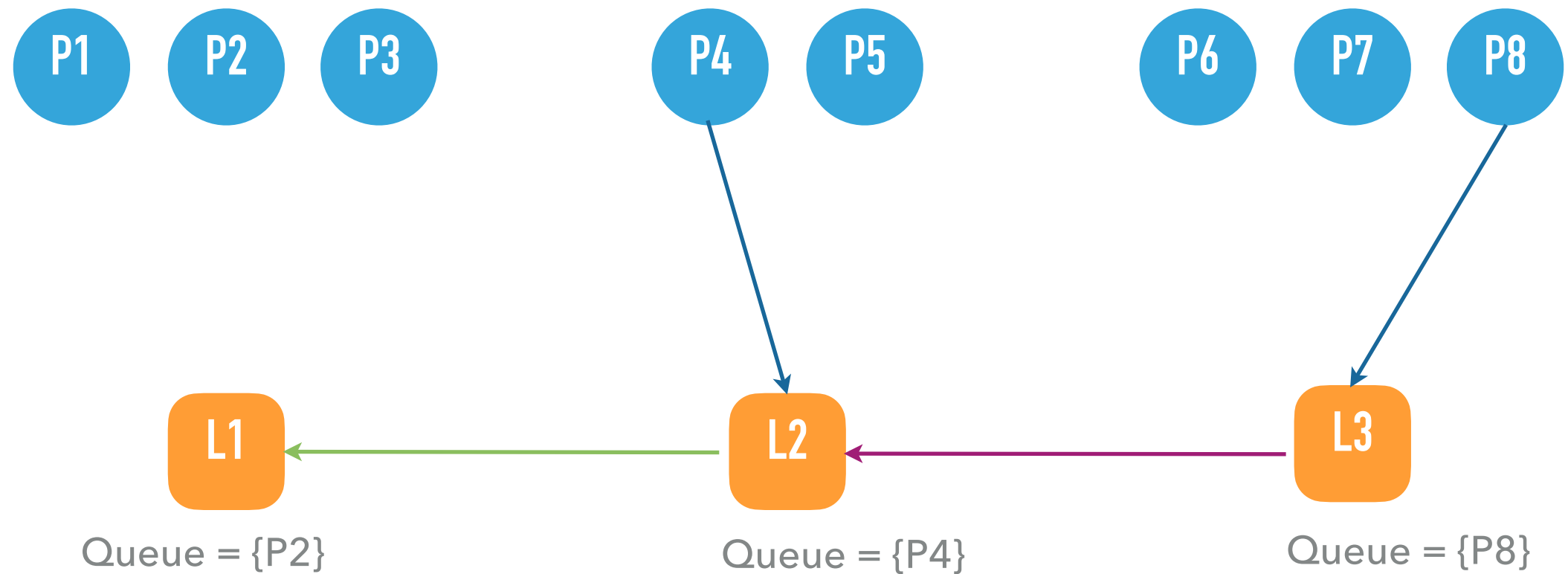


L1 requests token from L2

Legend:

- Process
- Leader
- Request
- Grant
- Release
- MST Edge

AN ILLUSTRATION

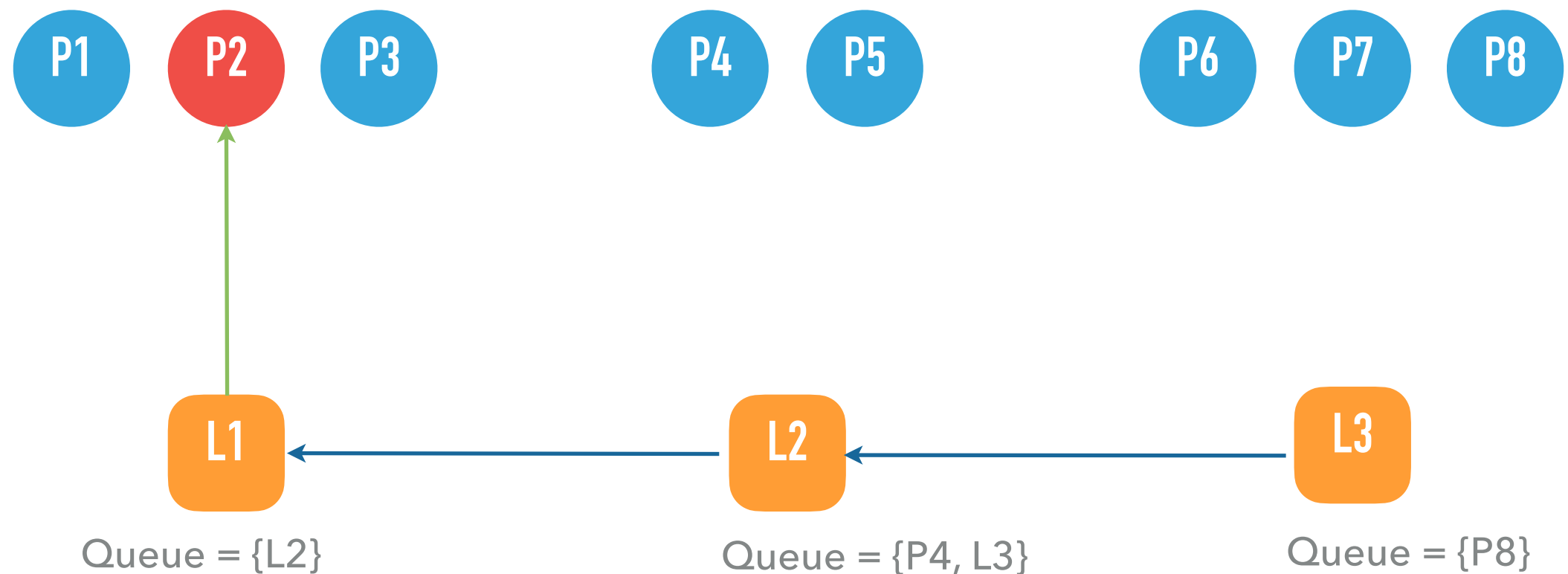


L2 sends token to L1
P8 sends request to L3
P4 sends request to L2

Legend:

