Symmetri

A Petri net library for controlling your ROS Application

Thomas Horstink (thomas@mainblades.com)
https://github.com/thorstink/Symmetri
ROSCon ’23, October 20th

MAINBLADES

Introduction

The “novel approach” introduced today

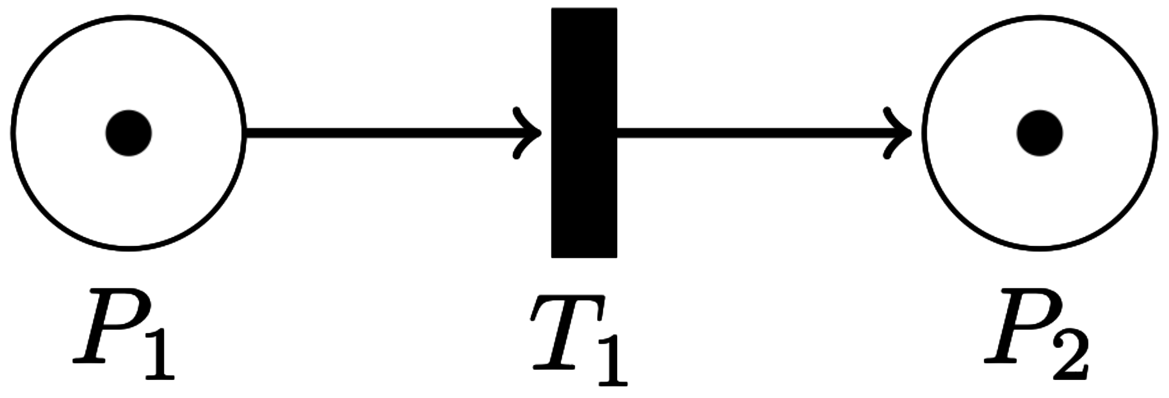
- Petri nets, sixties tech
  - Tooling (GreatSPN[1], etc) and standardisation
  - Mathematical formalism
- A C++ library, Symmetri, that executes Petri nets

This is not the first ROS-package that builds upon Petri nets (e.g. PetriNetPlans)

Petri nets

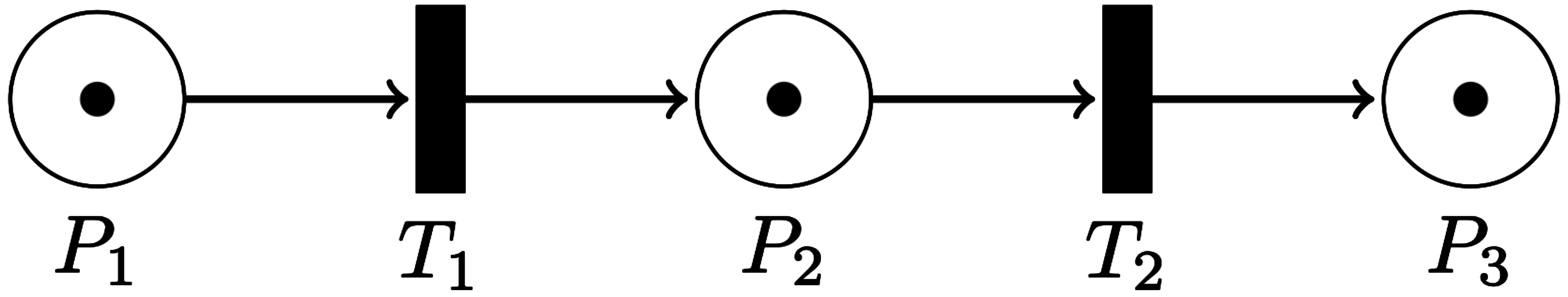
A mathematical modelling language for distributed systems

