Managing Layers in Flatland

Flatland Layers

- Flatland was created to be the main 2D robotics simulator to be used with ROS.
- The concept of layers turns Flatland into a “2.5D simulator” by simplifying the 3rd dimension to a discrete space.
- Layers work independently from each other in terms of physics.
- Each component of the simulator is contained in one to many layers and two components only collide if they have at least one layer in common.
- Flatland worlds allow up to 16 layers.

Problems managing layers

- The only default types of components that need to use layers are the maps, the models and the laser plugin.
- Models may collide or not with each other or the map and radars need to define what needs to be detected by them.
- Lasers can detect the model they are contained in. Example:
    - Square box model with radar inside. The radar can only detect its own model.
- The different bodies of a model may not all have the same collisions.
- Managing layers can quickly become a very complex task very prone to errors.

How to manage layers

The goal is to respect all collisions with the minimum number of layers possible.

Methodological approach:

- Build an undirected graph to represent all collisions between models and maps.
- Group the nodes in layers.
- Add the lasers one by one and add new layers if necessary.
- Apply the necessary layers to each component.

Building the Collision Graph - Manage Maps

- In the world.yaml file, each layer is assigned to a map.
- Maps are defined by an image or a .dat file with line segments.
- If a layer is inside of a map, it interacts with his walls.

Example of the layer configuration in the world.yaml file:

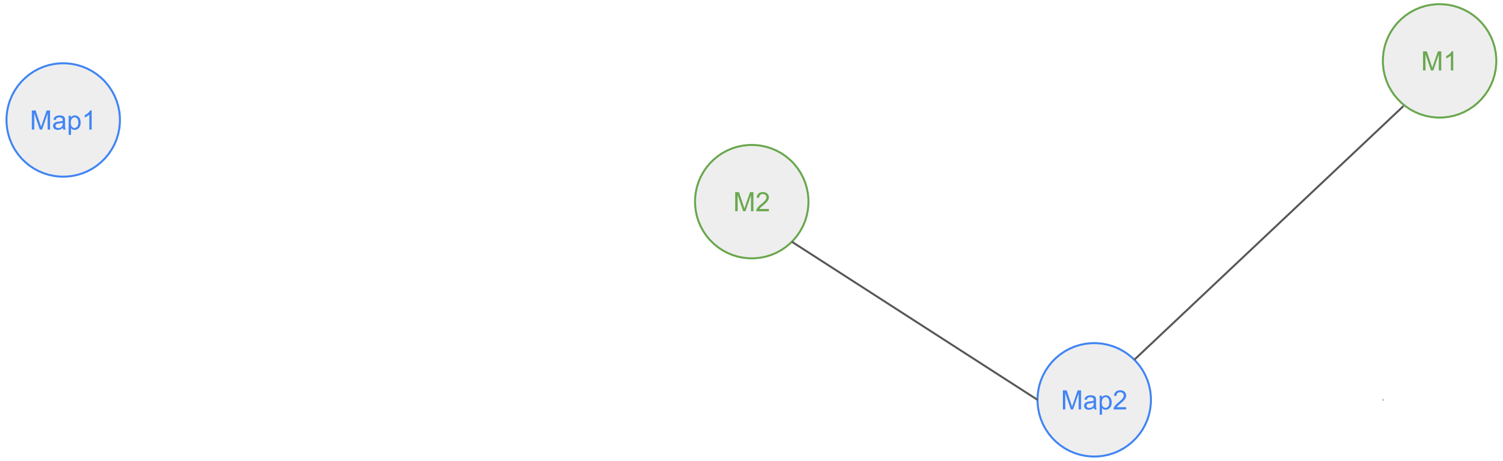
```
layers:
 # map 1
 - name: [<list of layers that interact with this map>]
   map: <image or dat file>
 # map 2, only needs more than one map if there are parts with different interactions
 - name: [<list of layers that interact with this map>]
   map: <image or dat file>
 # map 3, empty map; add if there are components that don’t interact with any maps
 - name: [<list of layers that interact with this map>]
   map: <empty image or dat file>
```

Building the Collision Graph - Add Maps

Add one node for each map except for the empty map.

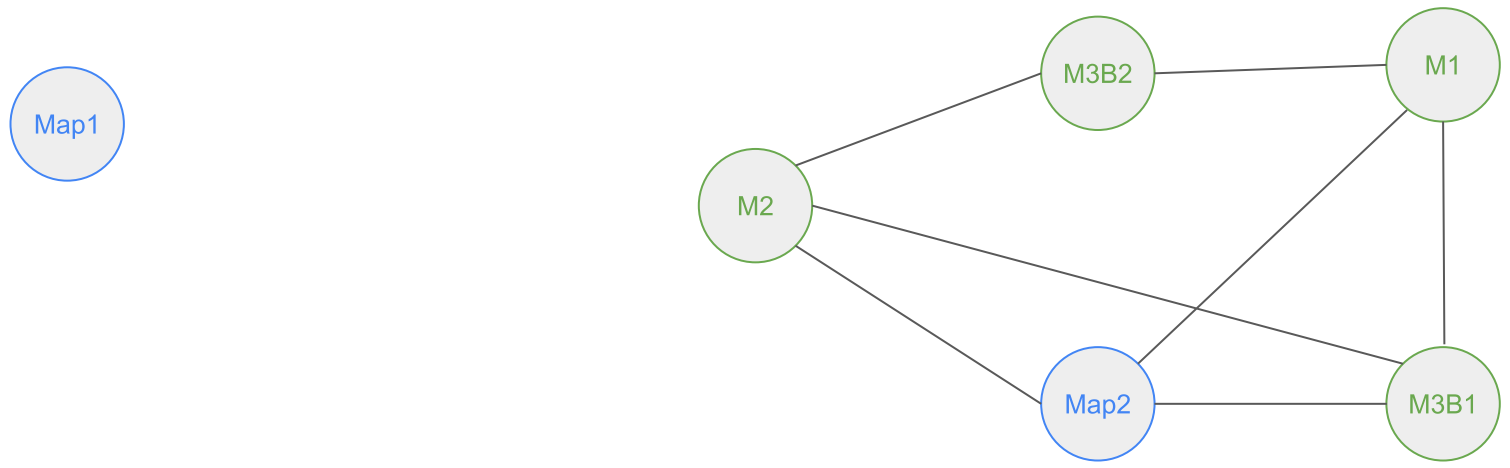
Building the Collision Graph - Add Models

Add the models one by one and connect them to the nodes they collide with.



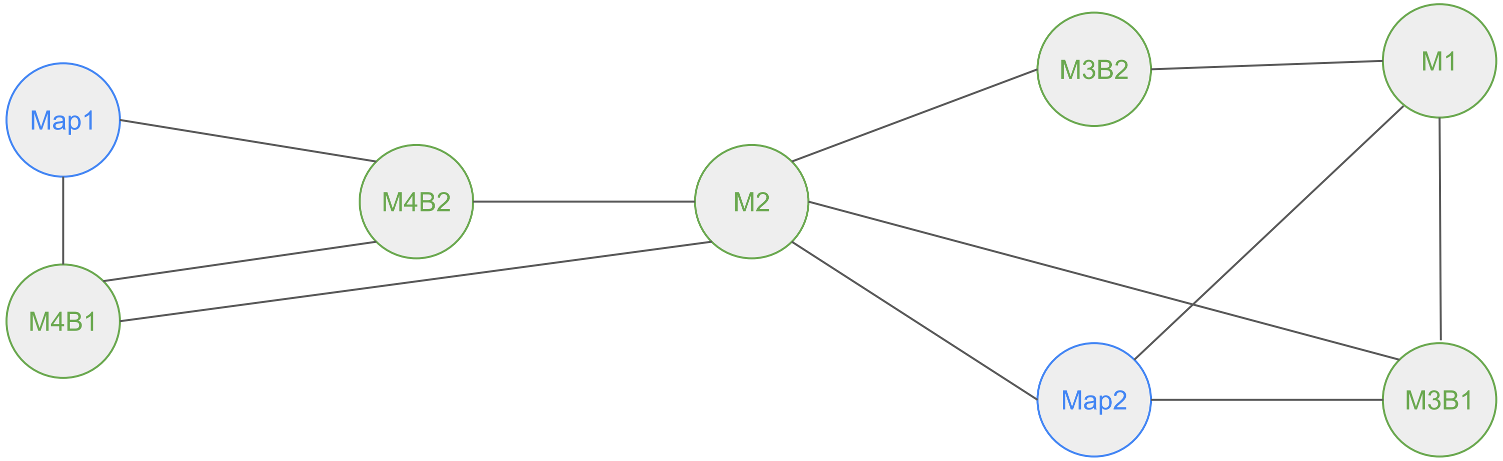
Building the Collision Graph - Add Models

Model 3 has 2 groups of bodies that have different collisions so they need to be separated. There are no collisions inside the same model, even if the layer is the same but, to avoid unnecessary layers, they should not be connected.



Building the Collision Graph - Add Models

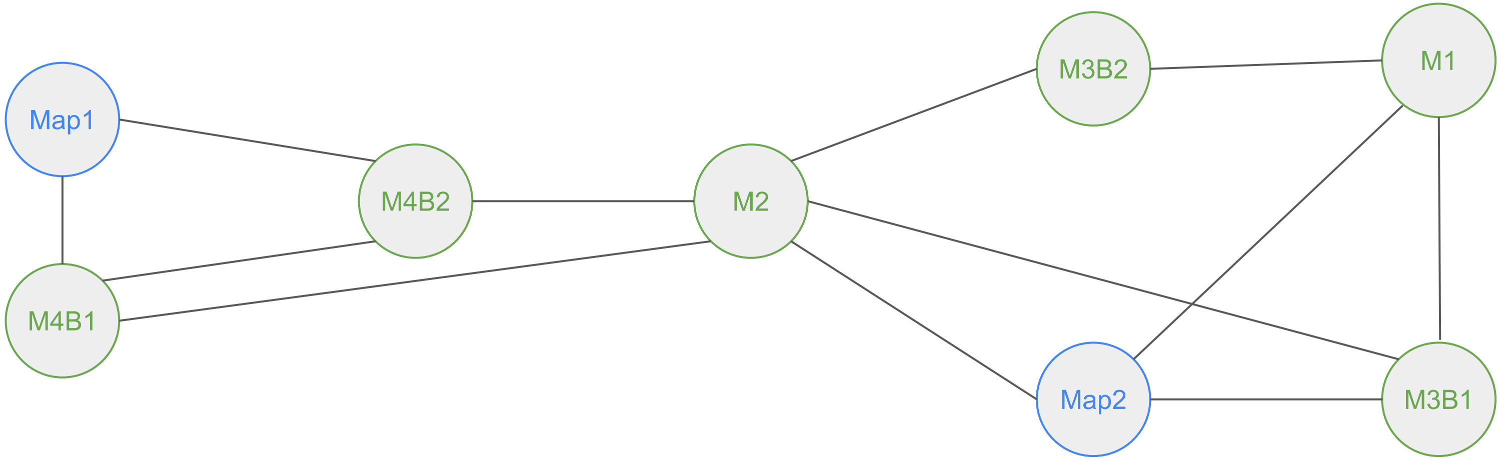
Model 4 has 1 group of bodies that needs to be detected by a laser and other that doesn't. They also need to be separated but, in this scenario, they interact the same way in terms of collisions so, to save layers, they should be connected.



Building the Collision Graph - Finding layers

The goal now is to find the minimum number of layers needed. This is done by obtaining the minimum number of complete subgraphs* that contain all the edges. Start by getting a list of all edges.

Edges:
Map1-M4B1
Map1-M4B2
M4B1-M4B2
M4B1-M2
M4B2-M2
M2-Map2
M2-M3B1
M2-M3B2
Map2-M3B1
Map2-M1
M3B1-M1
M3B2-M1

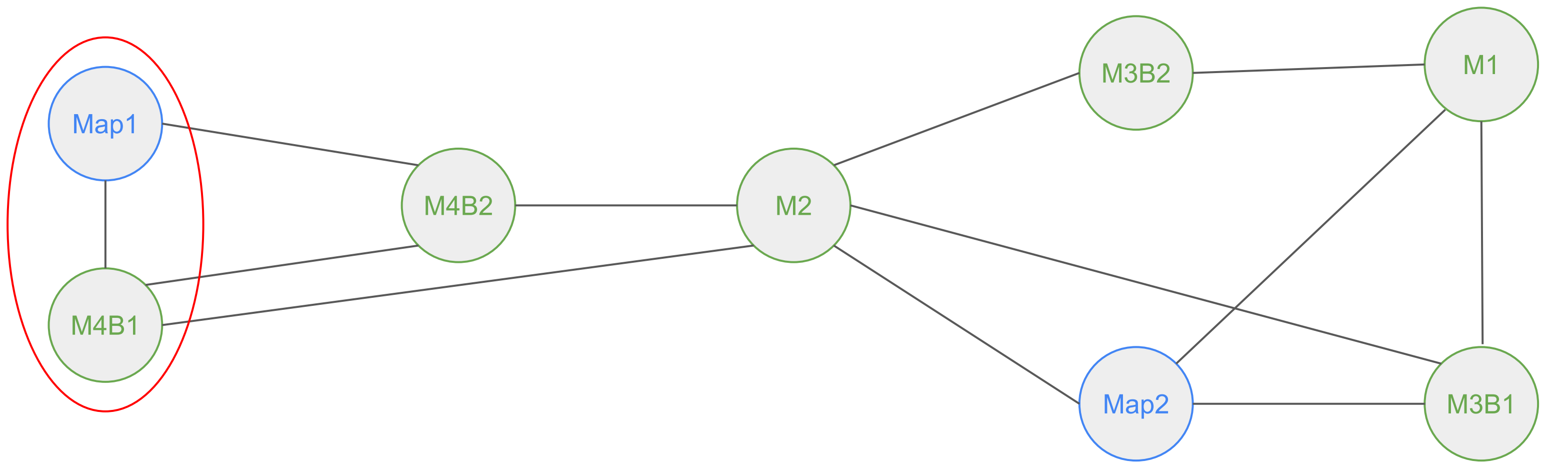


*Complete graph - each vertex is connected to every other vertex

Building the Collision Graph - Finding layers

Choose one of the edges and group their vertices.

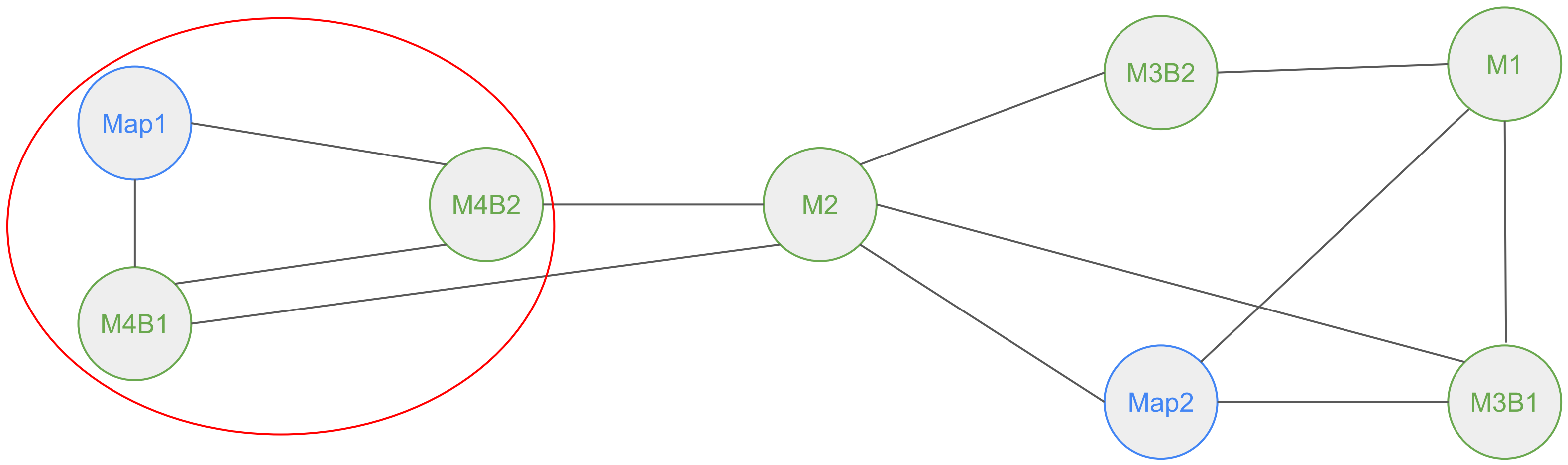
Edges:
Map1-M4B1
Map1-M4B2
M4B1-M4B2
M4B1-M2
M4B2-M2
M2-Map2
M2-M3B1
M2-M3B2
Map2-M3B1
Map2-M1
M3B1-M1
M3B2-M1



Building the Collision Graph - Finding layers

Add to the group all possible vertices so that the group remains a complete graph.