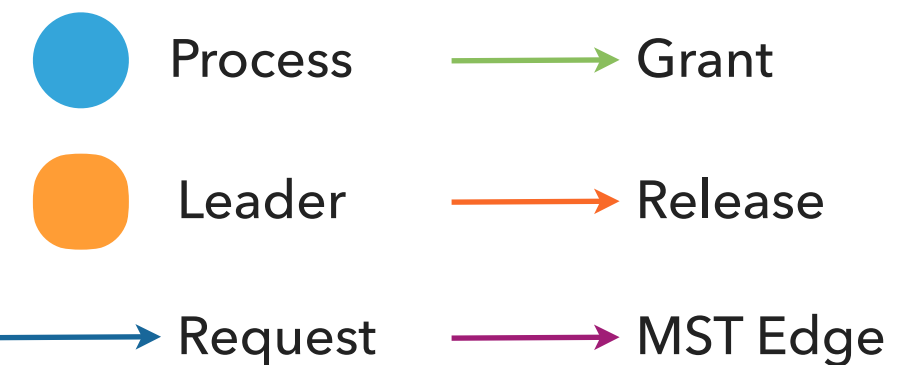


AN ILLUSTRATION

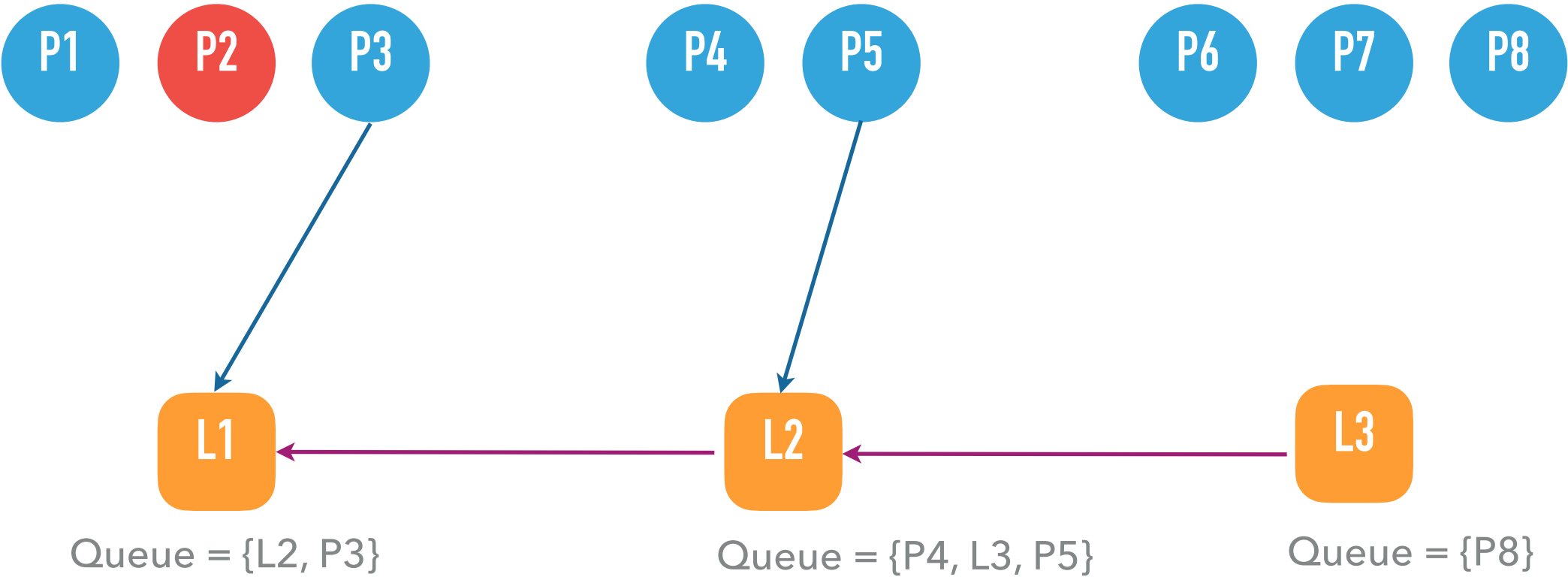


L1 sends grant to P2
P2 enter CS
L3 request token from L2
L2 requests token from L1

Legend:



AN ILLUSTRATION

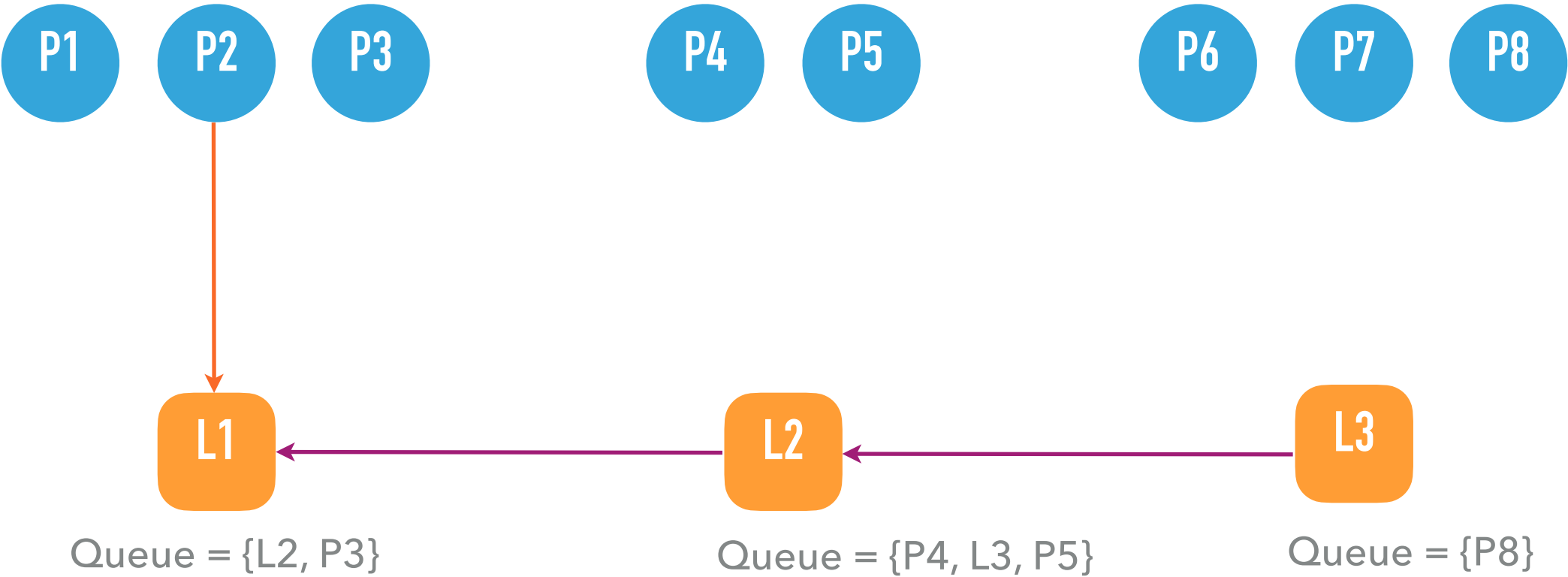


P3 sends request to L1
P5 sends request to L2

Legend:

- Process
- Leader
- Request
- Grant
- Release
- MST Edge

AN ILLUSTRATION

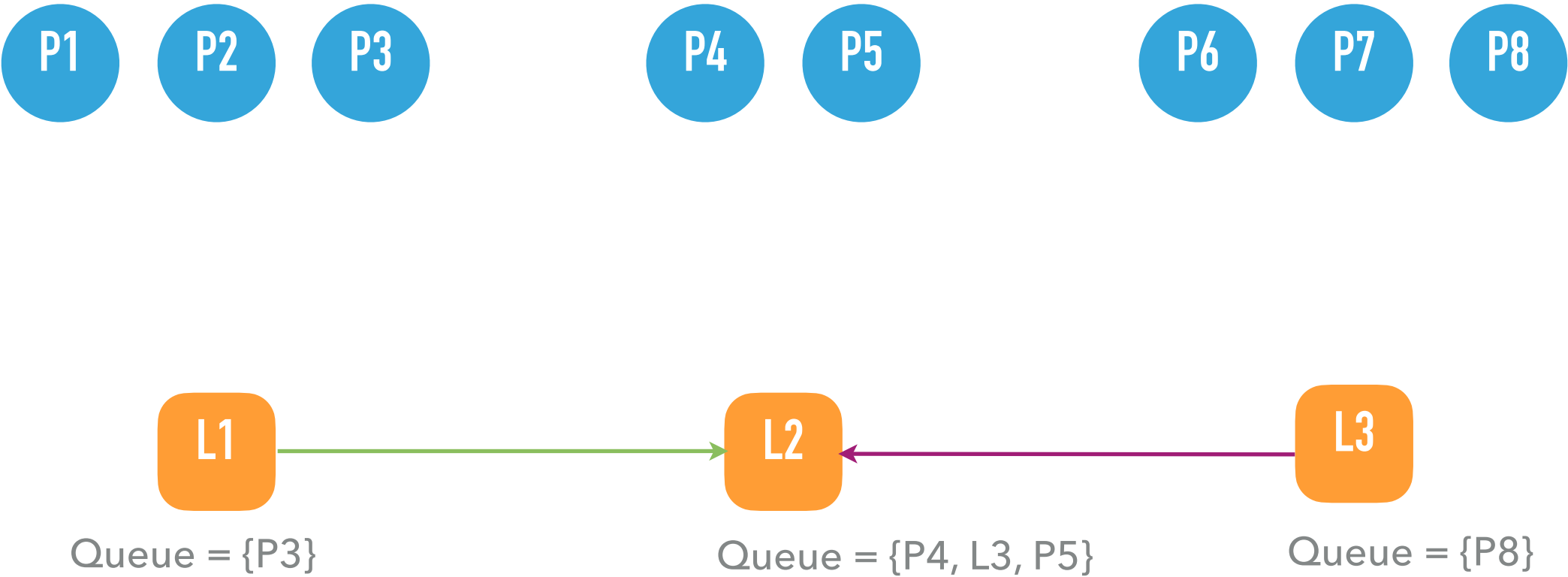


P2 exits CS
P2 sends release to L1

Legend:







- Process (blue circle)
- Leader (orange circle)
- Grant (green arrow)
- Release (red arrow)
- Request (blue arrow)
- MST Edge (purple arrow)