Places, transitions and tokens



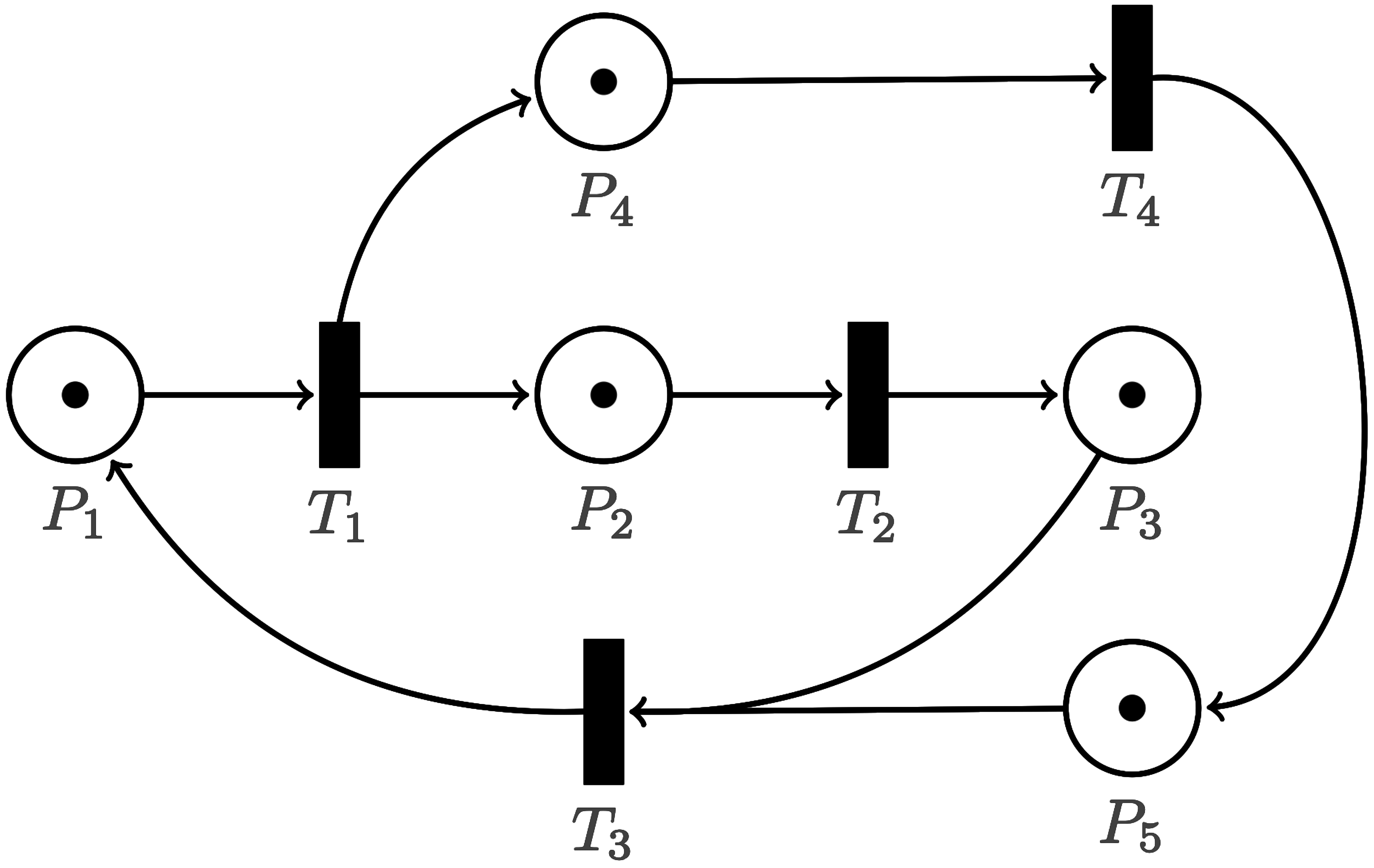
Petri nets

Two sequential transitions and three places



Petri nets

Looping and running transitions in parallel