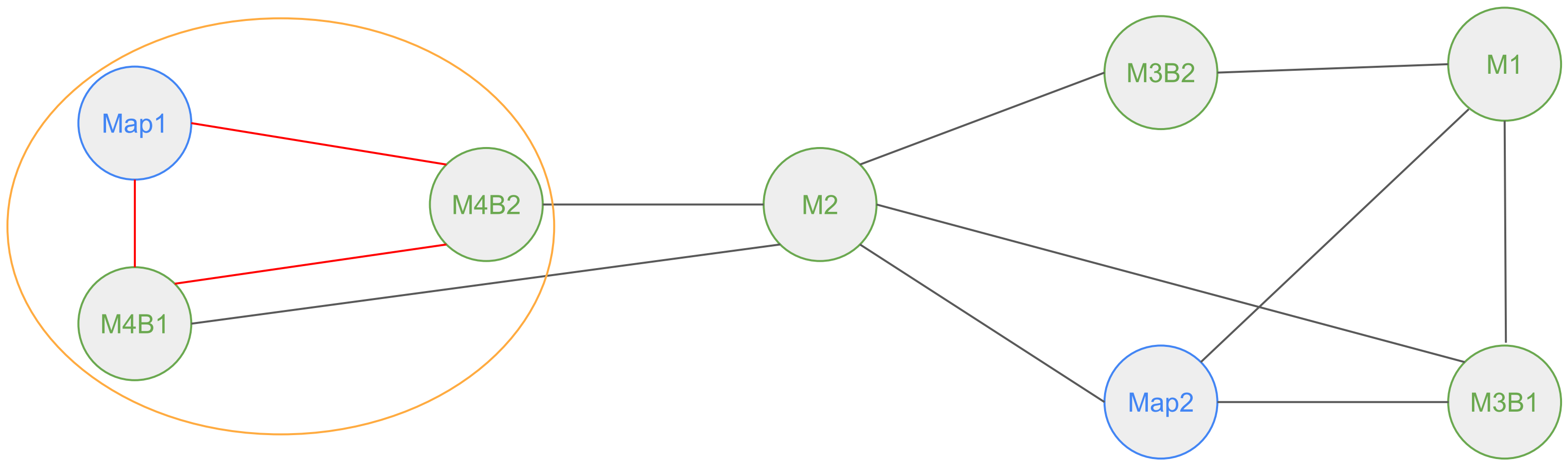
Edges:
Map1-M4B1
Map1-M4B2
M4B1-M4B2
M4B1-M2
M4B2-M2
M2-Map2
M2-M3B1
M2-M3B2
Map2-M3B1
Map2-M1
M3B1-M1
M3B2-M1



Building the Collision Graph - Finding layers

Scratch all edges contained in the group from the list.

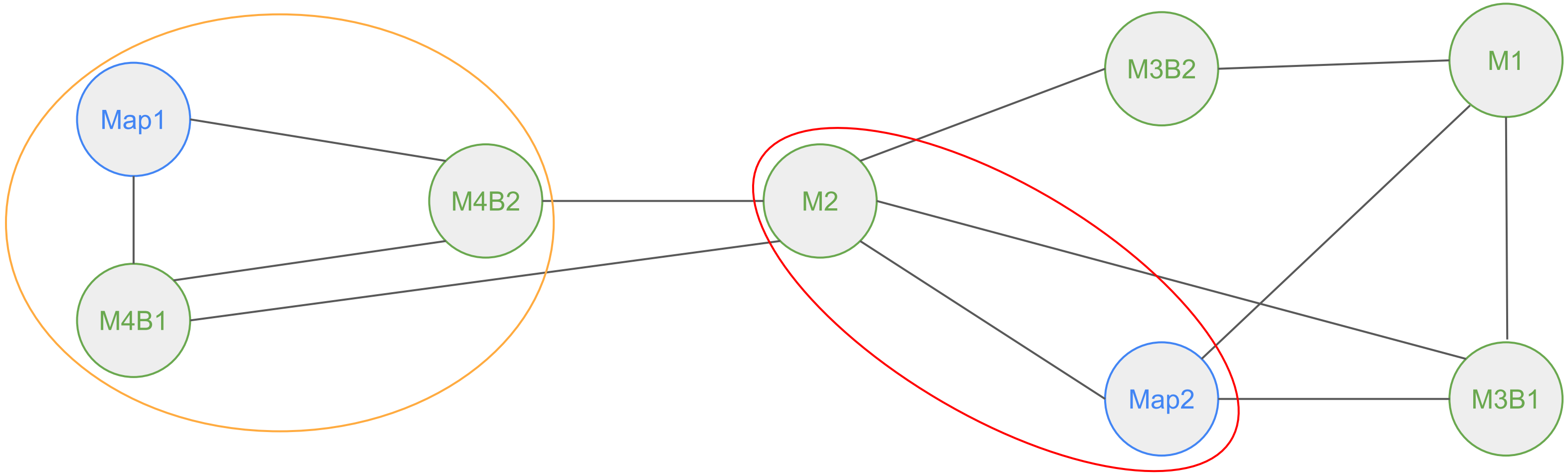
Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
M4B1-M2
M4B2-M2
M2-Map2
M2-M3B1
M2-M3B2
Map2-M3B1
Map2-M1
M3B1-M1
M3B2-M1



Building the Collision Graph - Finding layers

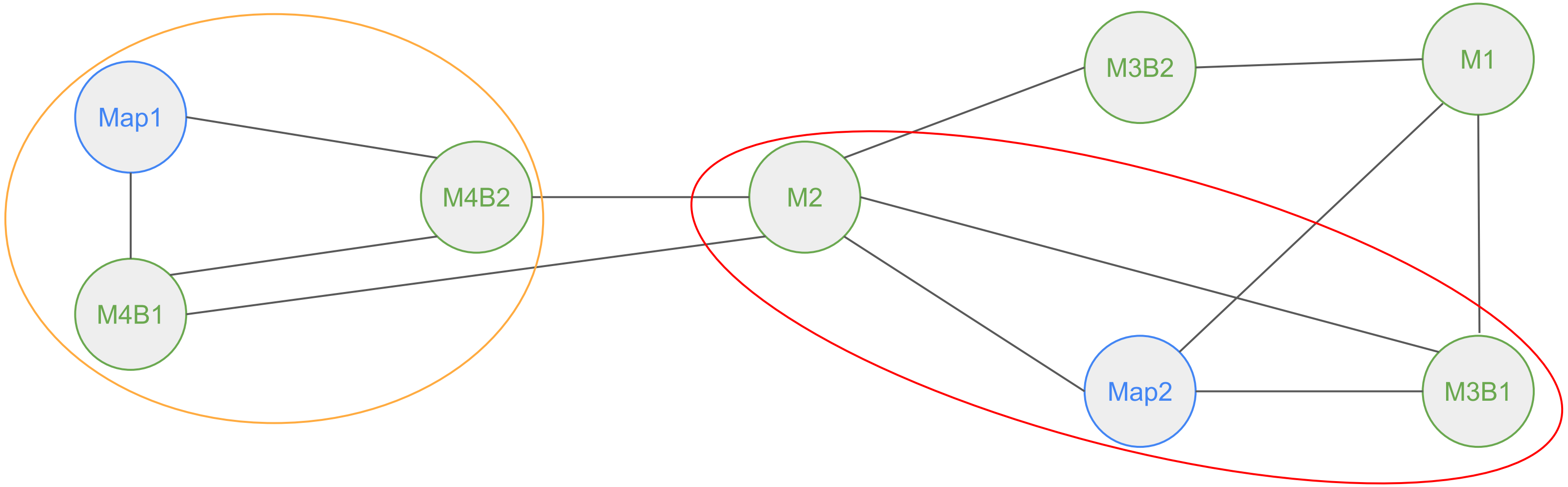
Choose another edge and repeat the process until the list is empty.

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
M4B1-M2
M4B2-M2
M2-Map2
M2-M3B1
M2-M3B2
Map2-M3B1
Map2-M1
M3B1-M1
M3B2-M1



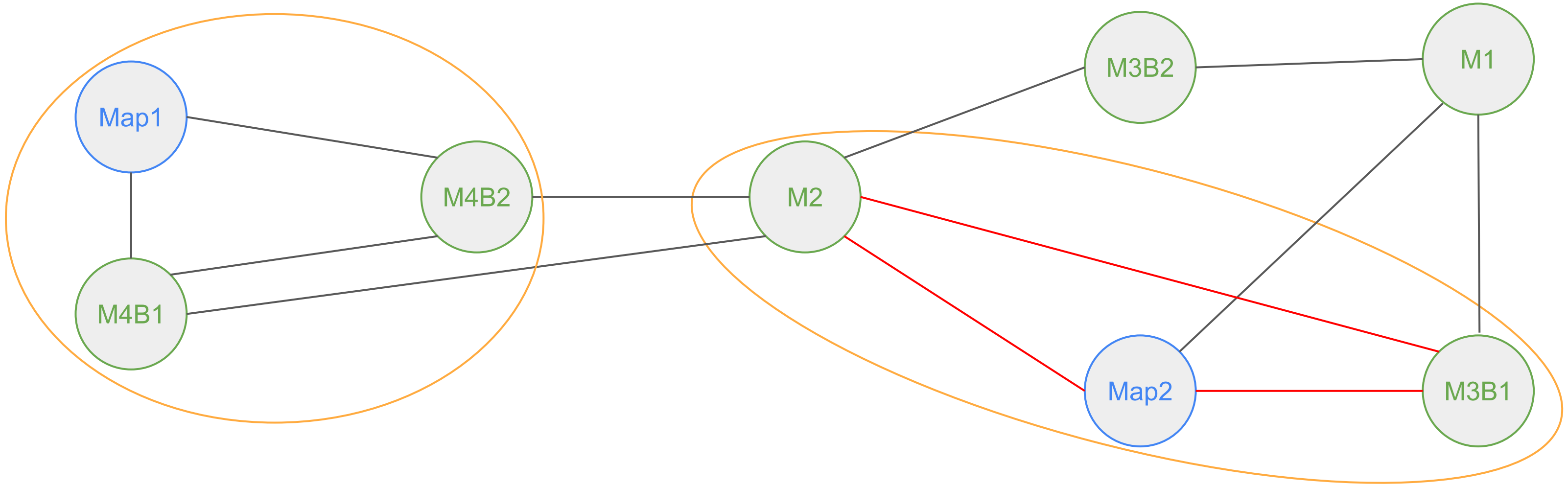
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
M4B1-M2
M4B2-M2
M2-Map2
M2-M3B1
M2-M3B2
Map2-M3B1
Map2-M1
M3B1-M1
M3B2-M1



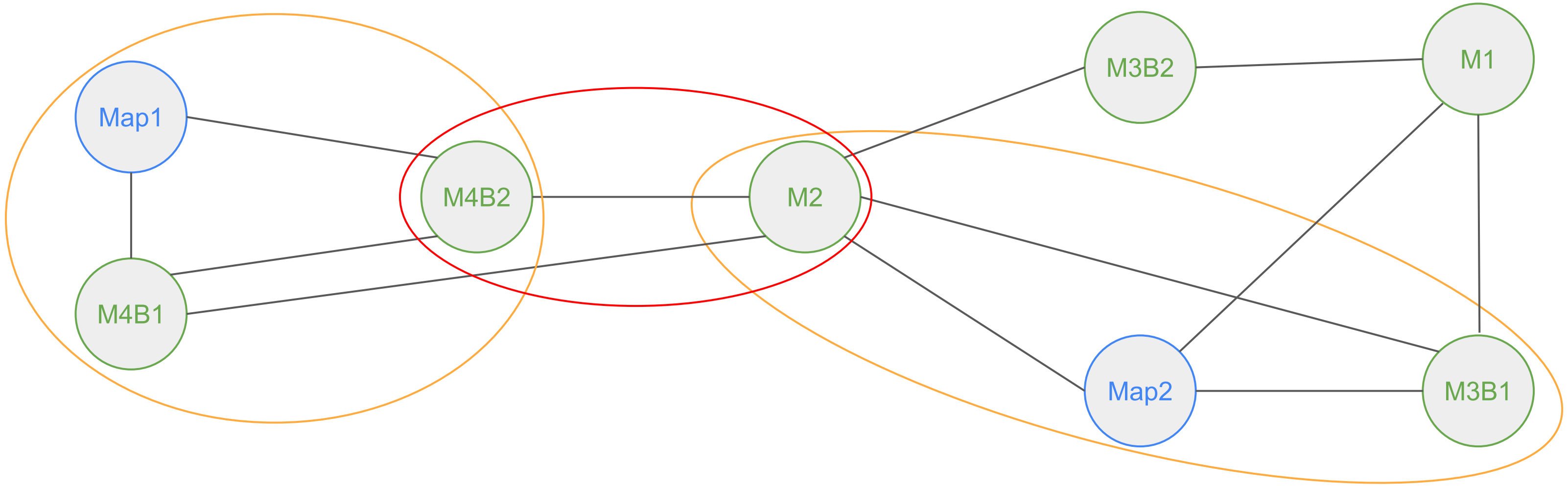
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
M4B1-M2
M4B2-M2
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
M3B2-M1



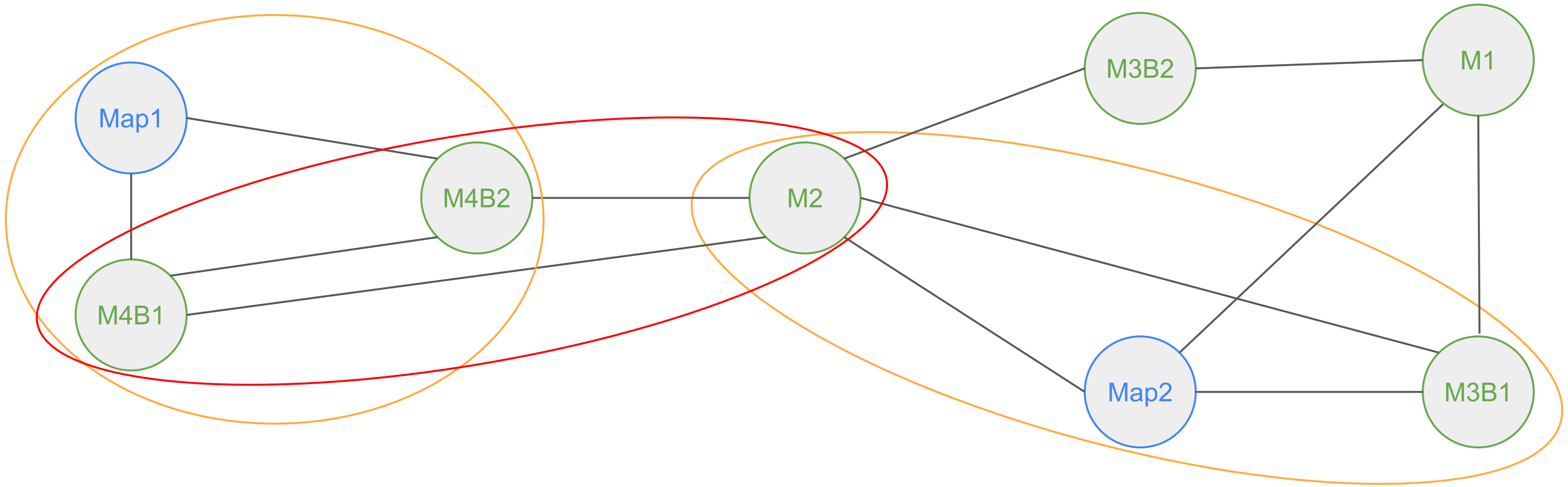
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
M4B1-M2
M4B2-M2
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
M3B2-M1



Building the Collision Graph - Finding layers

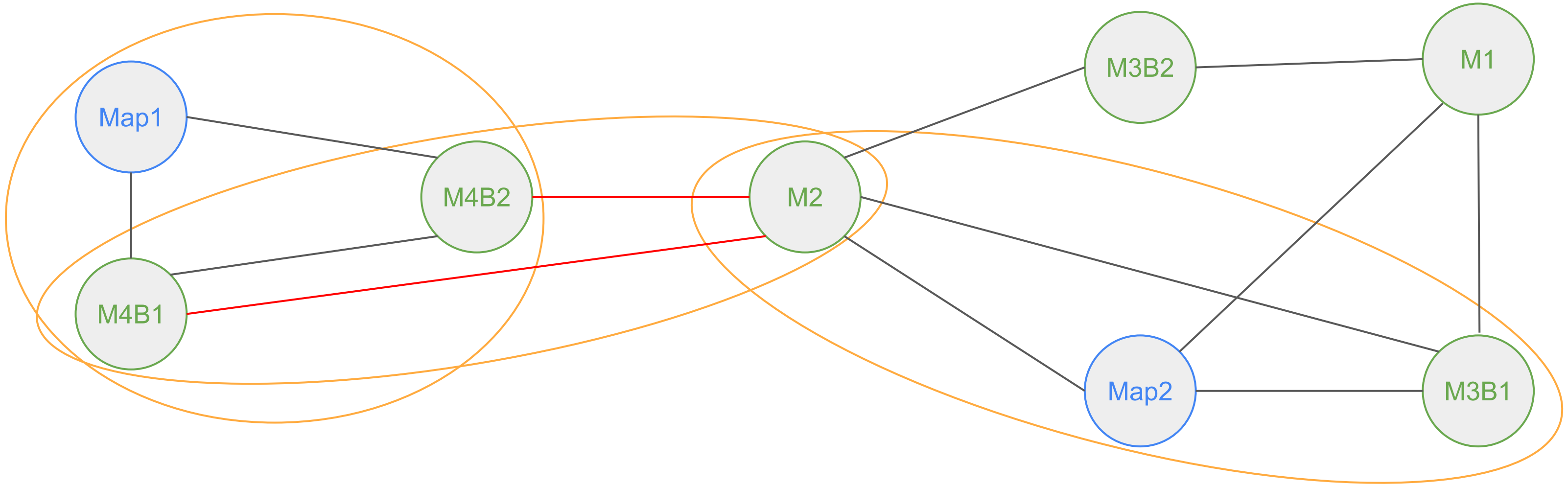
Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
M4B1-M2
M4B2-M2
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
M3B2-M1