AN ILLUSTRATION

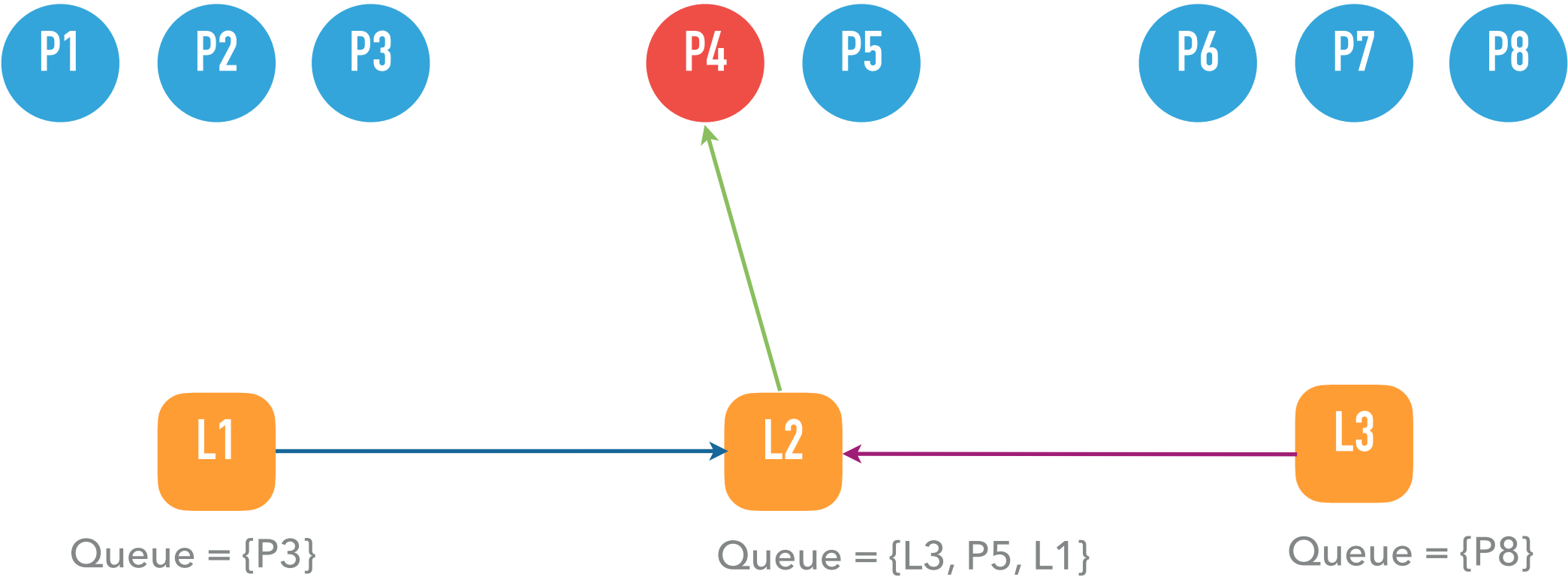


L1 sends token to L2

Legend:

	Process		Grant
	Leader		Release
	Request		MST Edge

AN ILLUSTRATION

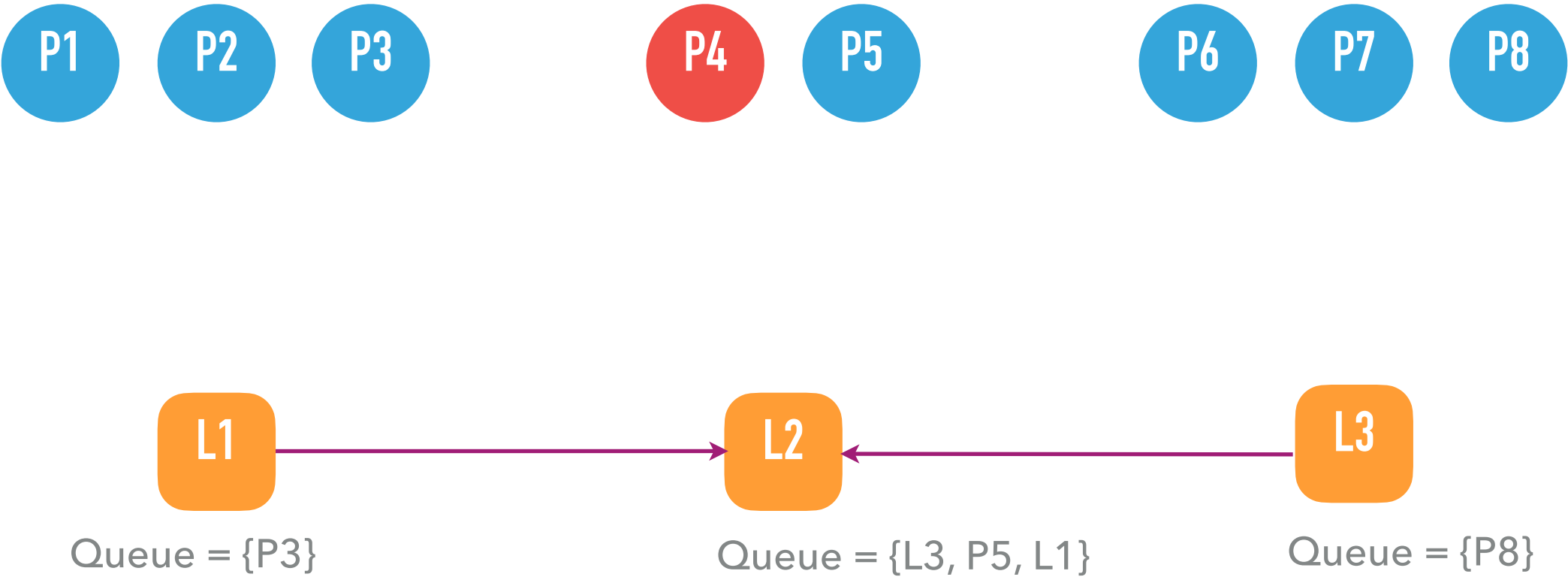


L2 sends grant to P4
P4 enter CS
L1 requests token from L2

Legend:

- Process (blue circle)
- Leader (orange circle)
- Grant (green arrow)
- Release (orange arrow)
- Request (blue arrow)
- MST Edge (purple arrow)

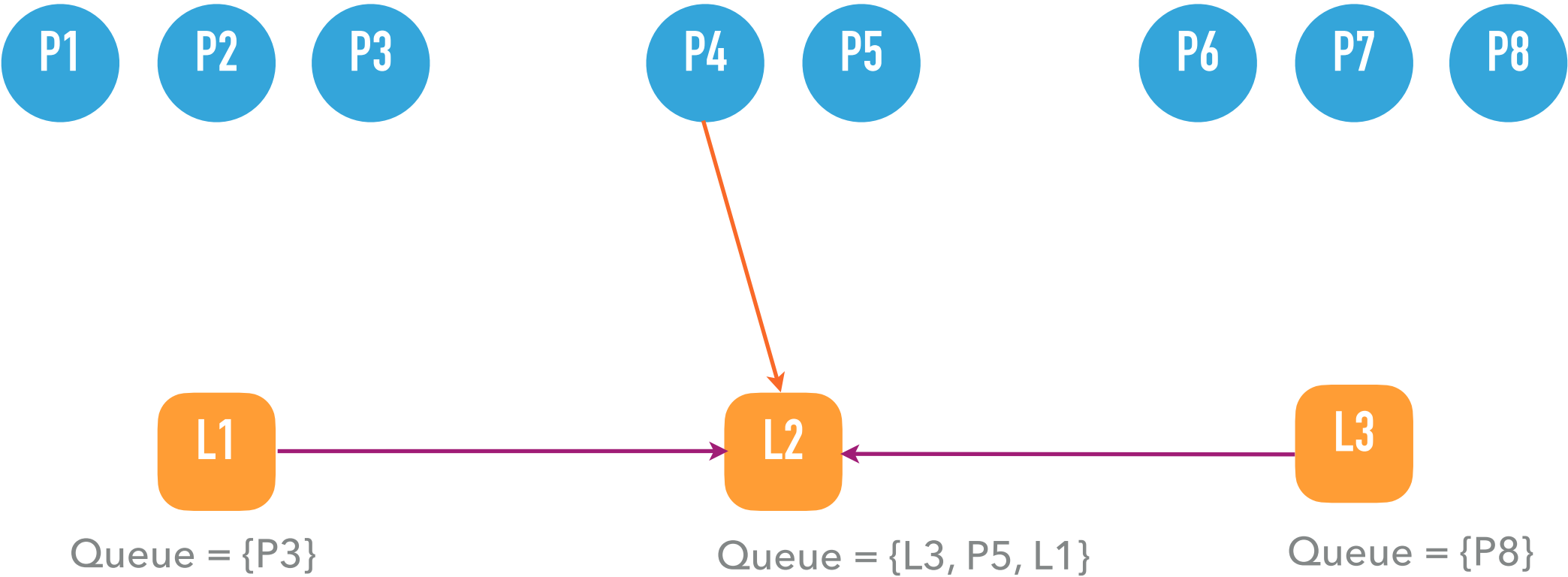
AN ILLUSTRATION



Legend:







- Process
- Leader
- Request
- Grant
- Release
- MST Edge

AN ILLUSTRATION

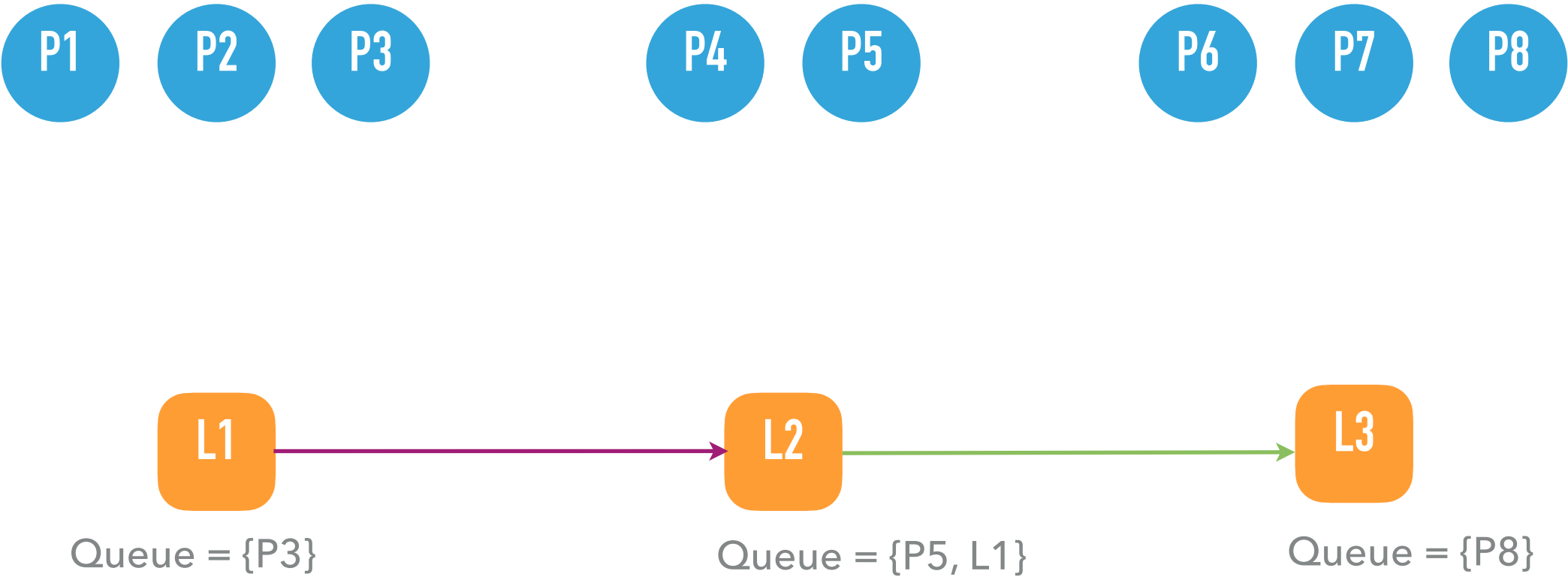


P4 exits CS
P4 sends release to L2

Legend:

	Process		Grant
	Leader		Release
	Request		MST Edge

AN ILLUSTRATION

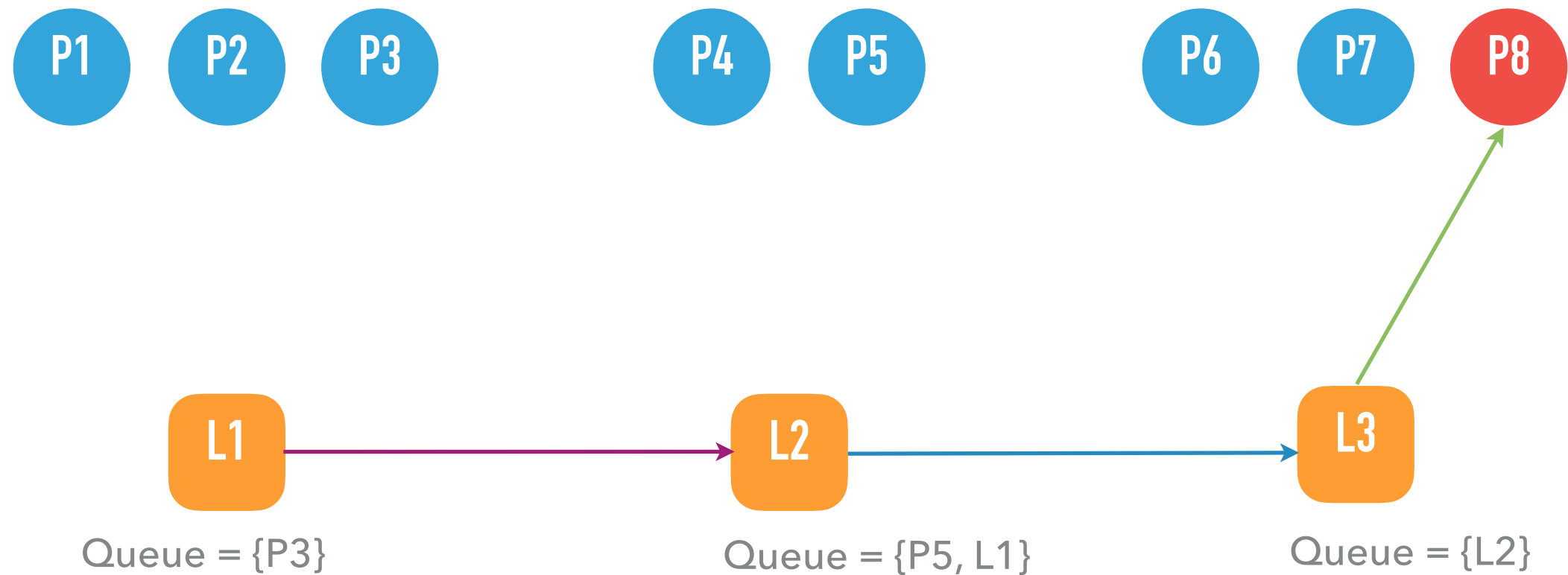


L2 sends token to L3

Legend:

- Process (blue circle)
- Leader (orange circle)
- Request (blue arrow)
- Grant (green arrow)
- Release (orange arrow)
- MST Edge (purple arrow)

AN ILLUSTRATION

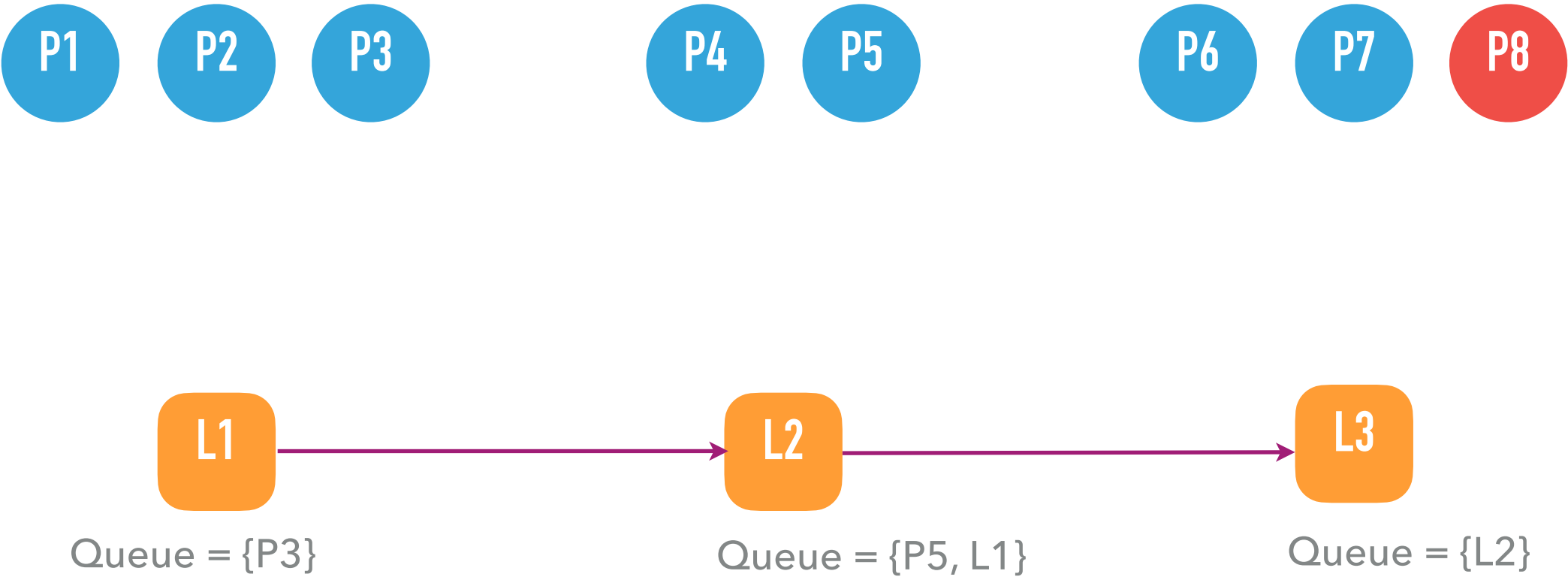


L3 sends grant to P8
P8 enters CS
L2 requests token from L3

Legend:



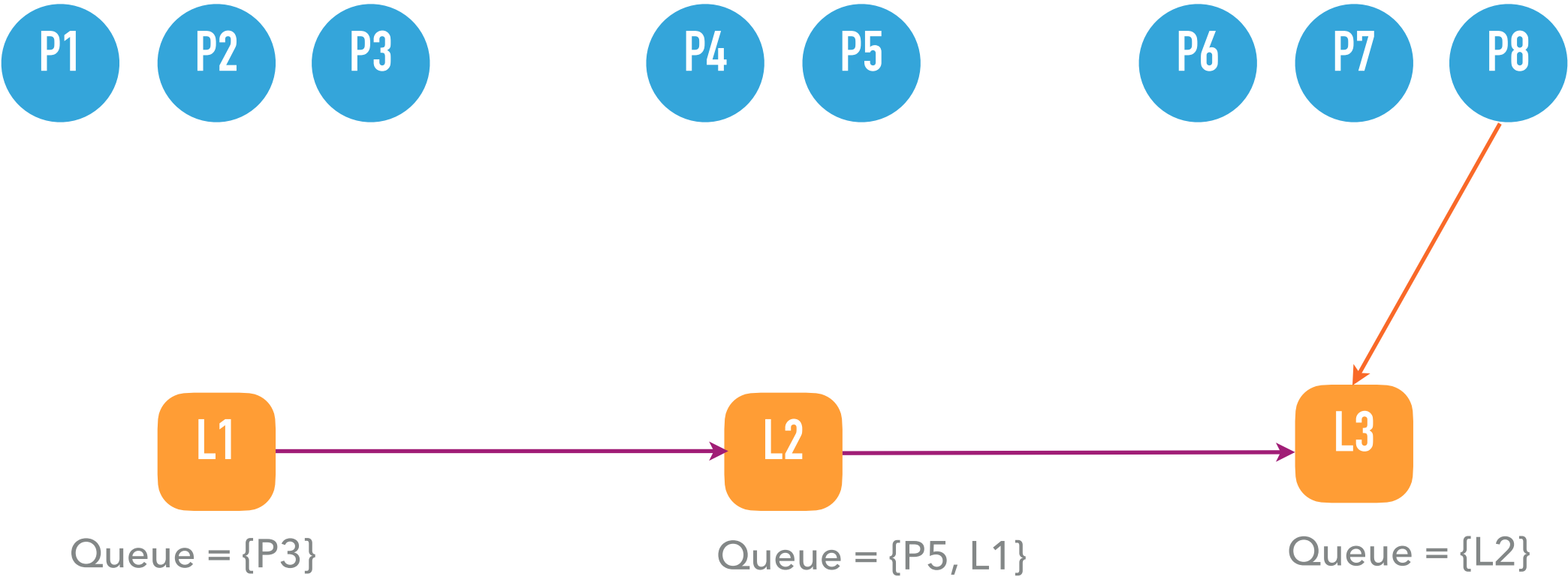
AN ILLUSTRATION



Legend:







- Process (blue circle)
- Leader (orange circle)
- Grant (green arrow)
- Release (orange arrow)
- Request (blue arrow)
- MST Edge (purple arrow)

AN ILLUSTRATION

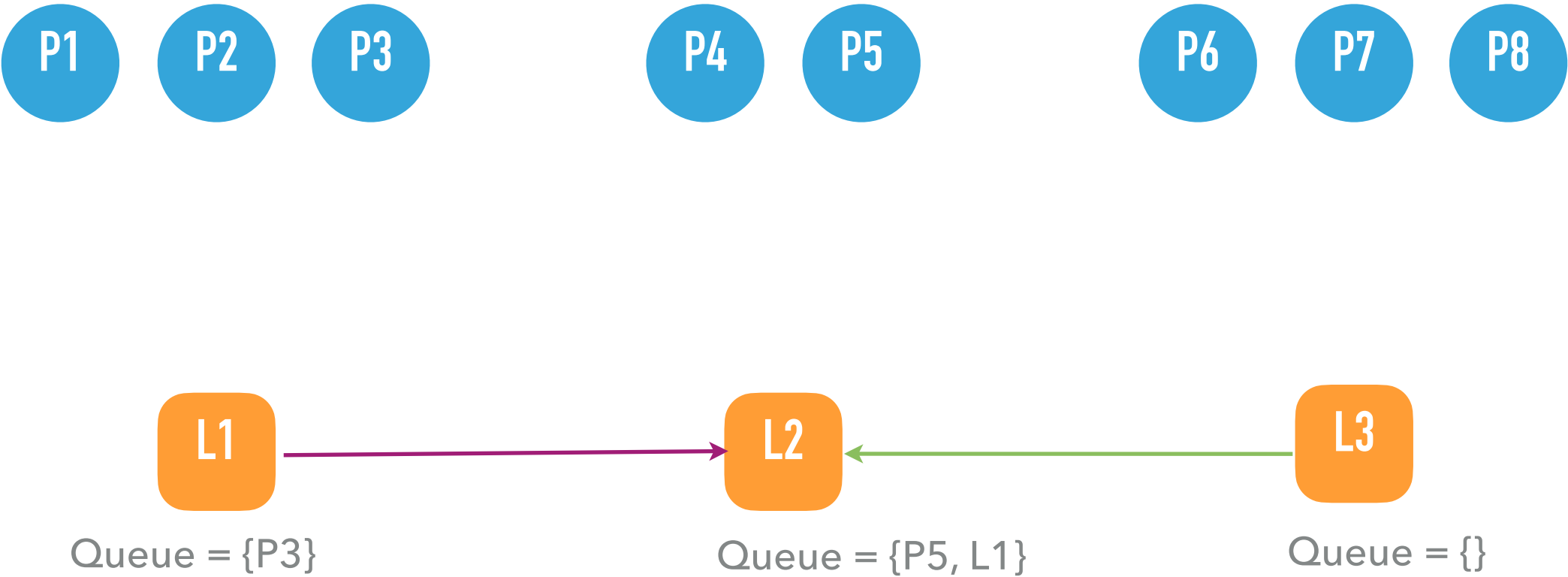


P8 exits CS
P8 sends release to L3

Legend:

	Process		Grant
	Leader		Release
	Request		MST Edge

AN ILLUSTRATION

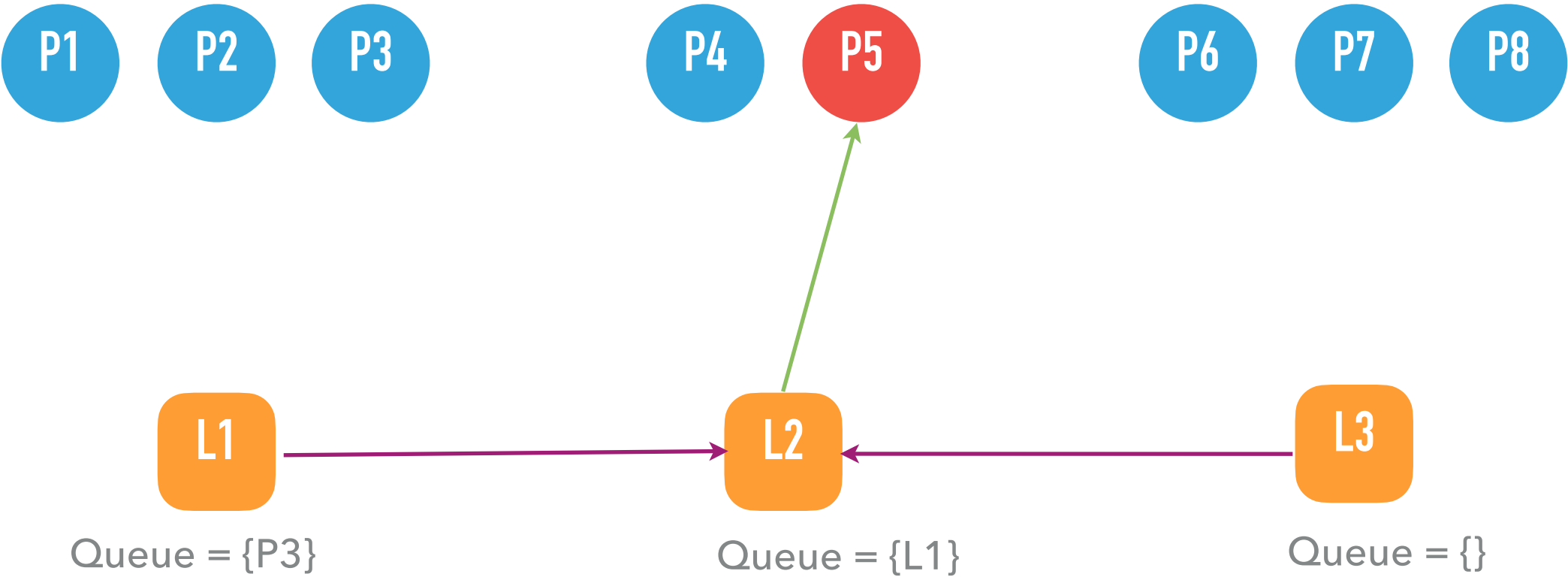


L3 sends token to L2

Legend:

- Process
- Leader
- Request
- Grant
- Release
- MST Edge

AN ILLUSTRATION

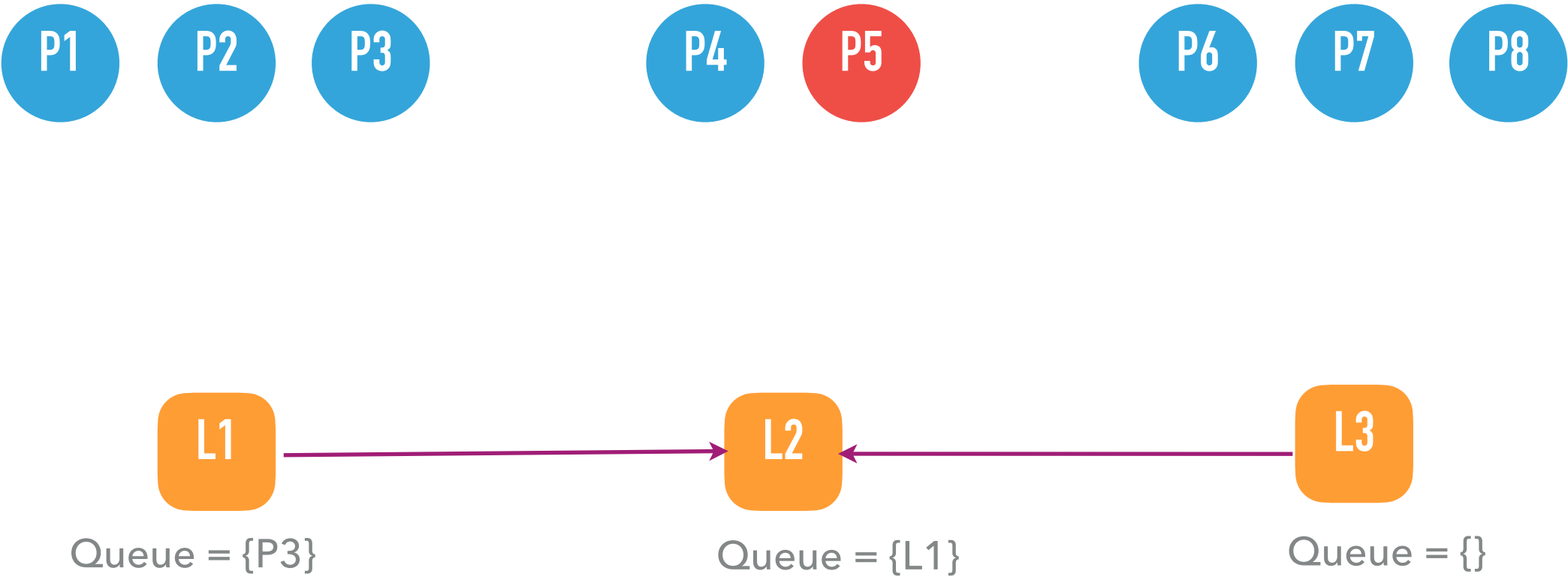


L2 sends grant to P5
P5 enters CS

Legend:

- Process (blue circle)
- Leader (orange circle)
- Grant (green arrow)
- Release (orange arrow)
- Request (blue arrow)
- MST Edge (purple arrow)

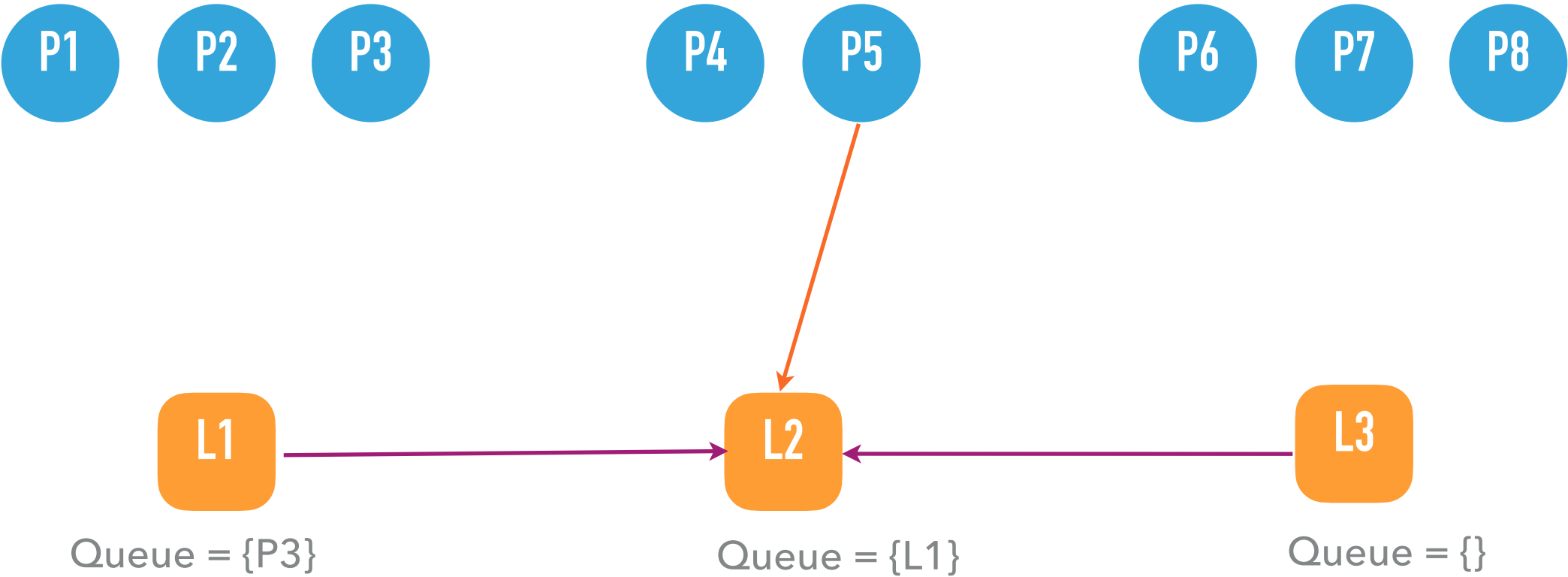
AN ILLUSTRATION



Legend:







- Process
- Leader
- Request
- Grant
- Release
- MST Edge

AN ILLUSTRATION

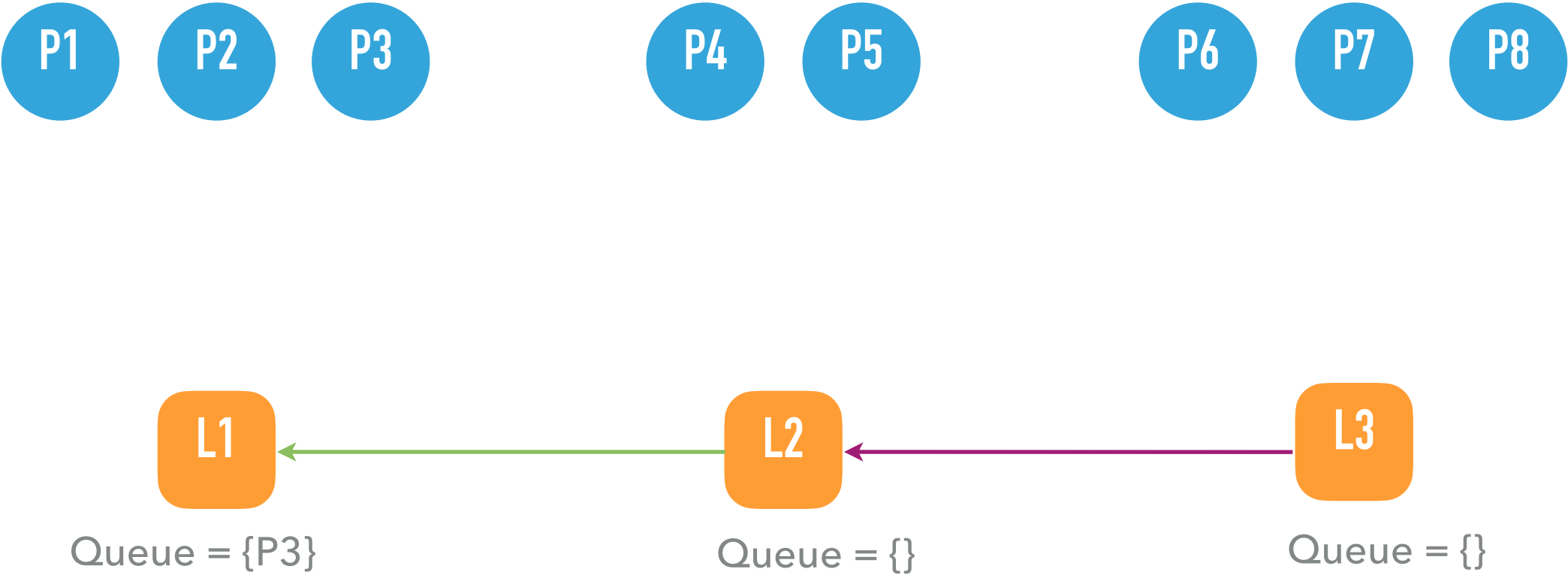


P5 exits CS
P5 sends release to L2

Legend:

	Process		Grant
	Leader		Release
	Request		MST Edge

AN ILLUSTRATION

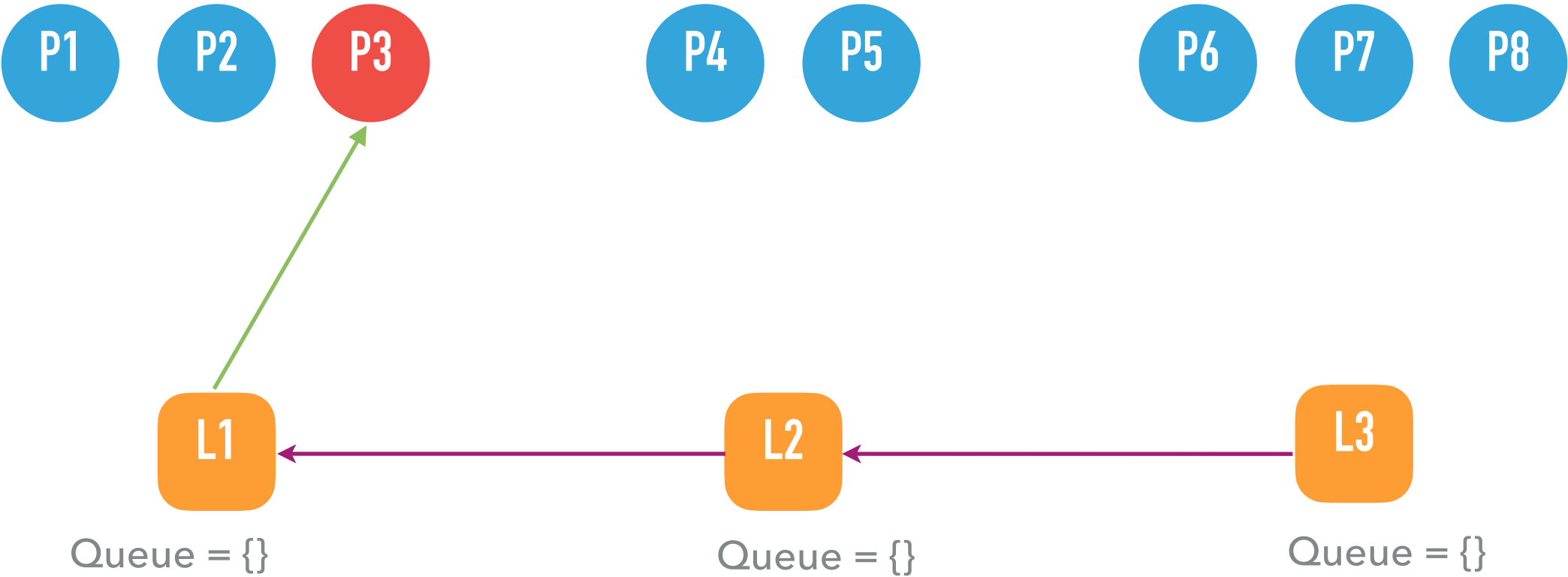


L2 sends token to L1
P5 sends release to L2

Legend:

- Process
- Leader
- Request
- Grant
- Release
- MST Edge

AN ILLUSTRATION

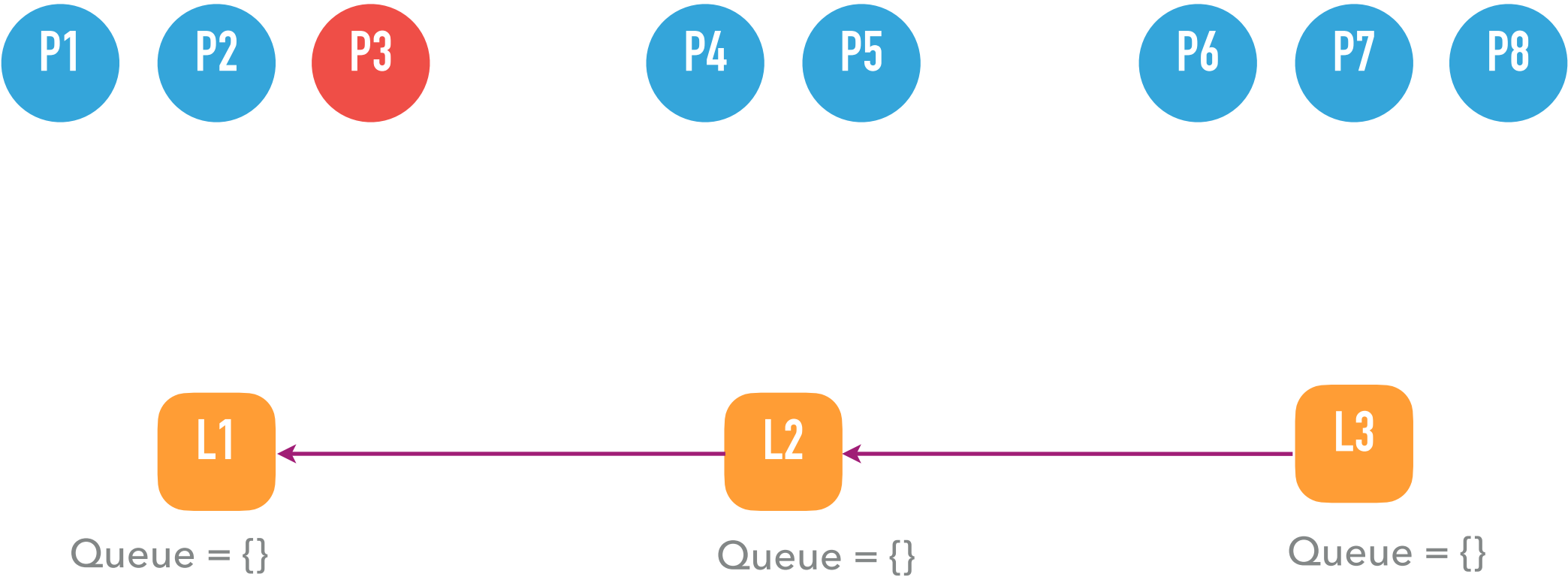


L1 sends grant P3
P3 enters CS

Legend:

- Process (blue circle)
- Leader (orange circle)
- Grant (green arrow)
- Release (orange arrow)
- Request (blue arrow)
- MST Edge (purple arrow)

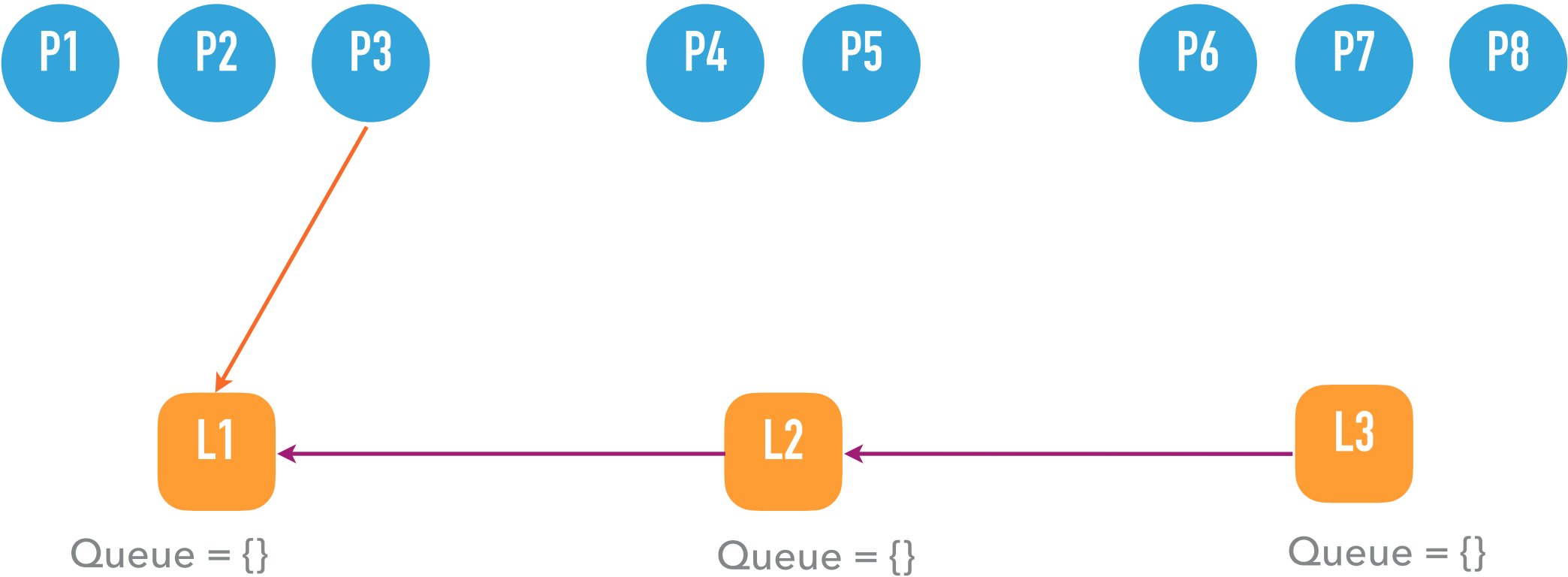
AN ILLUSTRATION



Legend:







- Process
- Leader
- Grant
- Release
- Request
- MST Edge

AN ILLUSTRATION

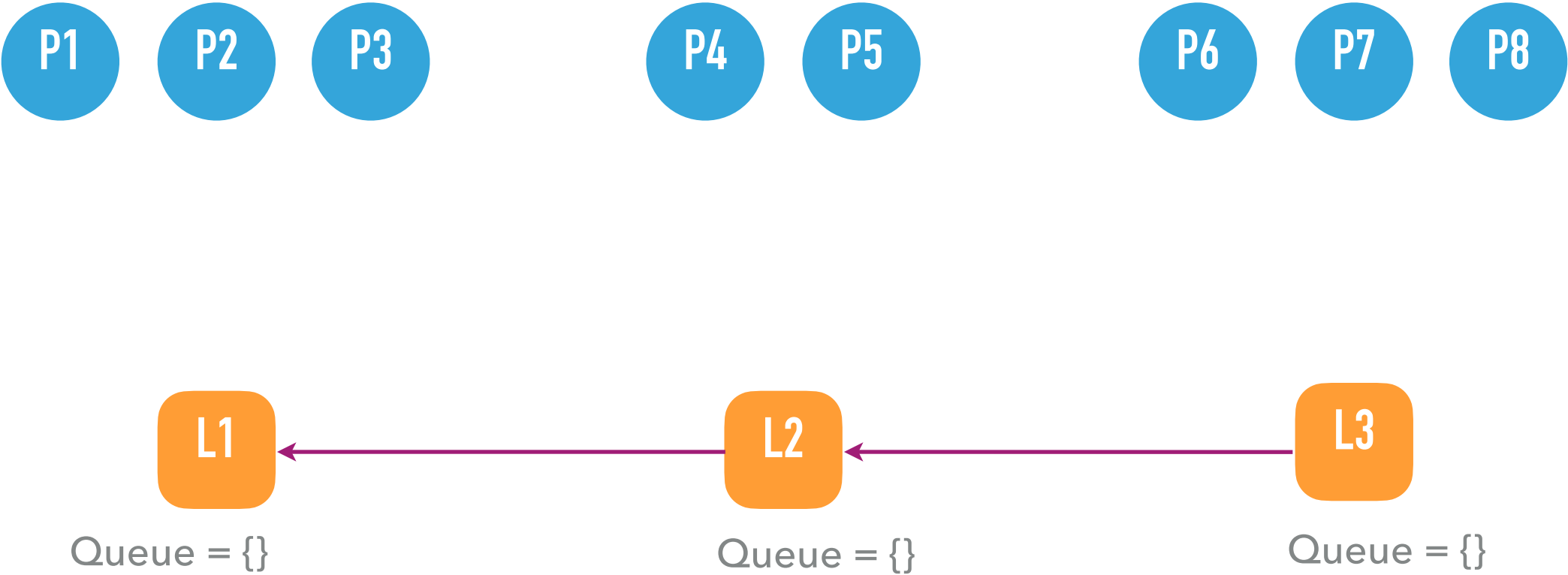


P3 exits CS
P3 sends release to CS

Legend:

	Process		Grant
	Leader		Release
	Request		MST Edge

AN ILLUSTRATION



Legend:

- Process
- Leader
- Grant
- Release
- Request
- MST Edge

COMPLEXITY ANALYSIS

- Parameters:

N: Number of Processes in the System.

T: Message Transmission time.

E: Critical Section Execution time.

L: No of Leaders in the System.

D: Diameter of the Spanning tree on Leaders ($O(L)$).

- Message Complexity:

Best Case: 3

Worst Case: $2D + 3$

- Message Size Complexity: $O(1)$.

- Response Time (Under Light Load):

- Best Case: $2T + E$

- Worst Case: $2D * T + 2T + E$

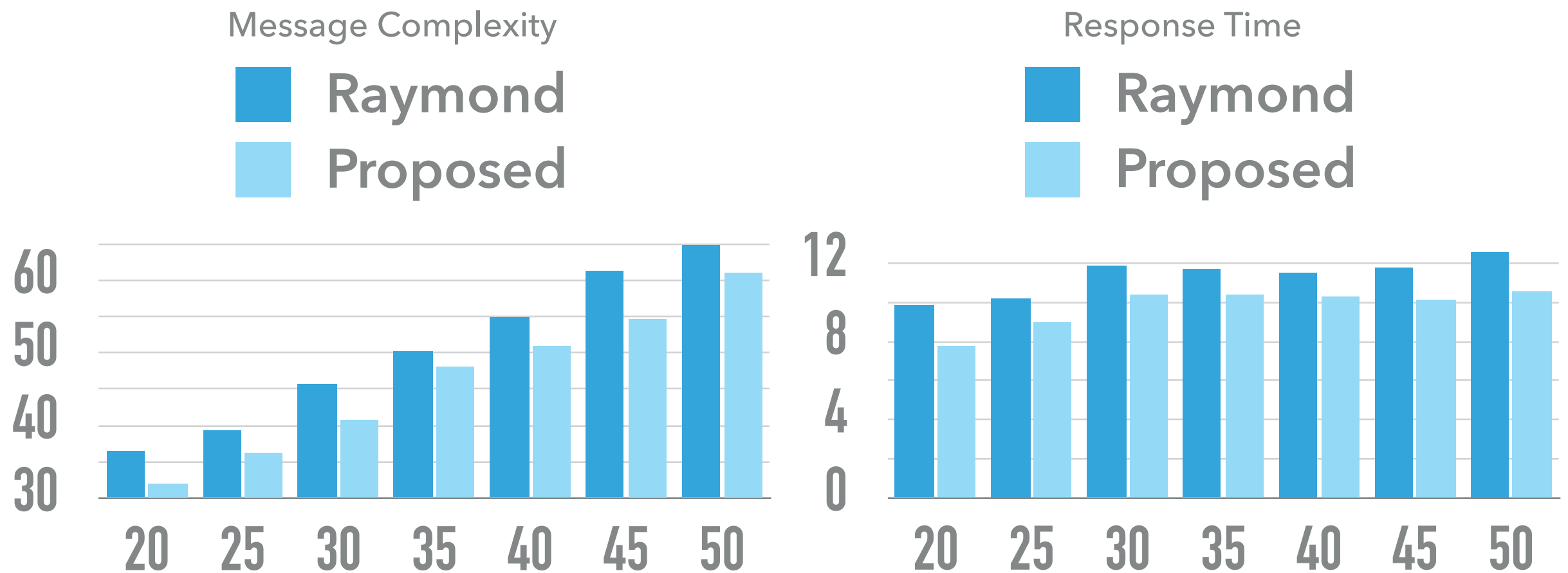
- Synchronization Delay:

- Best Case: $2E$

- Worst Case: $D * T + 2T$

COMPARISON

Comparing our results to Raymond's Algorithm



CONCLUSION

The comparison results shown in the previous slide don't do justice to our algorithm, as firstly the number of processes is very small to show much difference, and also since the comparison was made on a single system, time taken to pass messages was negligible, however in real world scenario this may not be the case thus further separating the results.

Also our algorithm performs best when the number of leaders in the system is very less compared to the total number of processes in the system, and the system is divided into clusters.

EXPANDING THE ALGORITHM

- If leader is experiencing high load, we can easily appoint another process as leader and split the load between the two.
- Instead of having just two levels we can have multiple levels, and have a separate token for each level.
- A token will never cross a level, i.e, a token will never pass from a process of one level to a process of another level.
- We are working to find an efficient solution to use Raymonds within each level and Coordinators across levels which can hopefully relax the topological requirements even further and also provide improvements in the various complexities.

ANY QUESTIONS?

Thank You