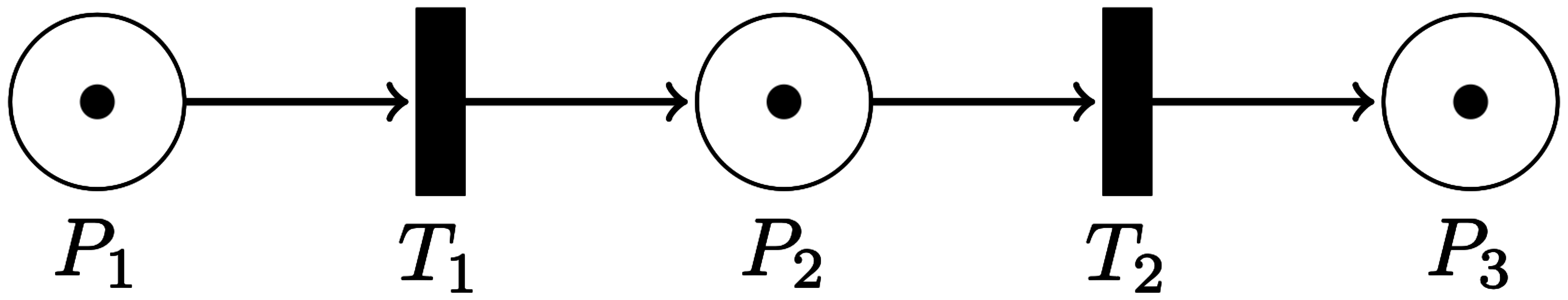
Symmetri

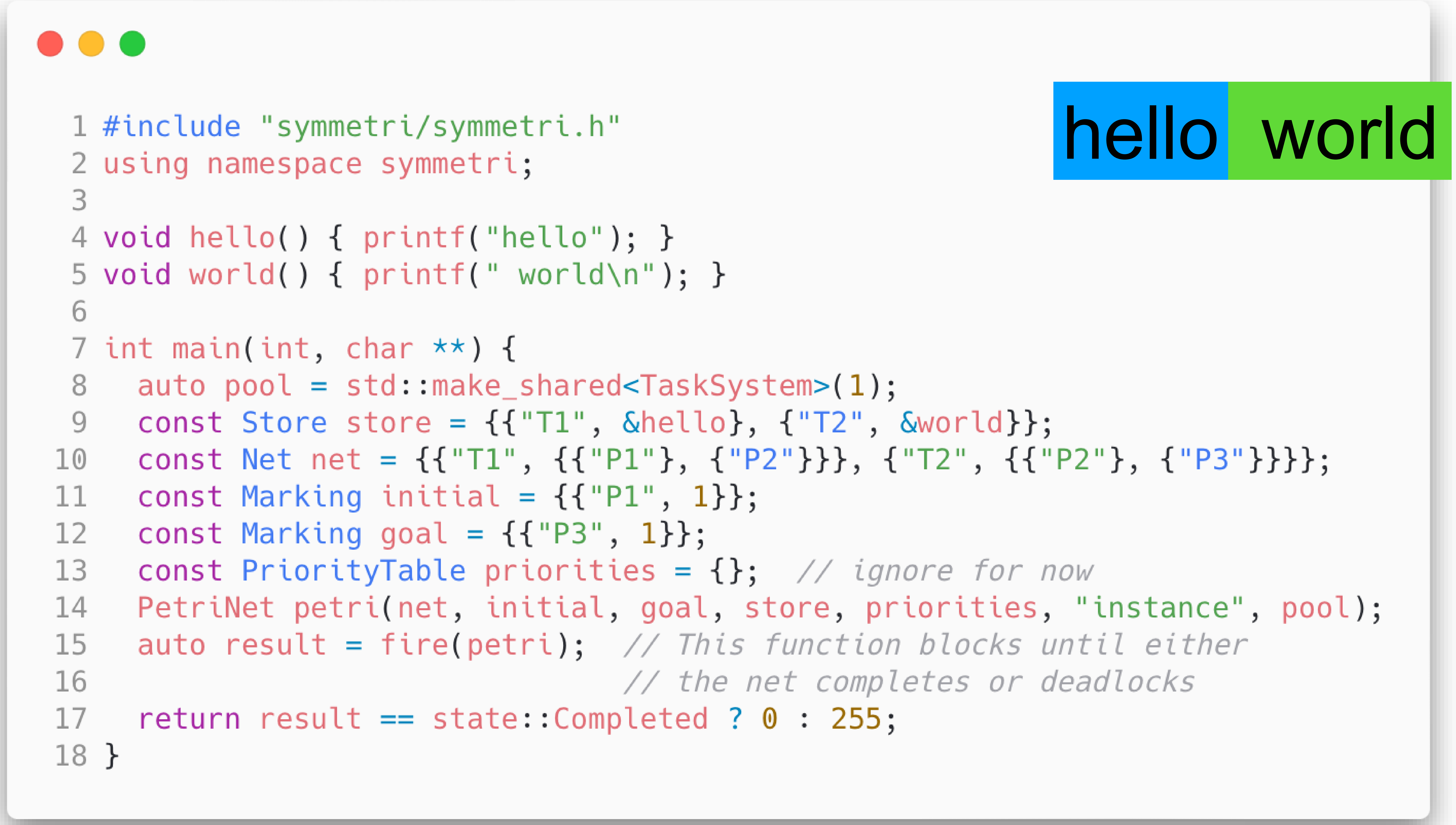
Symmetri is a C++ Petri net executor

- *Callbacks* are bound to transitions
- Optional special callbacks: *pause, resume* and *cancel*