Building the Collision Graph - Finding layers

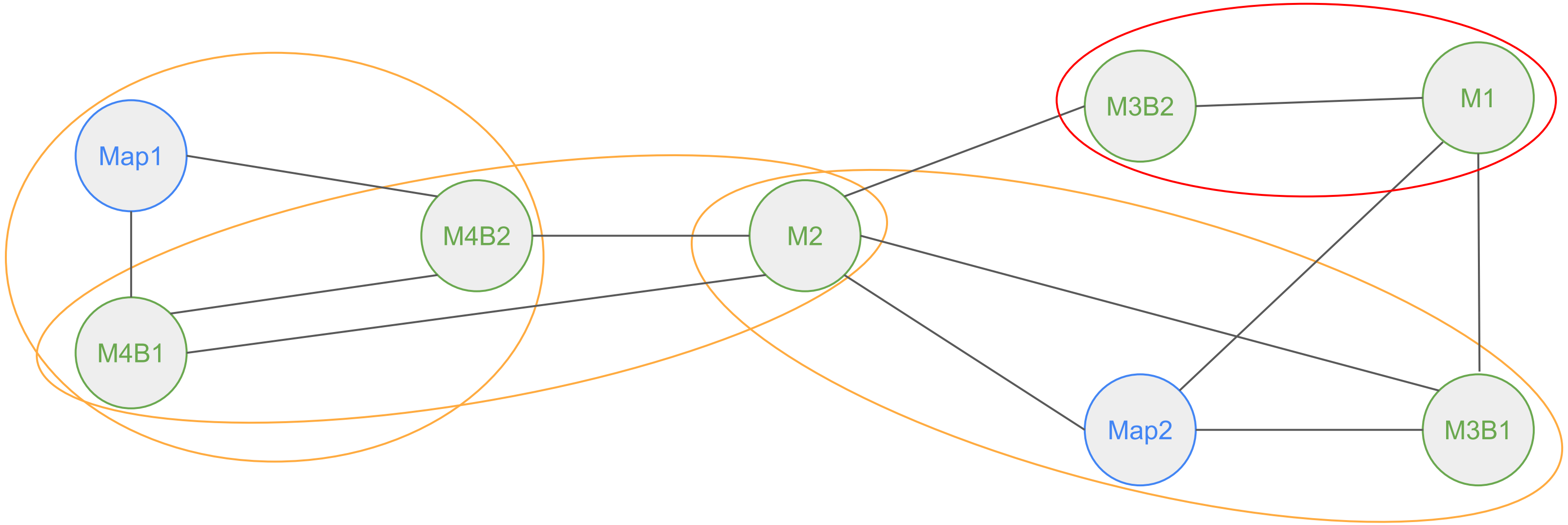
NOTE: Edges that were already removed from the list can still be used on new subgraphs. Example: M4B1-M4B2.

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
M3B2-M1



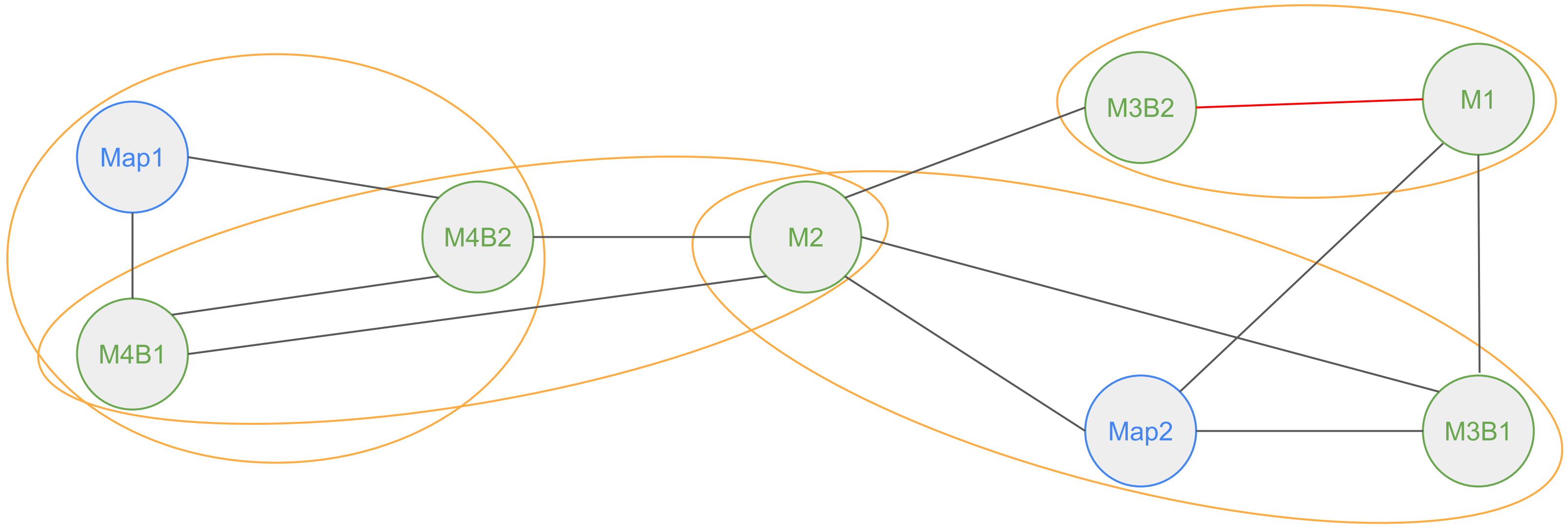
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
M3B2-M1



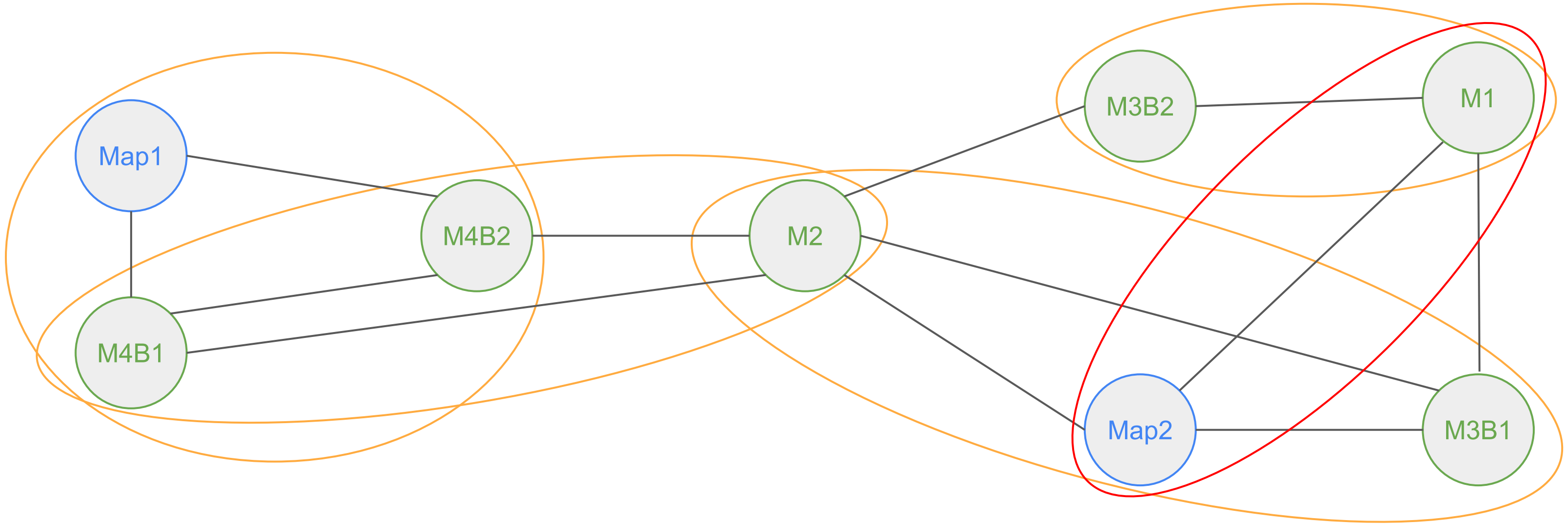
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
~~M3B2-M1~~



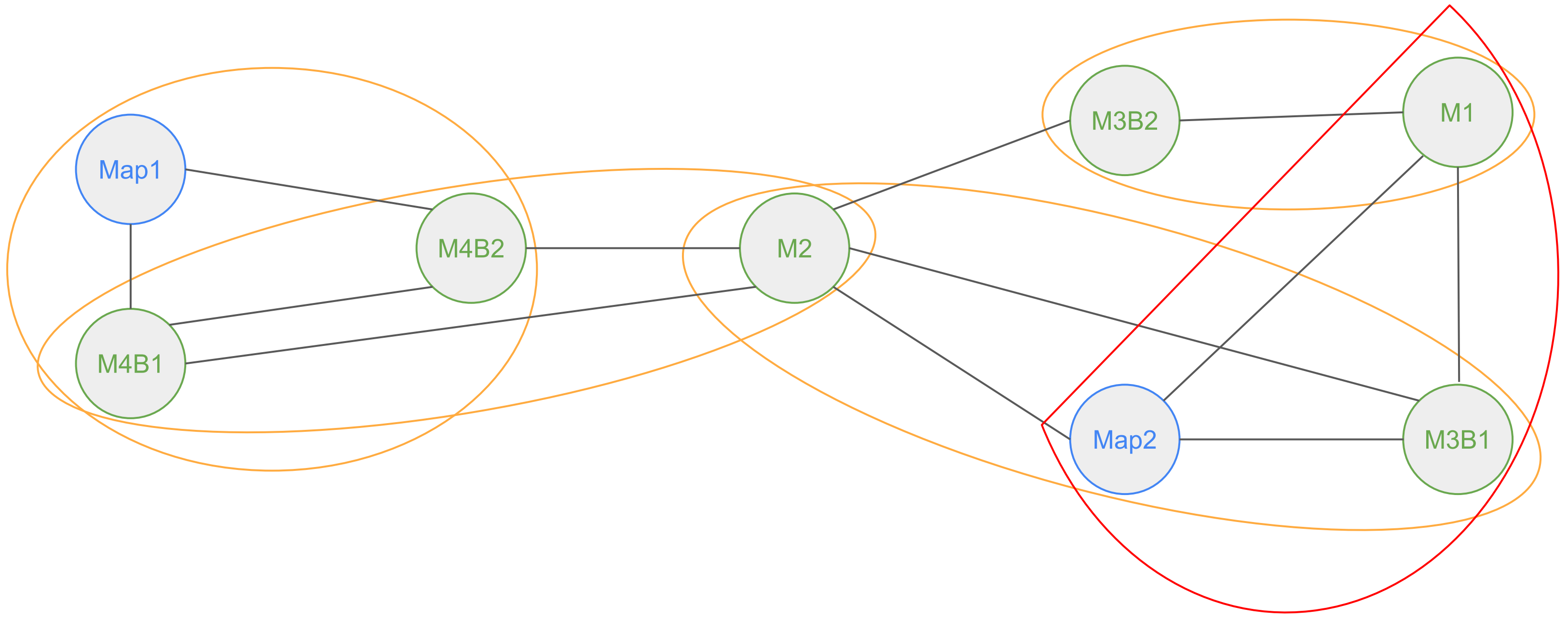
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
~~M3B2-M1~~



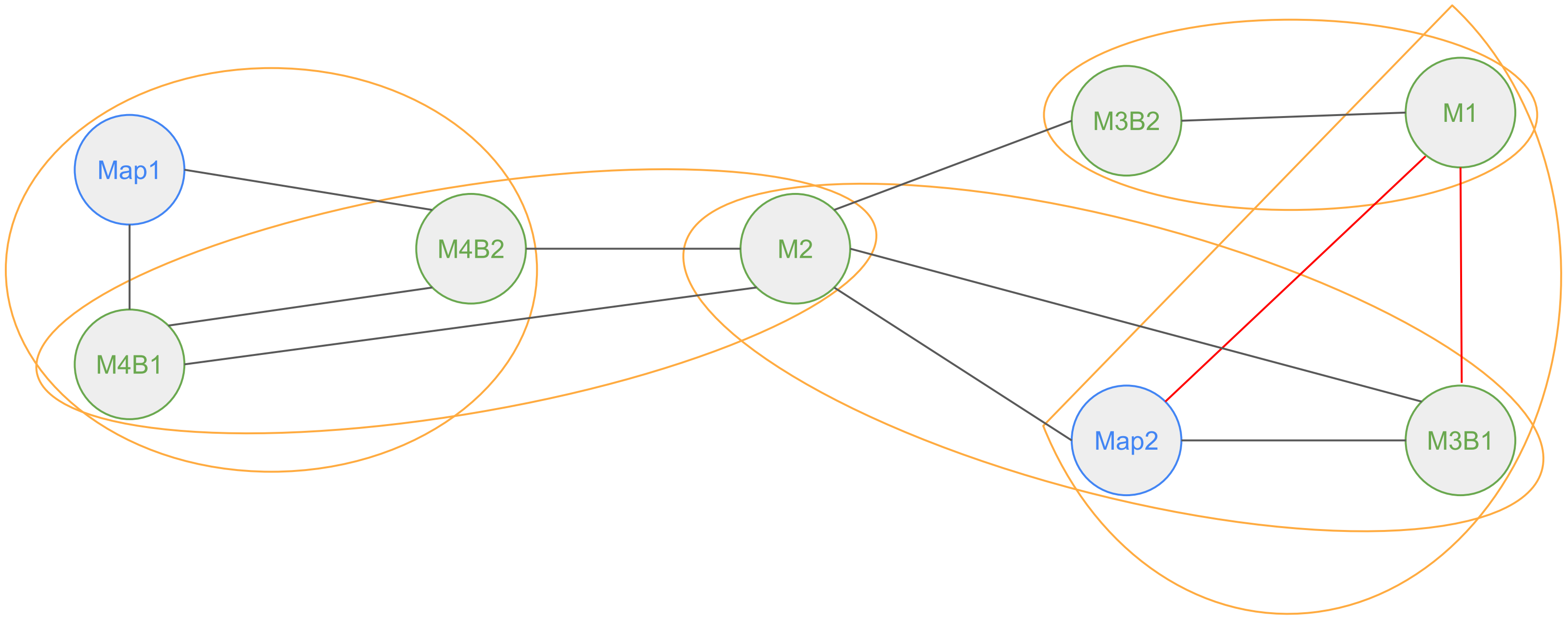
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
Map2-M1
M3B1-M1
~~M3B2-M1~~



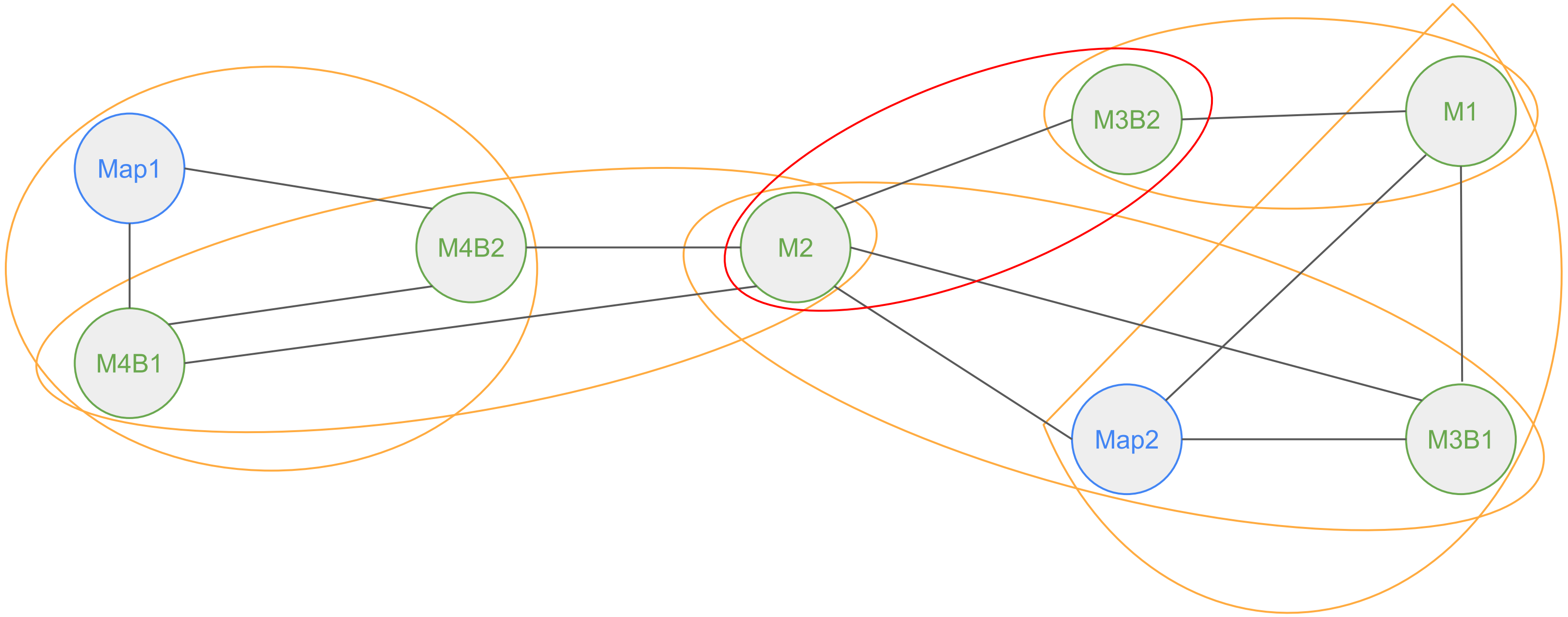
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
~~Map2-M1~~
~~M3B1-M1~~
~~M3B2-M1~~



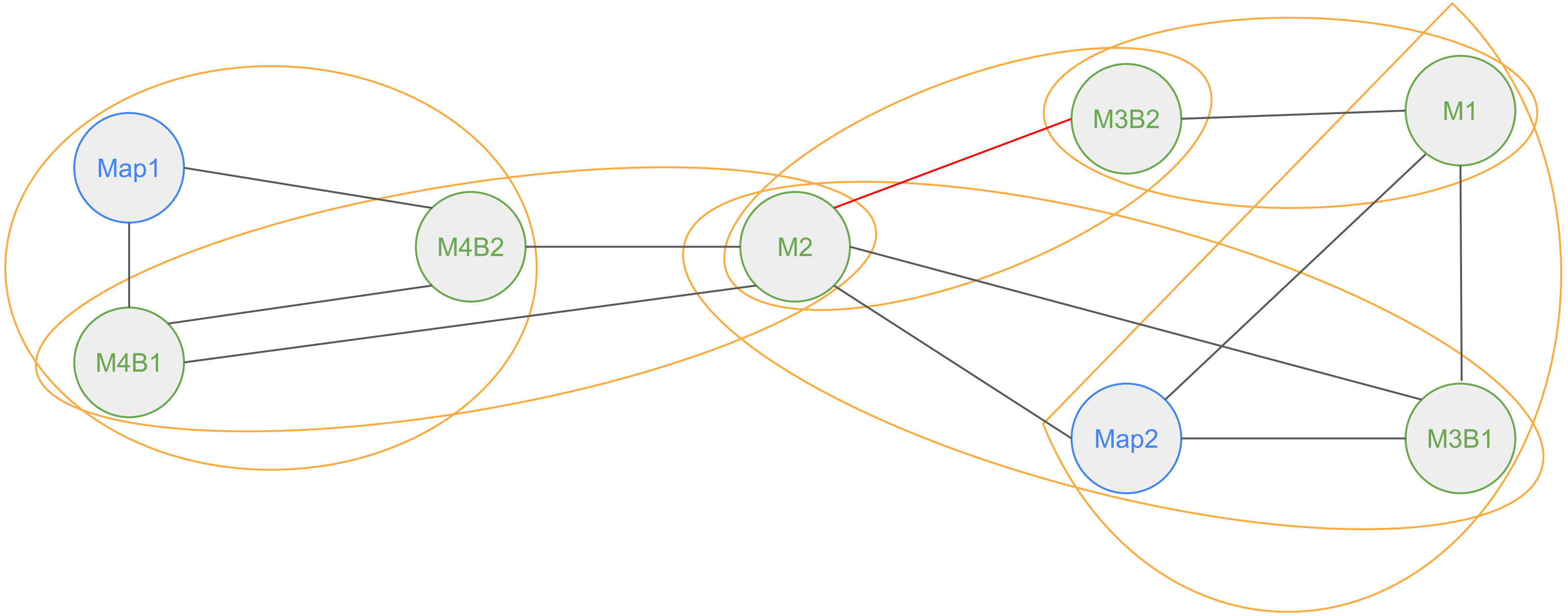
Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
M2-M3B2
~~Map2-M3B1~~
~~Map2-M1~~
~~M3B1-M1~~
~~M3B2-M1~~

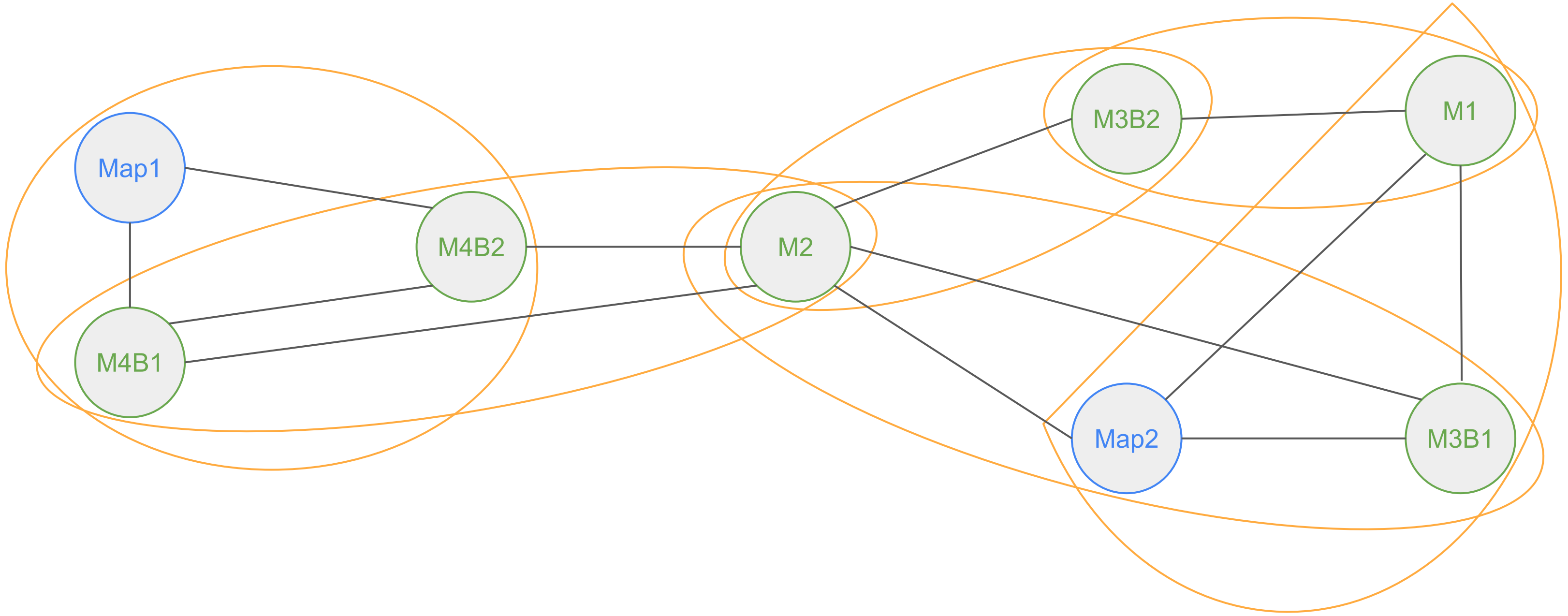


Building the Collision Graph - Finding layers

Edges:
~~Map1-M4B1~~
~~Map1-M4B2~~
~~M4B1-M4B2~~
~~M4B1-M2~~
~~M4B2-M2~~
~~M2-Map2~~
~~M2-M3B1~~
~~M2-M3B2~~
~~Map2-M3B1~~
~~Map2-M1~~
~~M3B1-M1~~
~~M3B2-M1~~

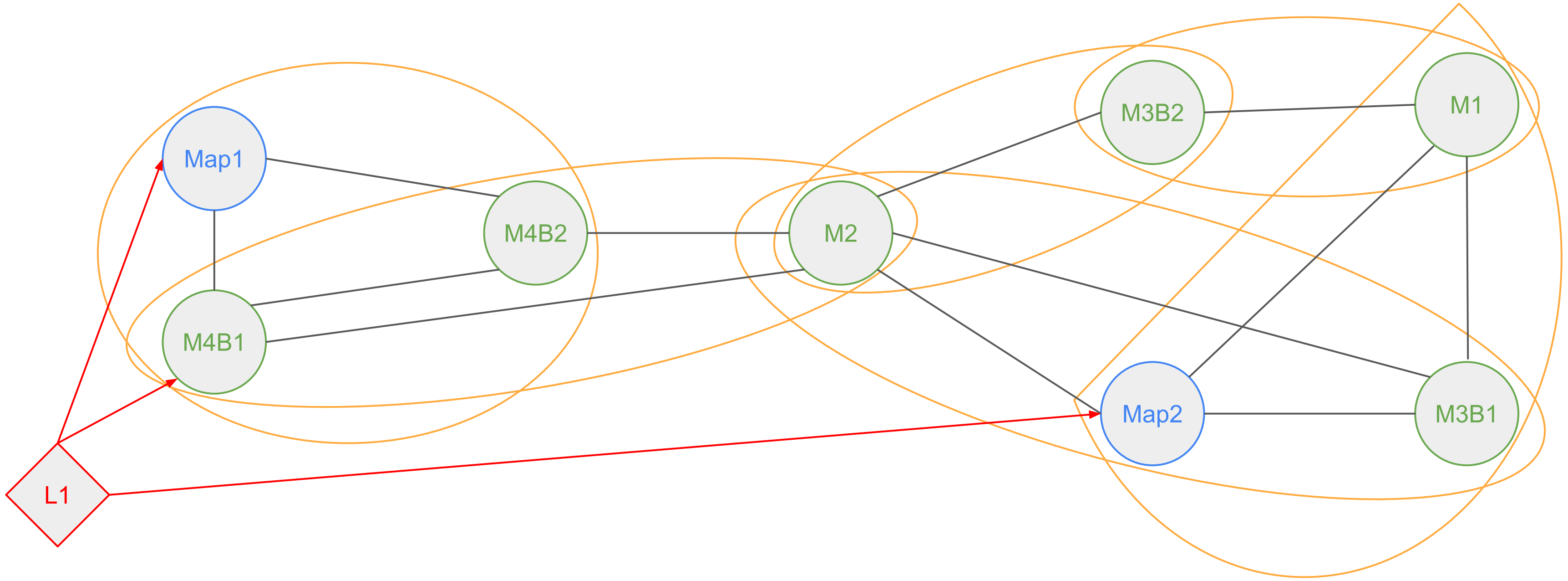


Final Collision Graph



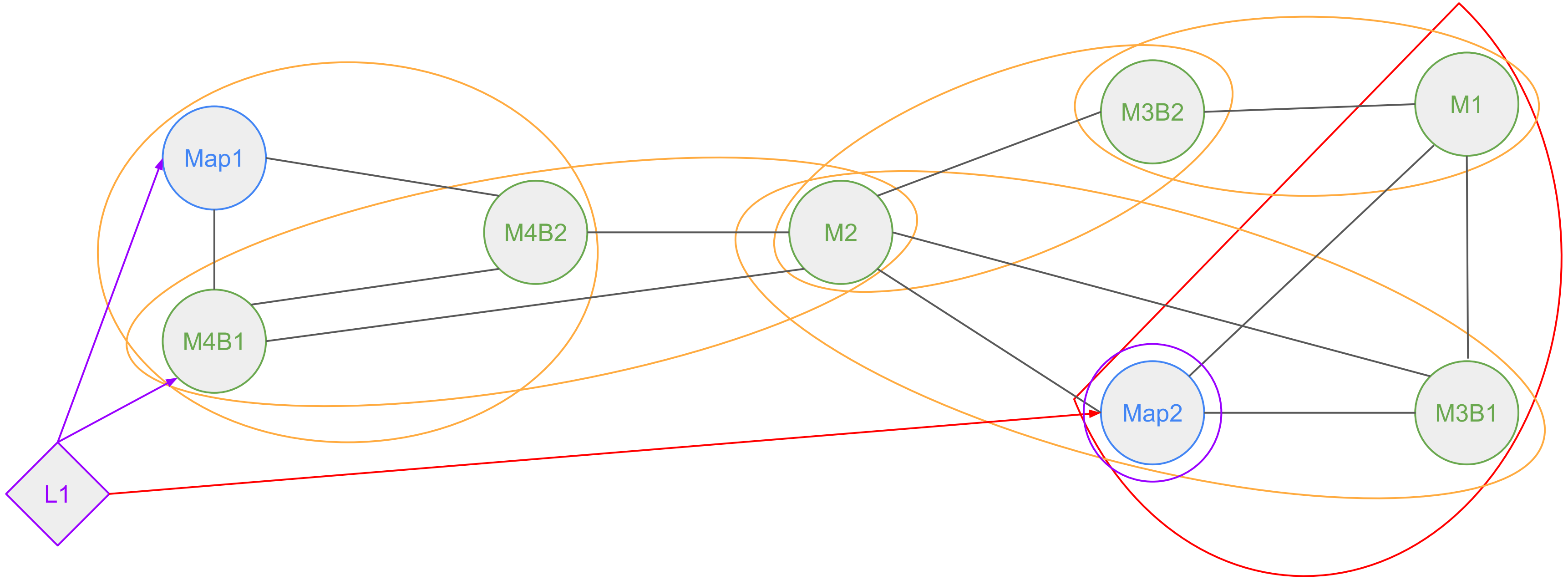
Adding Lasers to the Graph

Start adding lasers one at a time and connecting them to all nodes they must detect.