Symmetri examples

Two sequential transitions



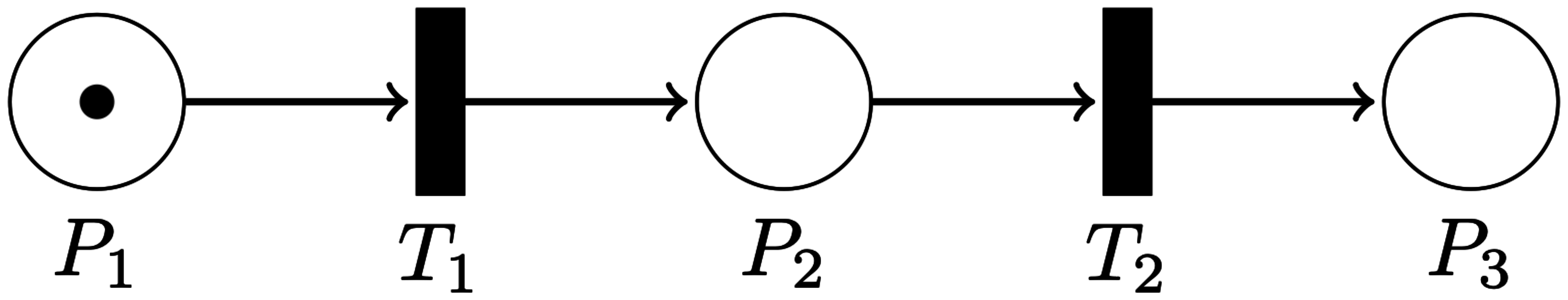


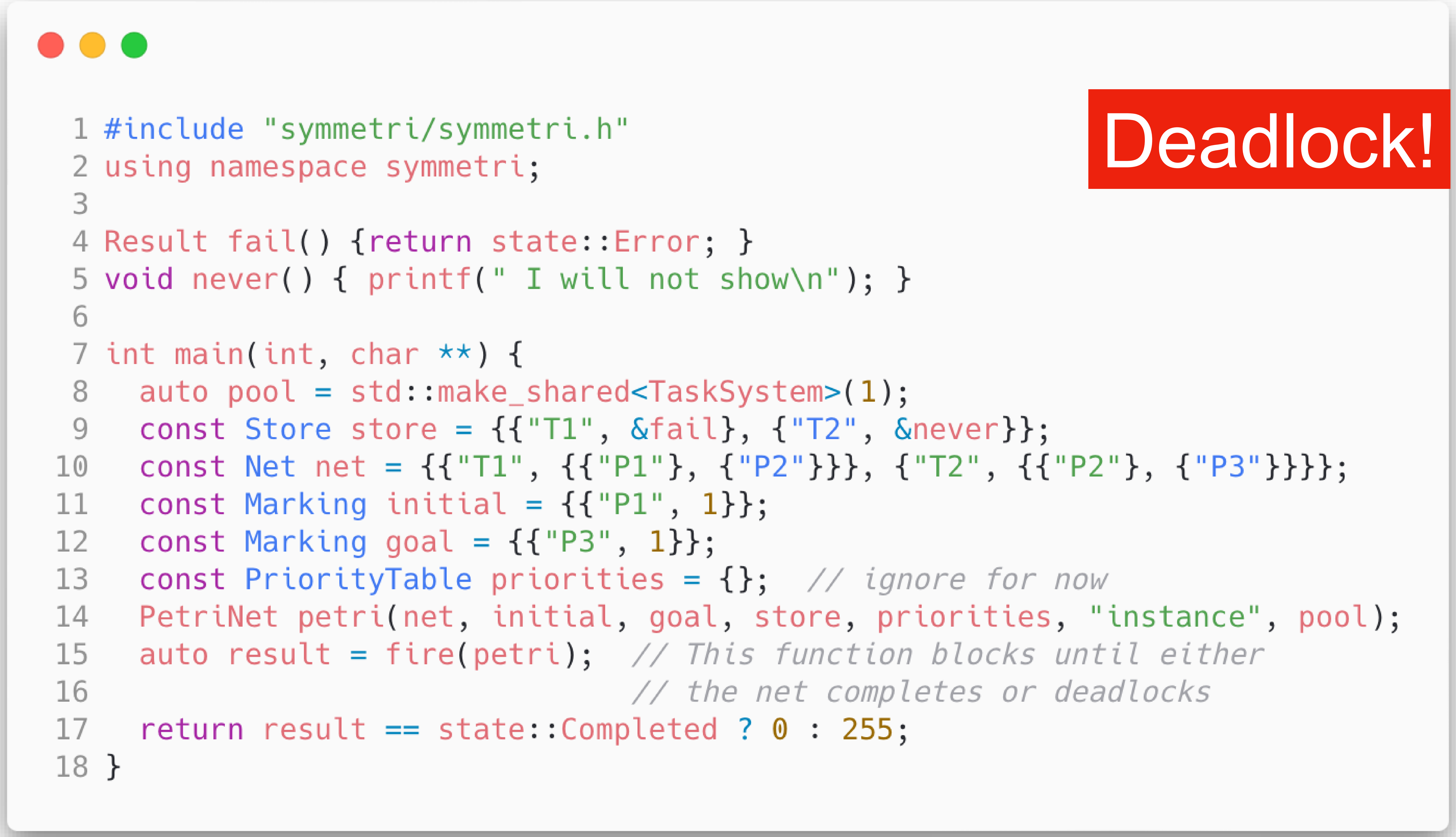
```cpp
#include "symmetri/symmetri.h"
using namespace symmetri;

void hello() { printf("hello"); }
void world() { printf(" world\n"); }

int main(int, char **) {
  auto pool = std::make_shared<TaskSystem>(1);
  const Store store = {{"T1", &hello}, {"T2", &world}};
  const Net net = {{"T1", {{"P1"}, {"P2"}}}, {"T2", {{"P2"}, {"P3"}}}};
  const Marking initial = {{"P1", 1}};
  const Marking goal = {{"P3", 1}};
  const PriorityTable priorities = {};  // ignore for now
  PetriNet petri(net, initial, goal, store, priorities, "instance", pool);
  auto result = fire(petri);  // This function blocks until either
                              // the net completes or deadlocks
  return result == state::Completed ? 0 : 255;
}
```