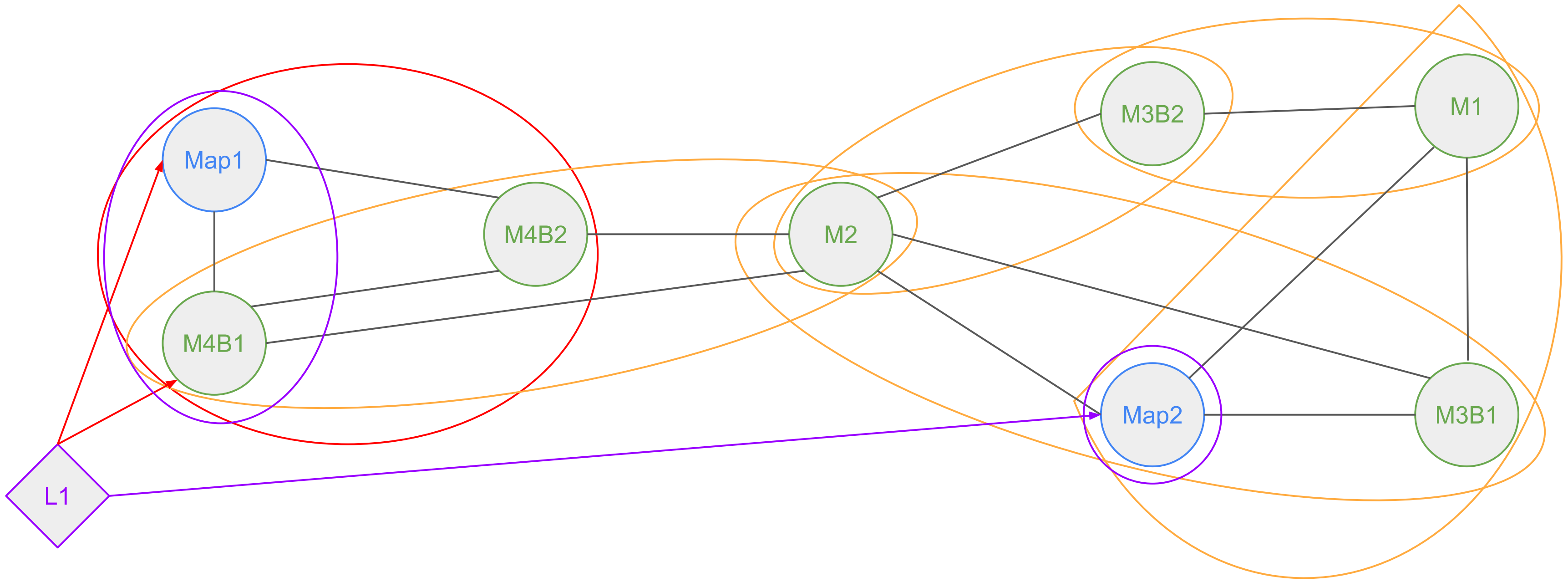


Adding Lasers to the Graph

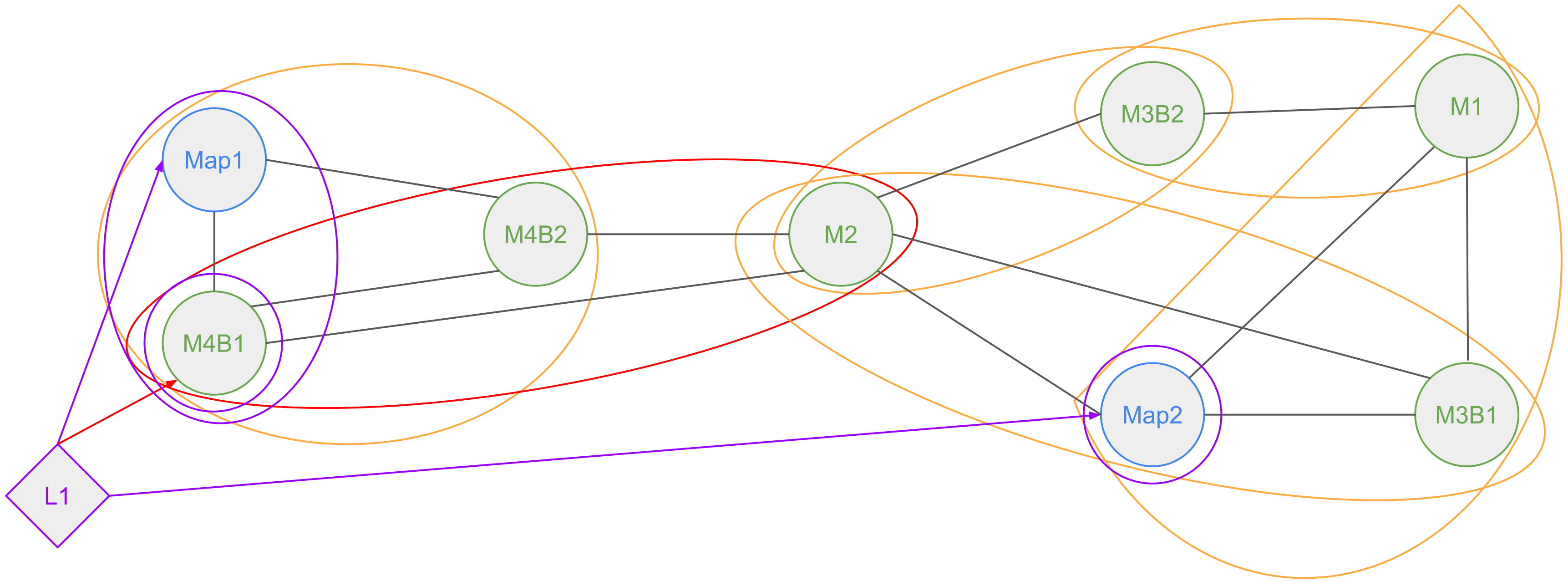
For each collision layer the laser interacts with, group the vertices inside it that are detected.



Adding Lasers to the Graph

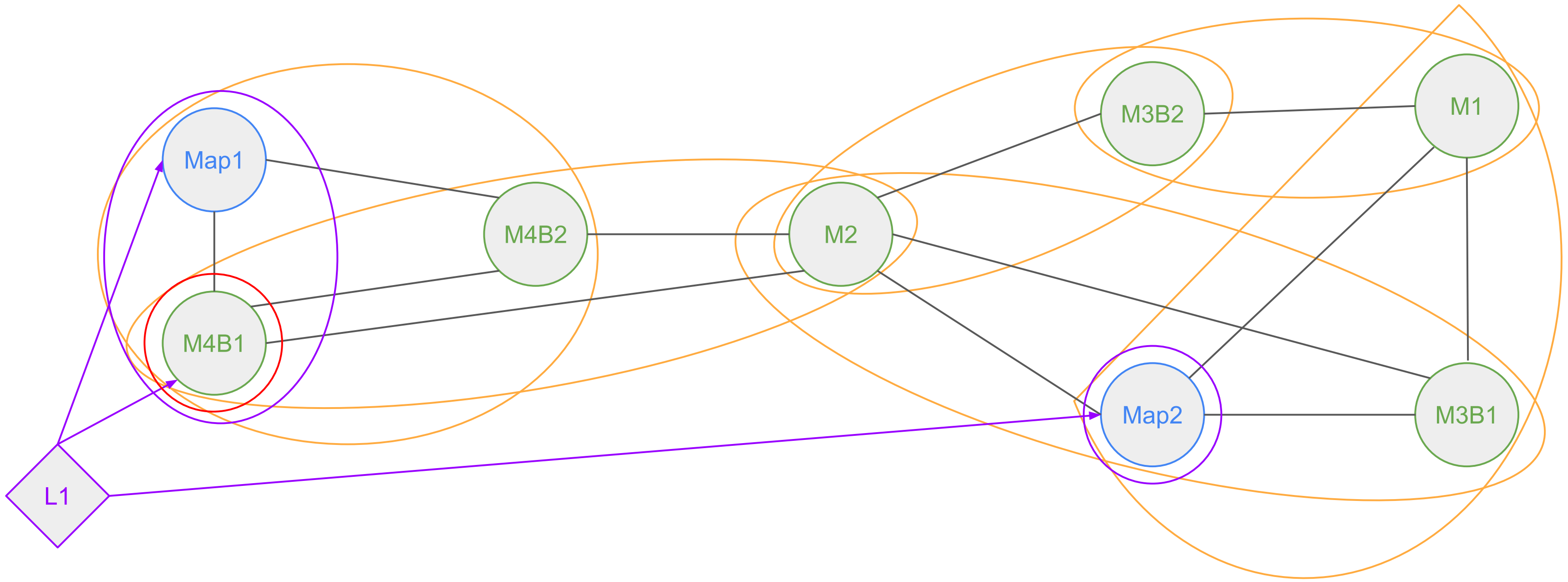


Adding Lasers to the Graph



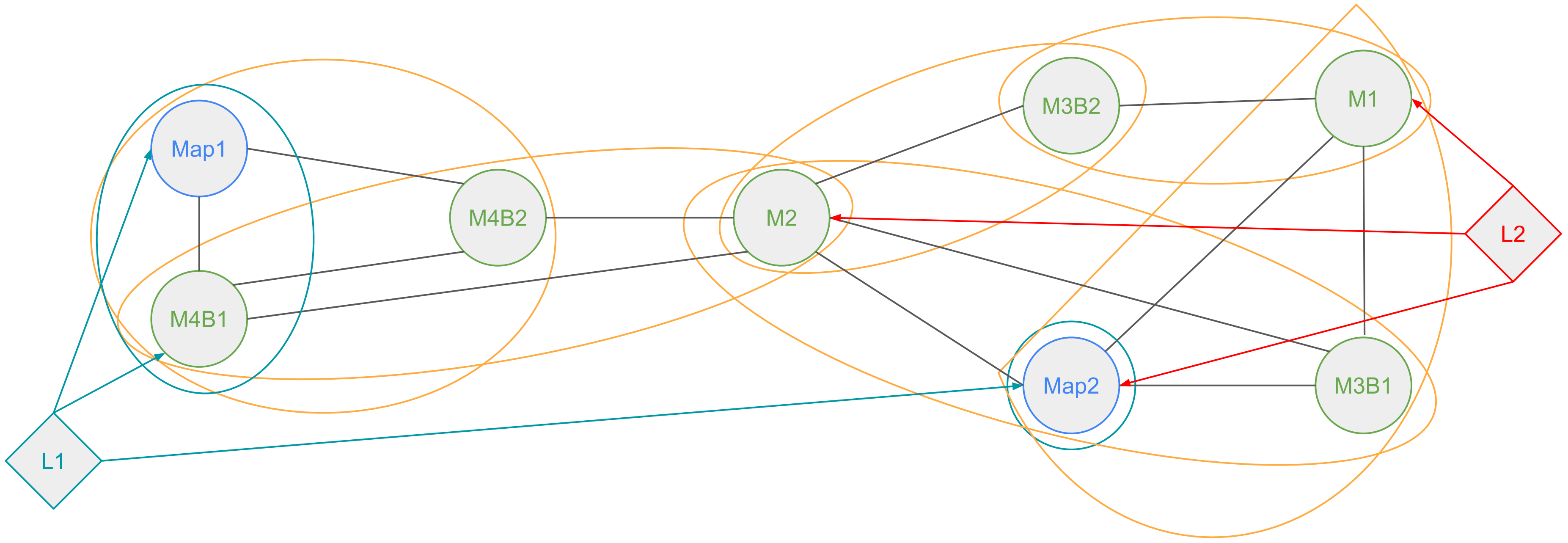
Adding Lasers to the Graph

After checking all layers, if all the vertices of one of the new groups created is contained inside another of the new ones, it can be removed.

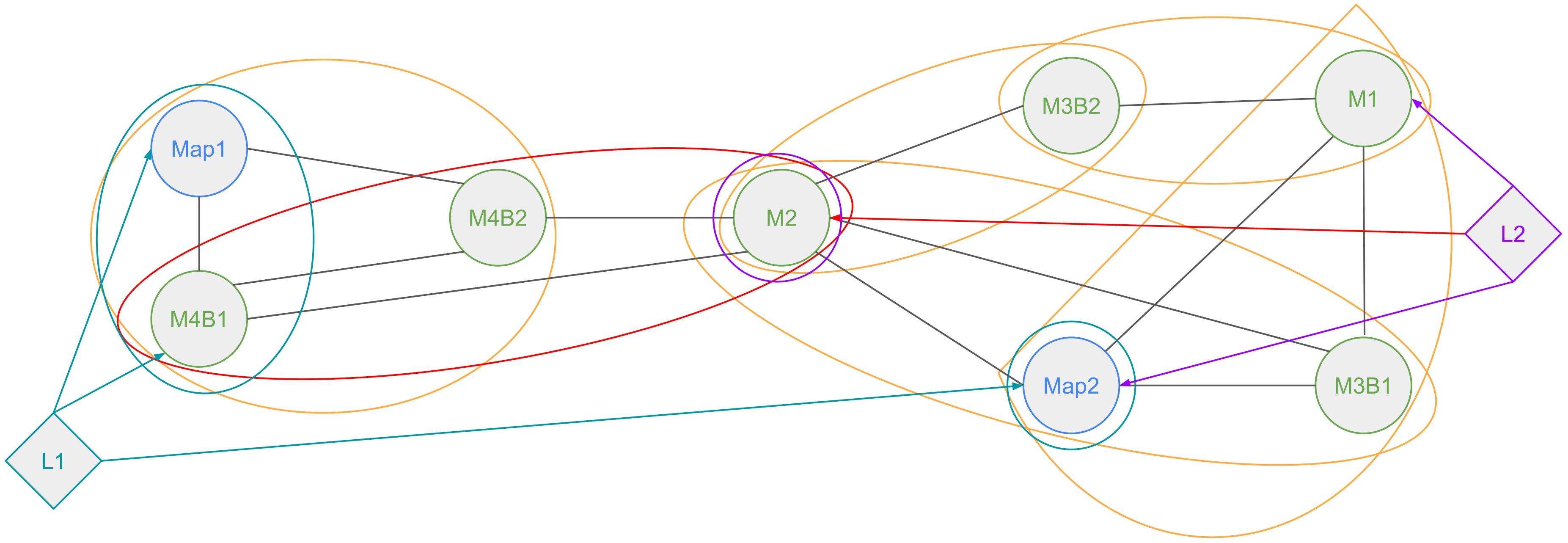


Adding Lasers to the Graph

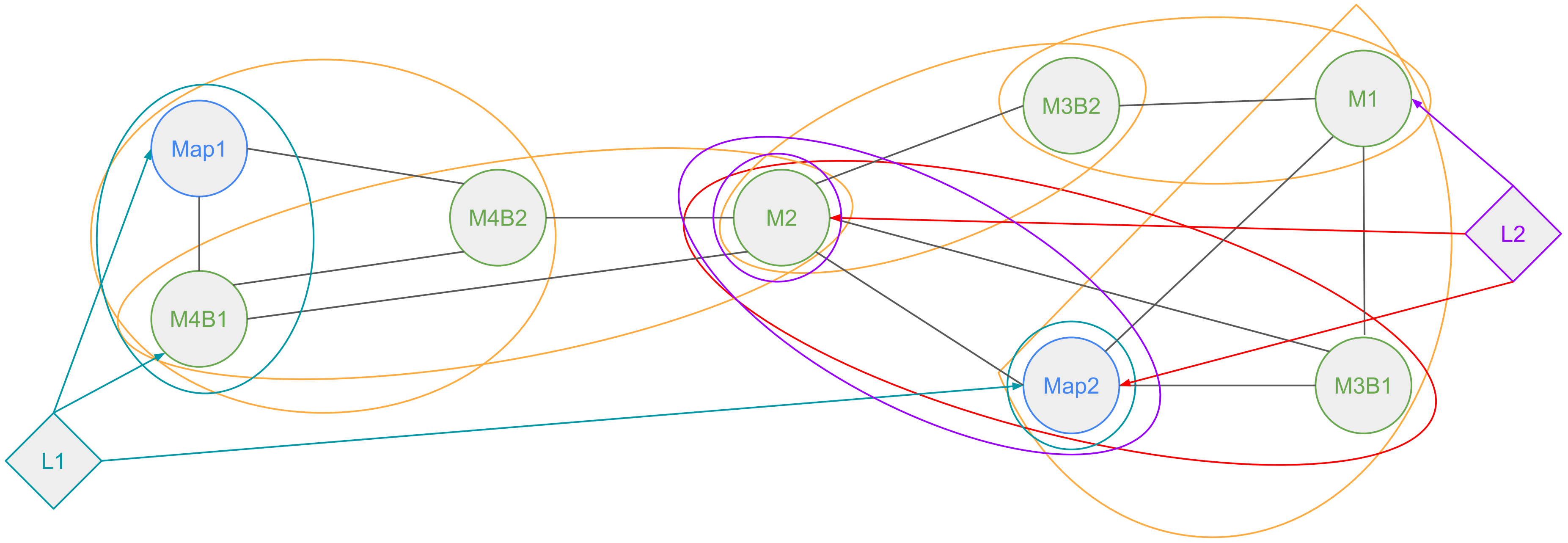
Repeat the same process for all the lasers.



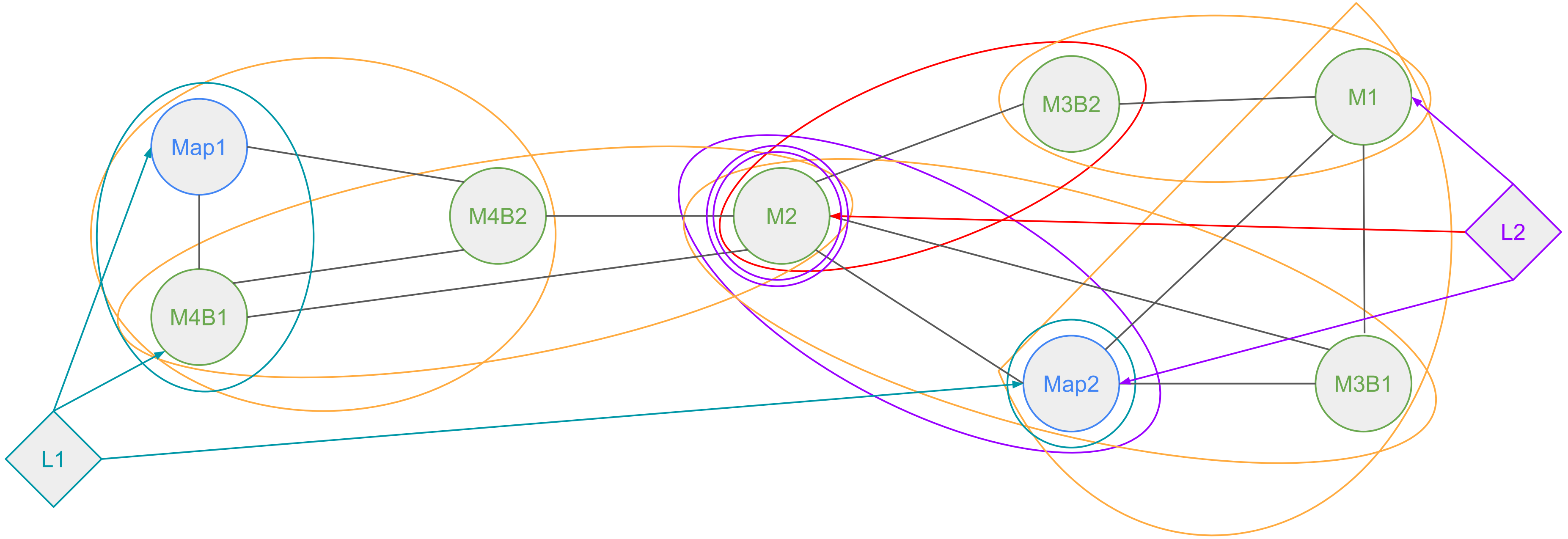
Adding Lasers to the Graph



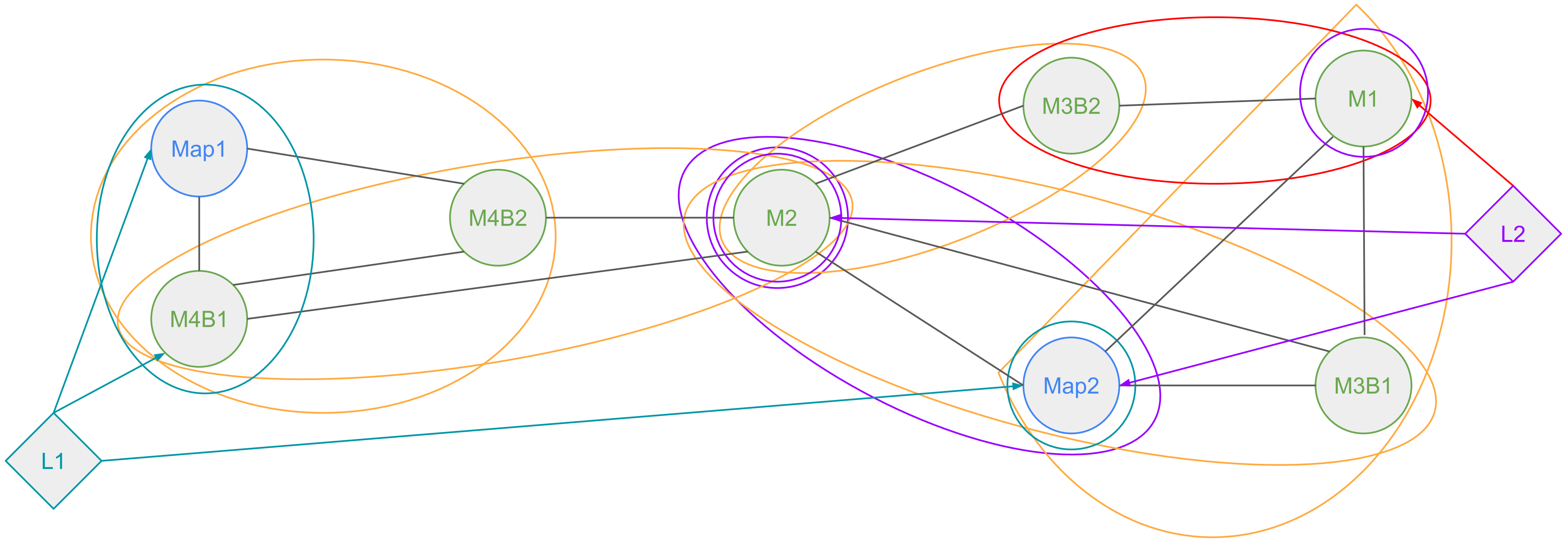
Adding Lasers to the Graph



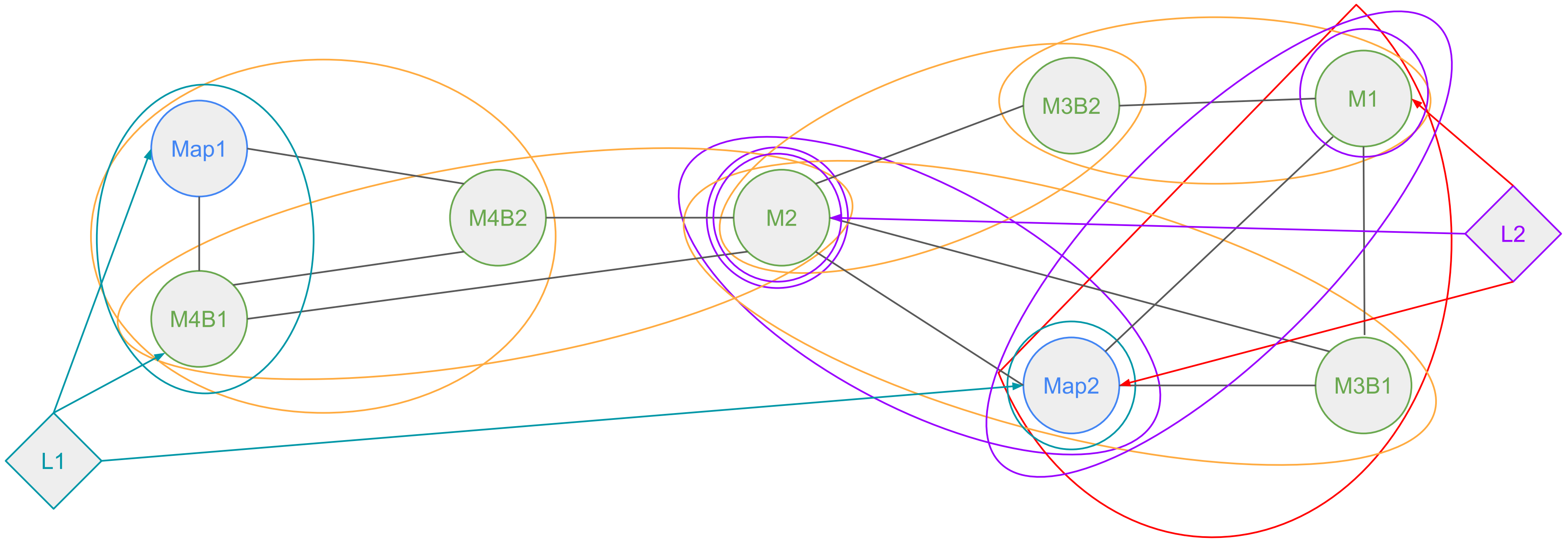
Adding Lasers to the Graph



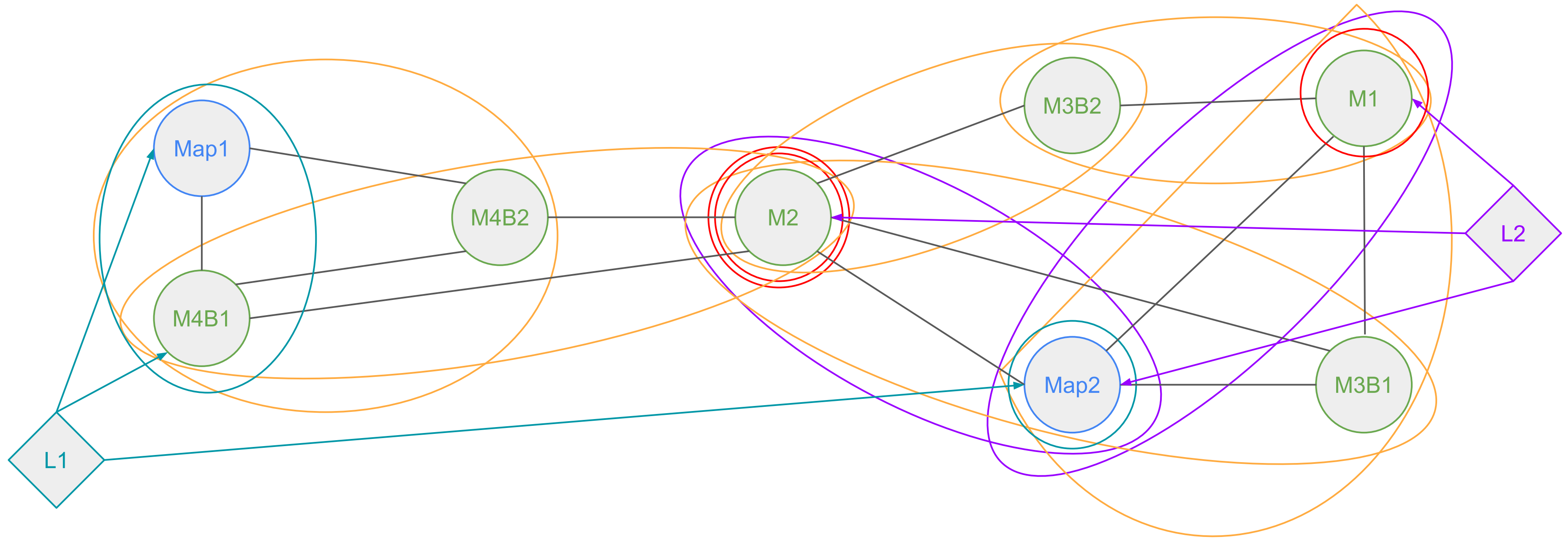
Adding Lasers to the Graph



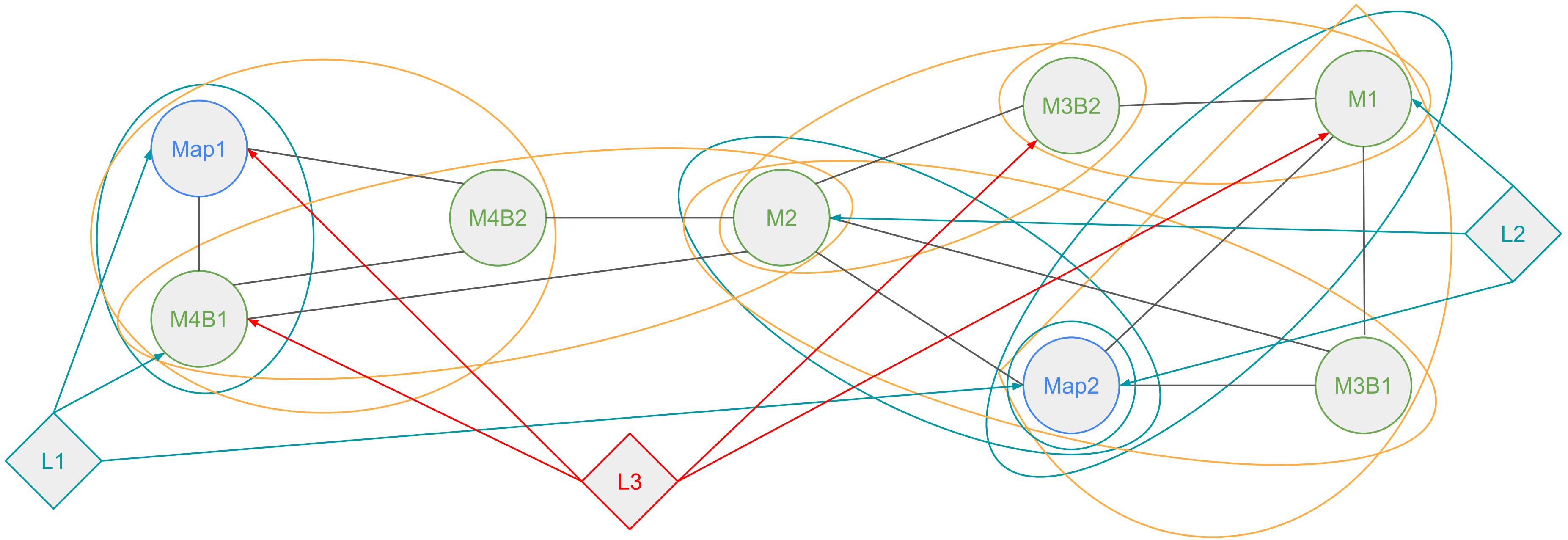
Adding Lasers to the Graph



Adding Lasers to the Graph

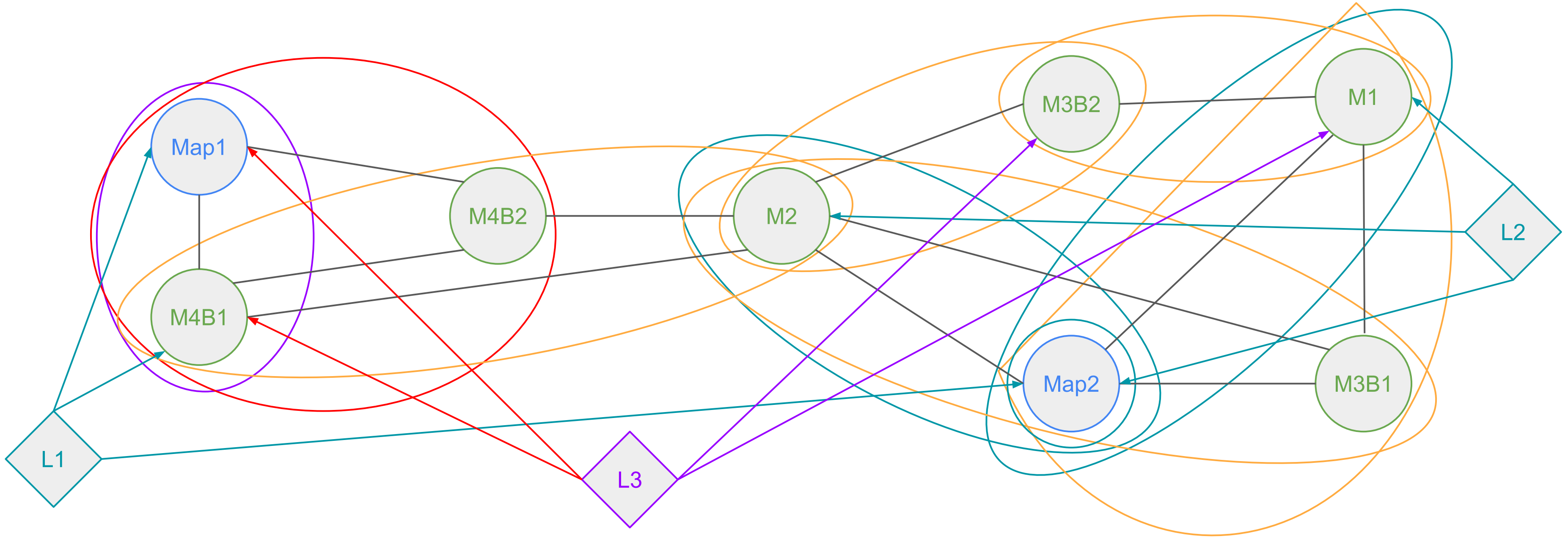


Adding Lasers to the Graph

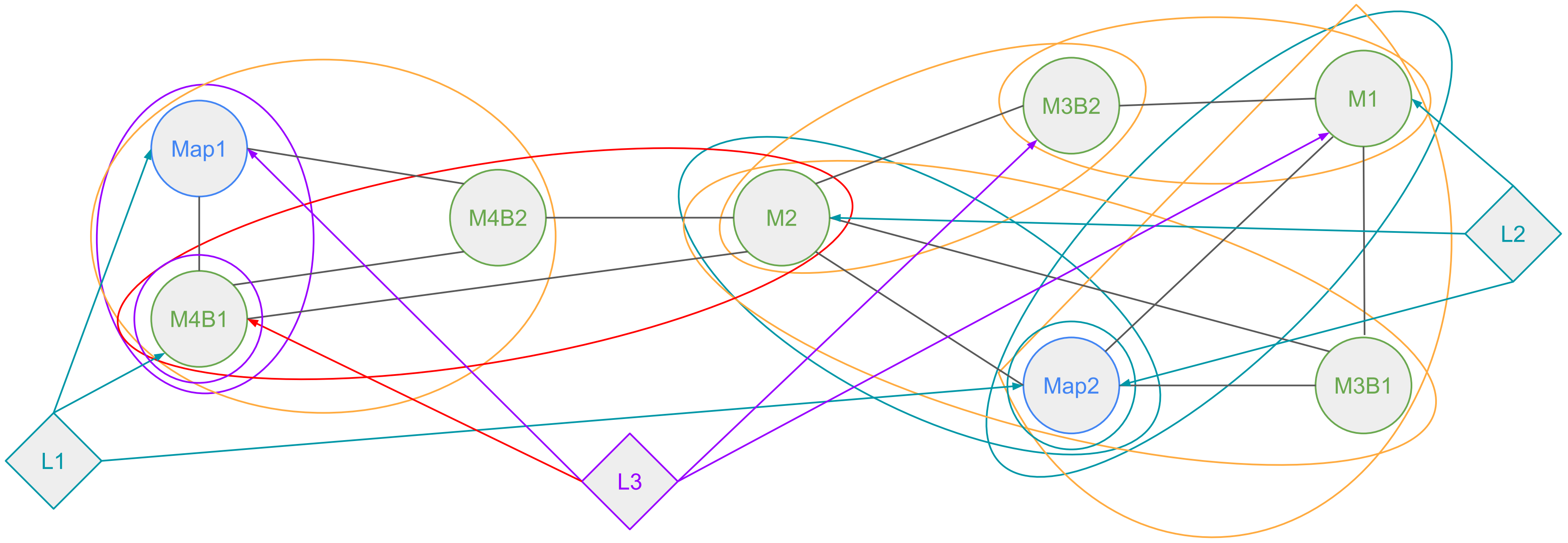