hello world

Symmetri examples

Customisation points





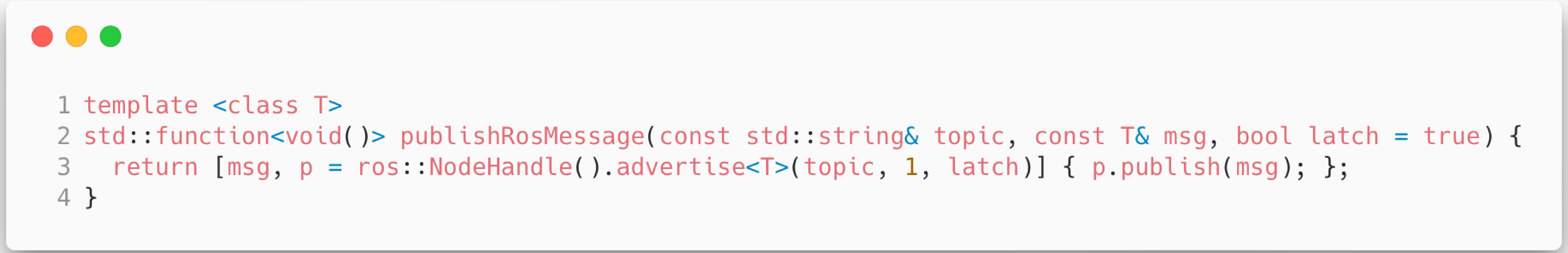
Deadlock!

```cpp
#include "symmetri/symmetri.h"
using namespace symmetri;

Result fail() {return state::Error; }
void never() { printf(" I will not show\n"); }

int main(int, char **) {
  auto pool = std::make_shared<TaskSystem>(1);
  const Store store = {{"T1", &fail}, {"T2", &never}};
  const Net net = {{"T1", {{"P1"}, {"P2"}}}, {"T2", {{"P2"}, {"P3"}}}};
  const Marking initial = {{"P1", 1}};
  const Marking goal = {{"P3", 1}};
  const PriorityTable priorities = {};   // ignore for now
  PetriNet petri(net, initial, goal, store, priorities, "instance", pool);
  auto result = fire(petri);  // This function blocks until either
                              // the net completes or deadlocks
  return result == state::Completed ? 0 : 255;
}
```

How can I

Symmetri & ROS (1)