Adding Lasers to the Graph

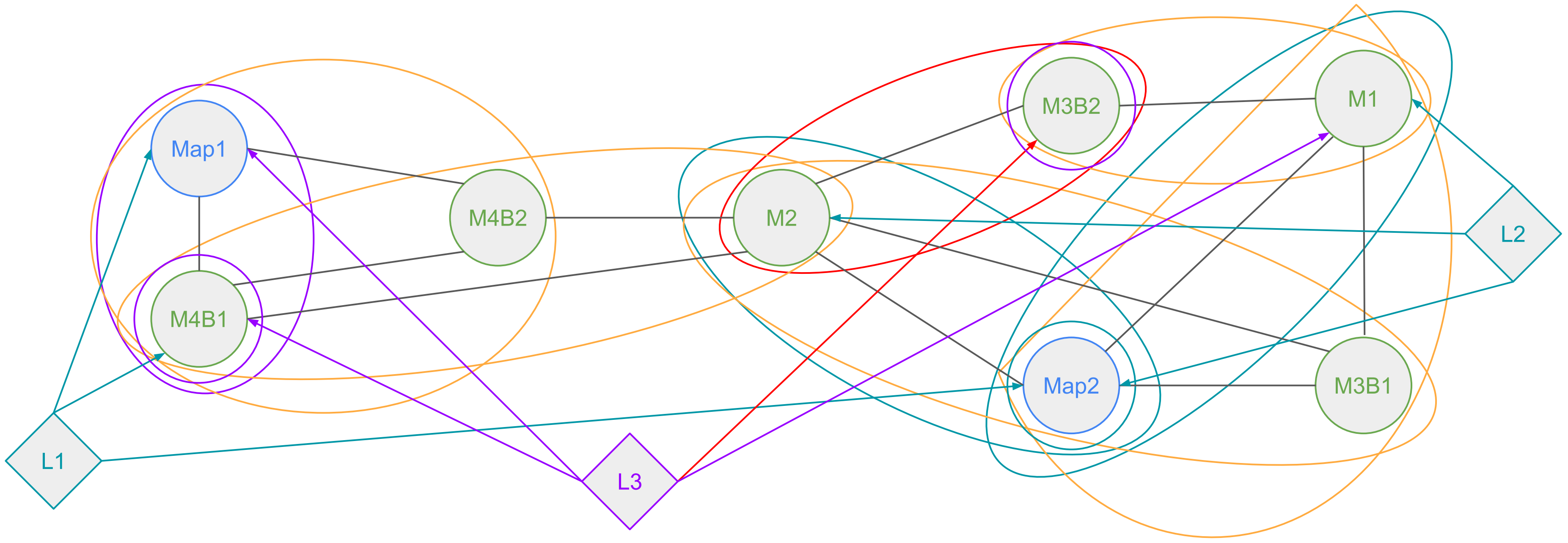
If at any point you get a group that already exist from a previous laser, it can be reused.



Adding Lasers to the Graph

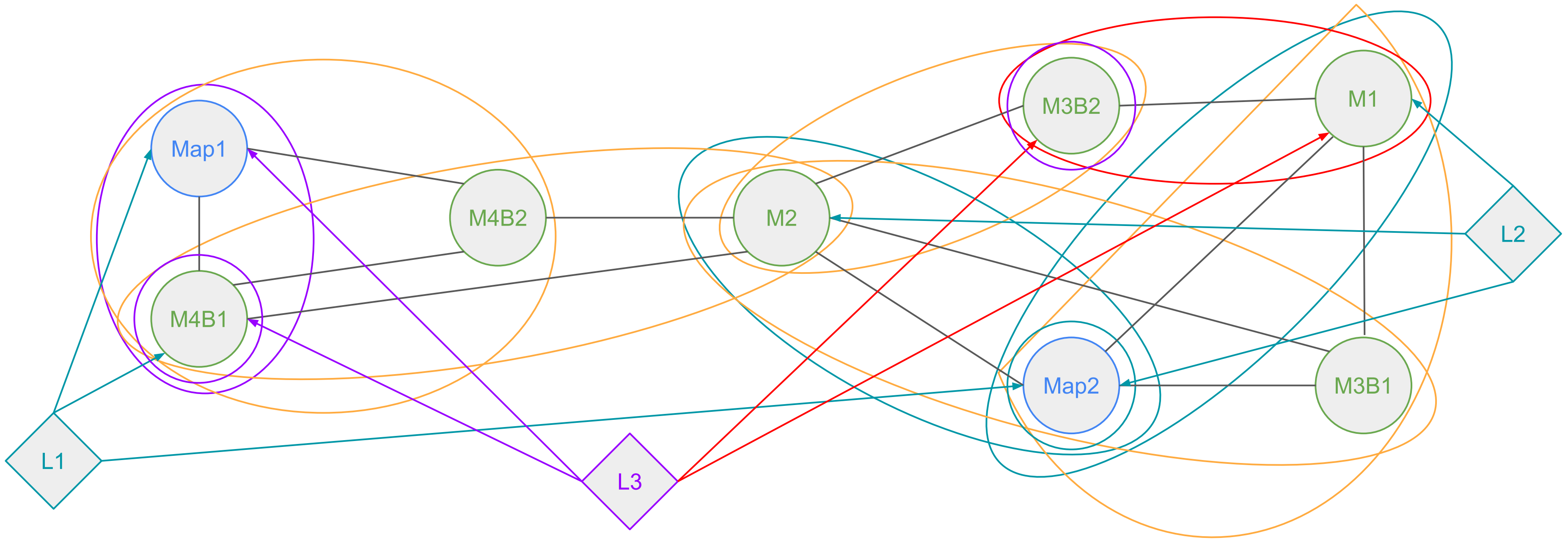


Adding Lasers to the Graph

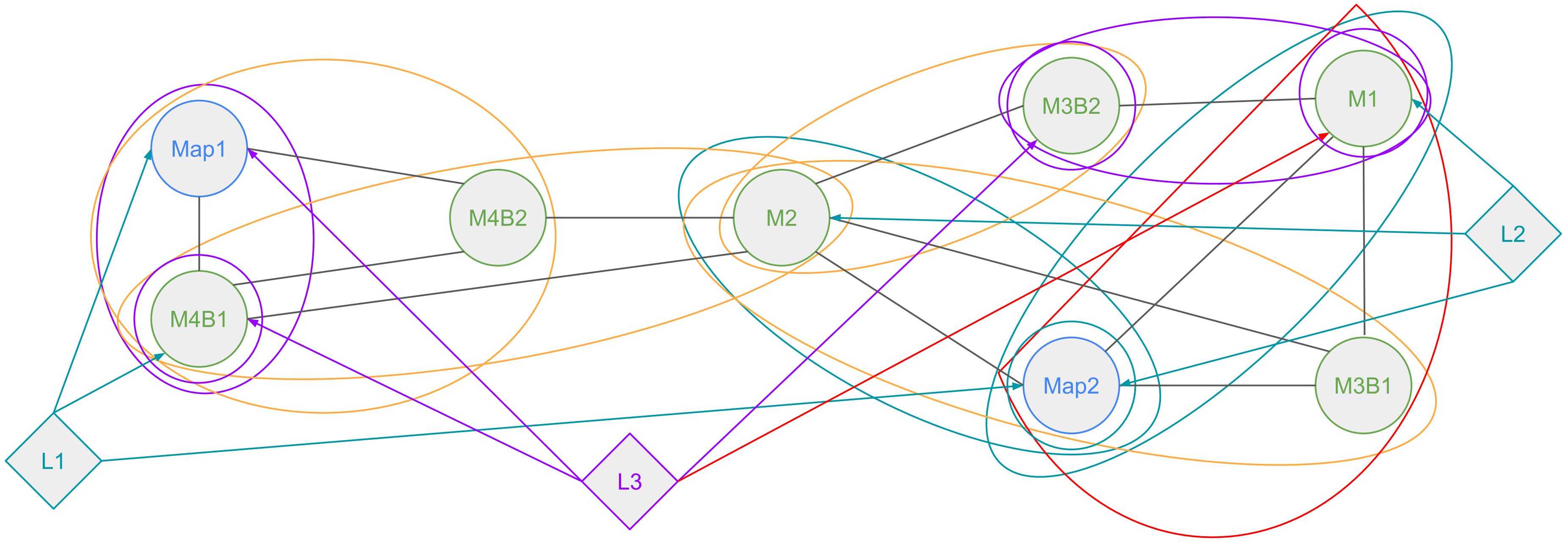


Adding Lasers to the Graph

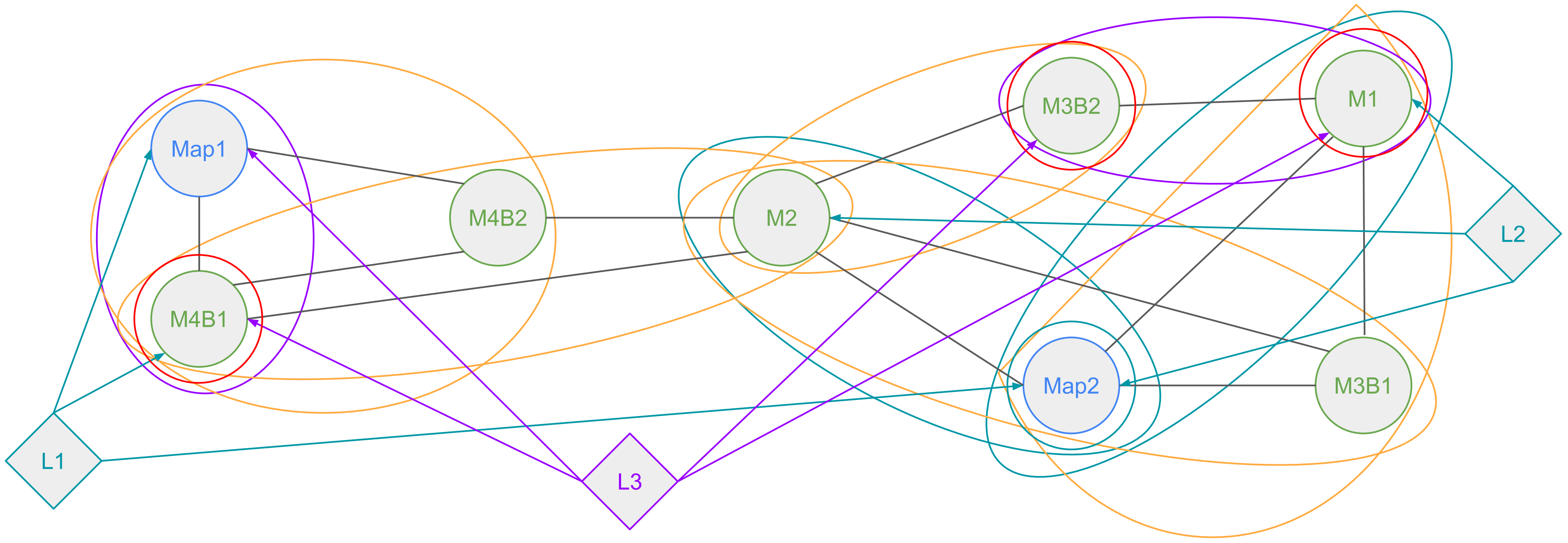
Collision groups can also be reused if all its elements need to be detected.



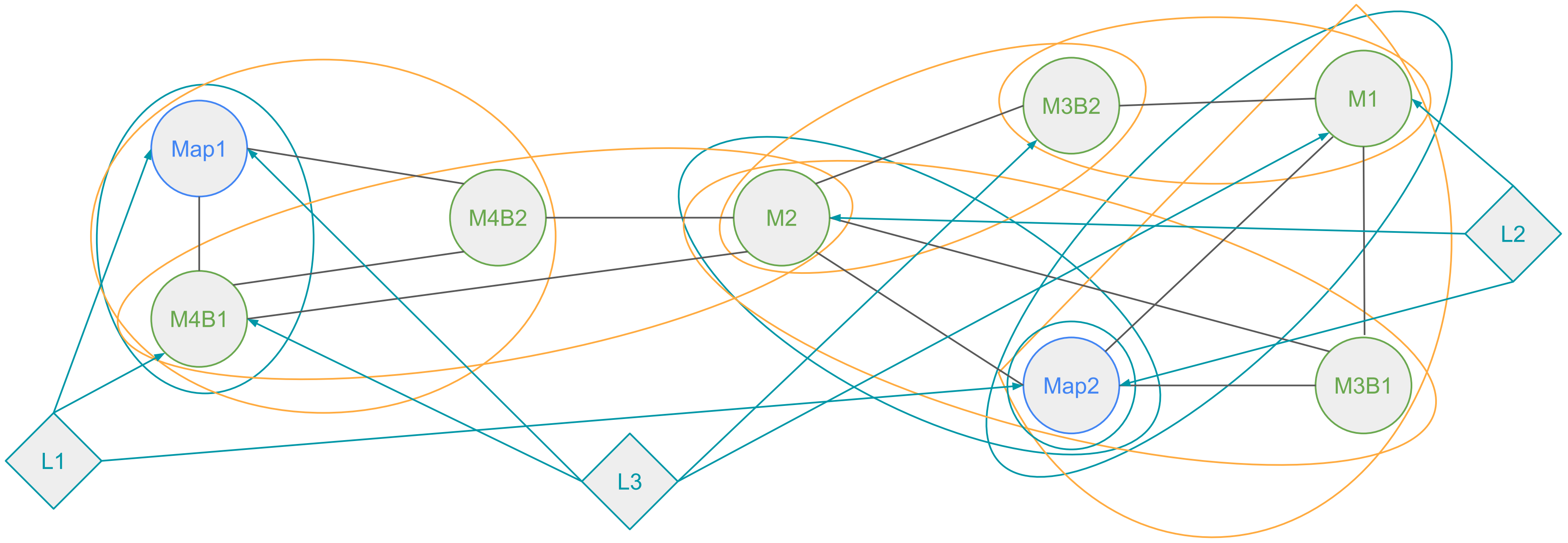
Adding Lasers to the Graph



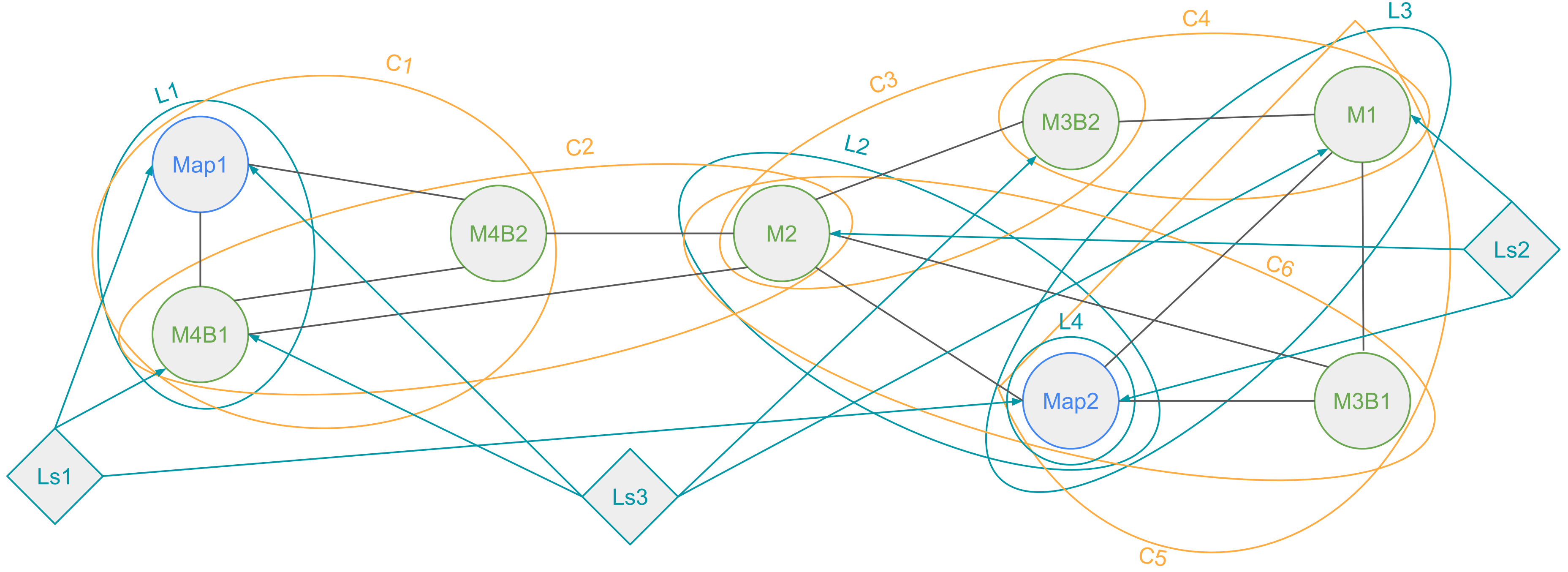
Adding Lasers to the Graph



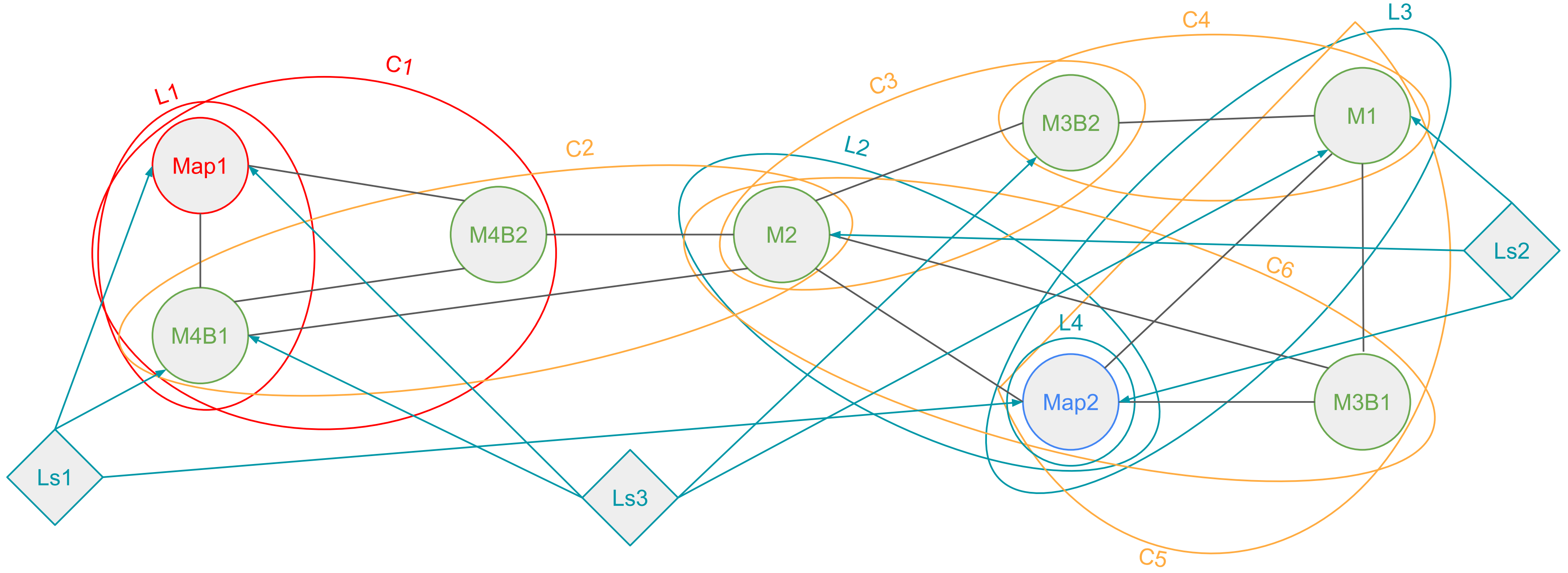
Final Graph



Label the layers

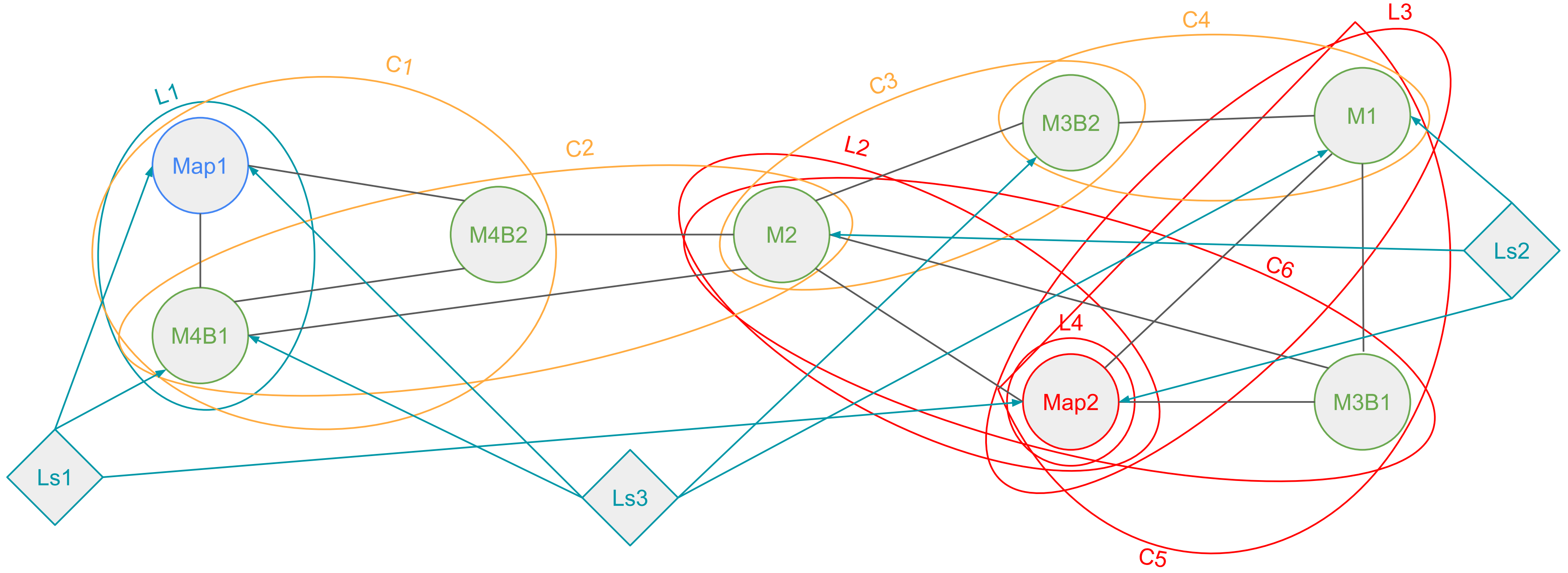


Add layers to the World file - Map1



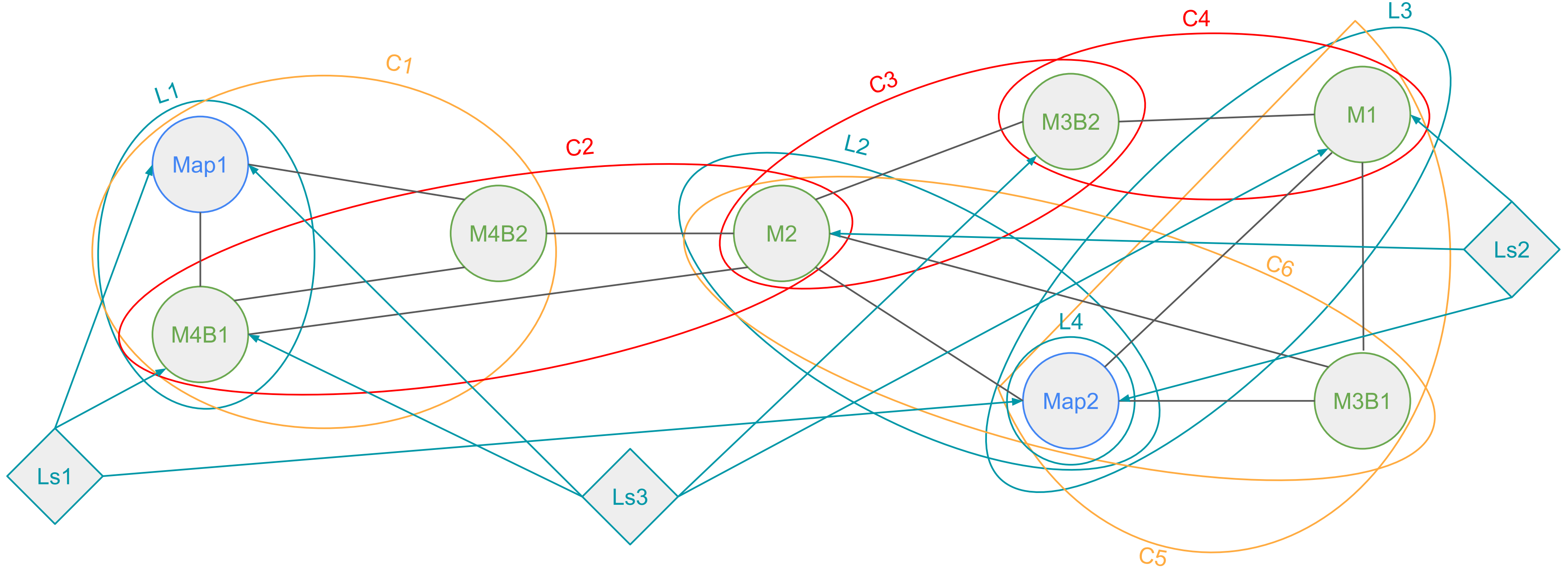
```
# map 1, add the layers that contain the node
- name: ["C1", "L1"]
  map: <image or dat file>
```

Add layers to the World file - Map2



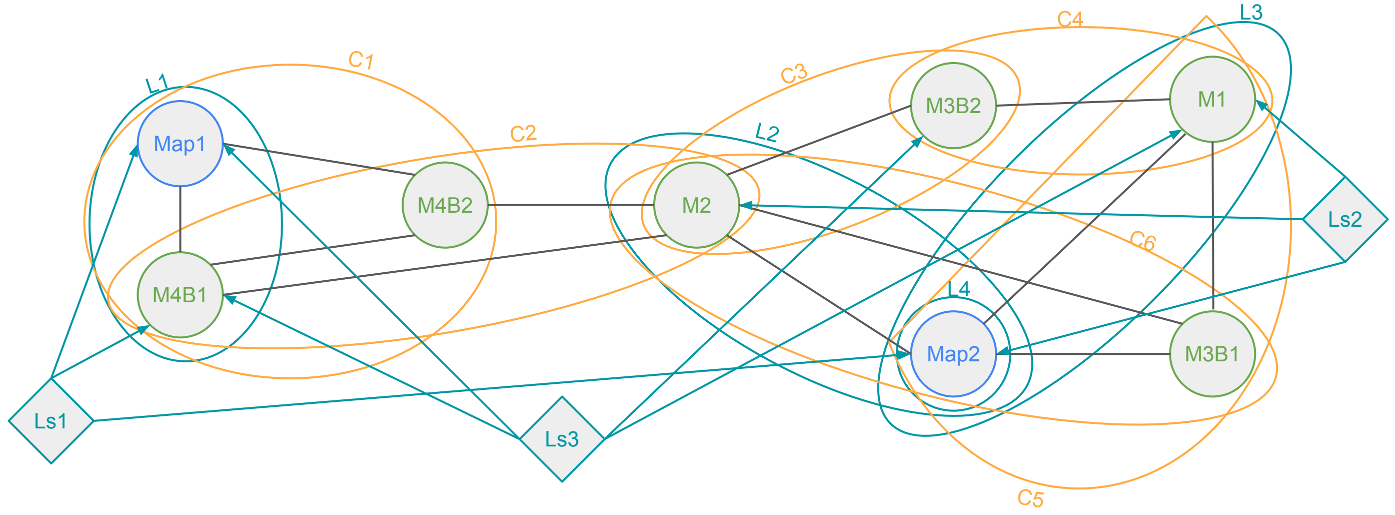
```
# map 2, add the layers that contain the node
- name: ["C5", "C6", "L2", "L3", "L4"]
  map: <image or dat file>
```

Add layers to the World file - Empty Map



```
# empty map, add the layers that are not contained in any other map
- name: ["C2", "C3", "C4"]
  map: <image or dat file>
```

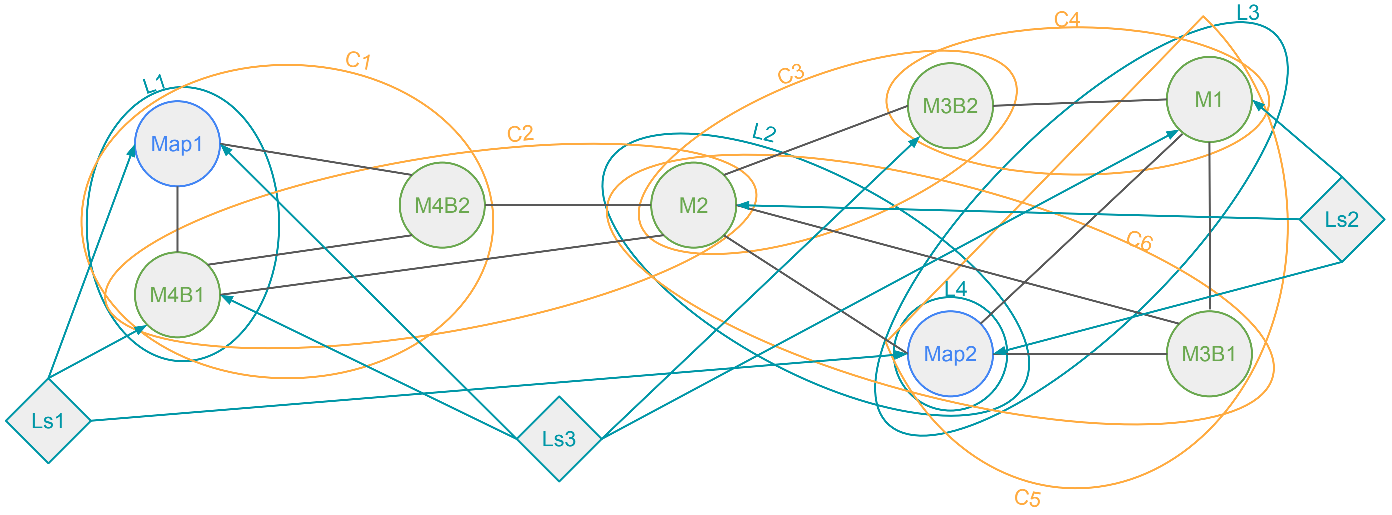
Assign layers to the models



Models (or bodies of the model) belong to all layers they are contained in:

- M1: C4, C5, L3
- M2: C2, C3, C6, L2
- M3:
  - B1: C5, C6
  - B2: C3, C4
- M4:
  - B1: C1, C2, L1
  - B2: C1, C2

Assign layers to the lasers

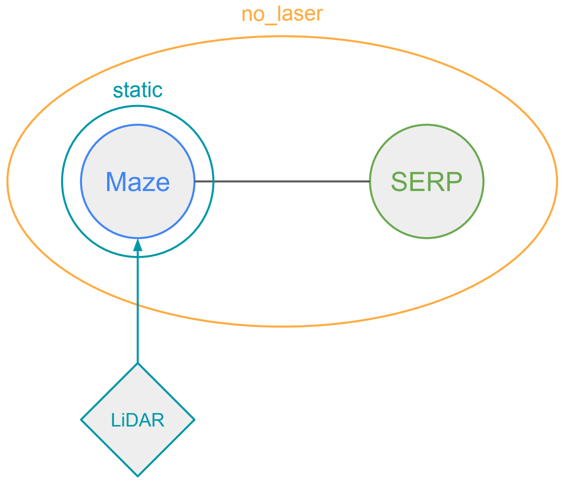


Lasers need to detect layers they are pointing to:

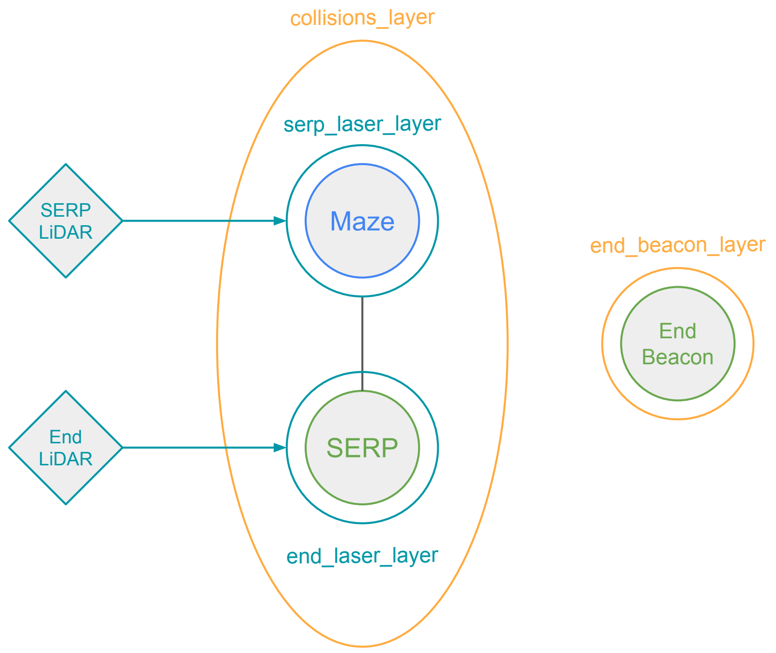
- Ls1: L1, L4
- Ls2: L2, L3
- Ls3: L1, C4

NOTE: There can be some redundancies in this step. Example: Ls2 is also pointing to L4 but since L2 (or L3) contains L4, it can be removed.

Graphs for the tutorial examples - Teleopkeys



Graphs for the tutorial examples - Reinforcement Learning


collisions_layer
serp_laser_layer
SERP LiDAR
Maze
end_beacon_layer
End Beacon
End LiDAR
SERP
end_laser_layer