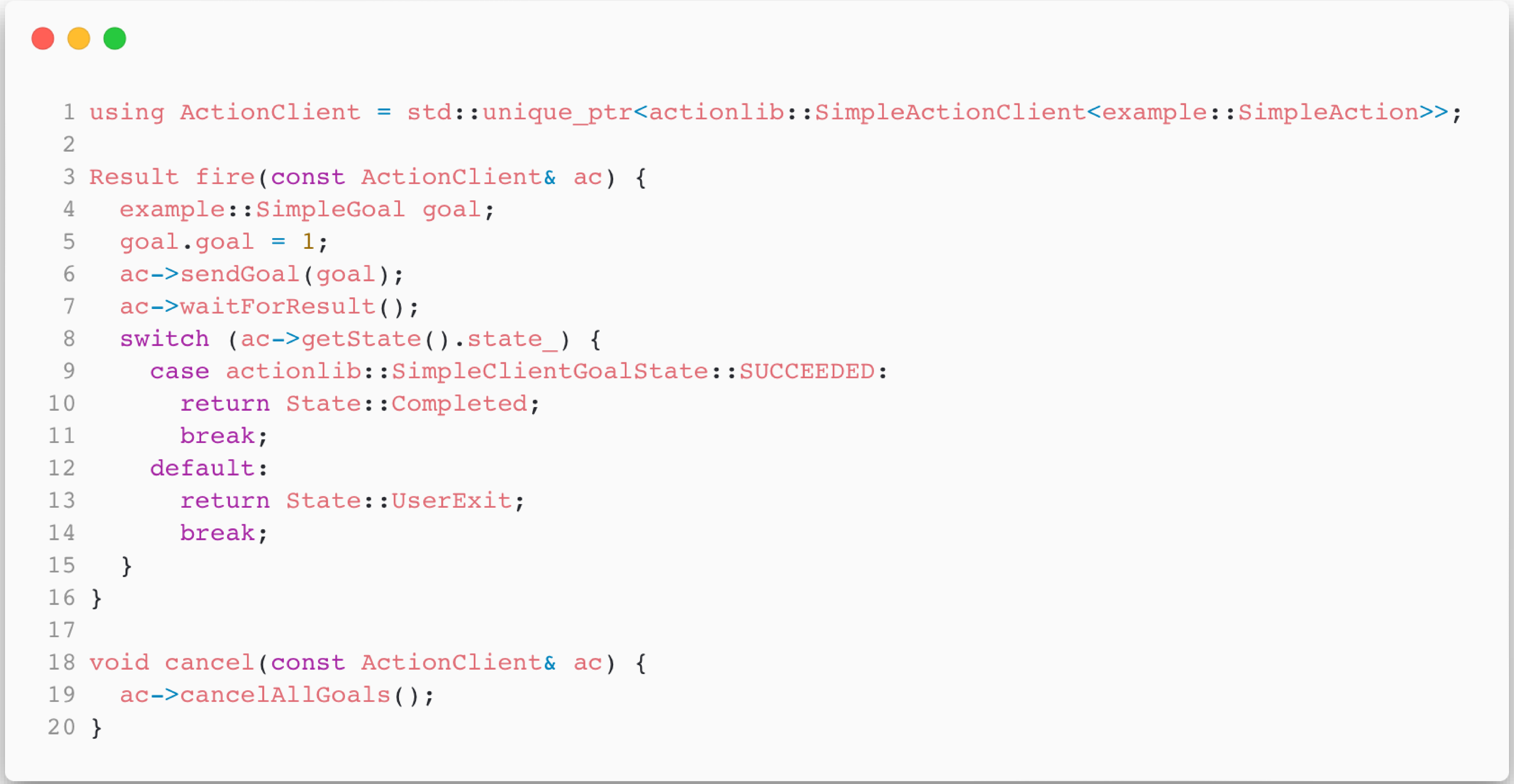
Example: a publisher transition



```
template <class T>
std::function<void()> publishRosMessage(const std::string& topic, const T& msg, bool latch = true) {
  return [msg, p = ros::NodeHandle().advertise<T>(topic, 1, latch)] { p.publish(msg); };
}
```

Symmetri & ROS (1)

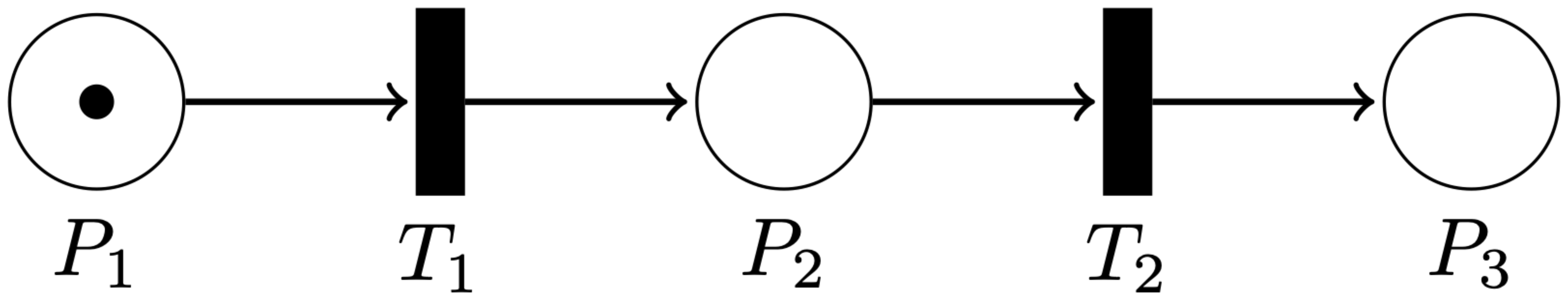
Example: SimpleActionClient transition

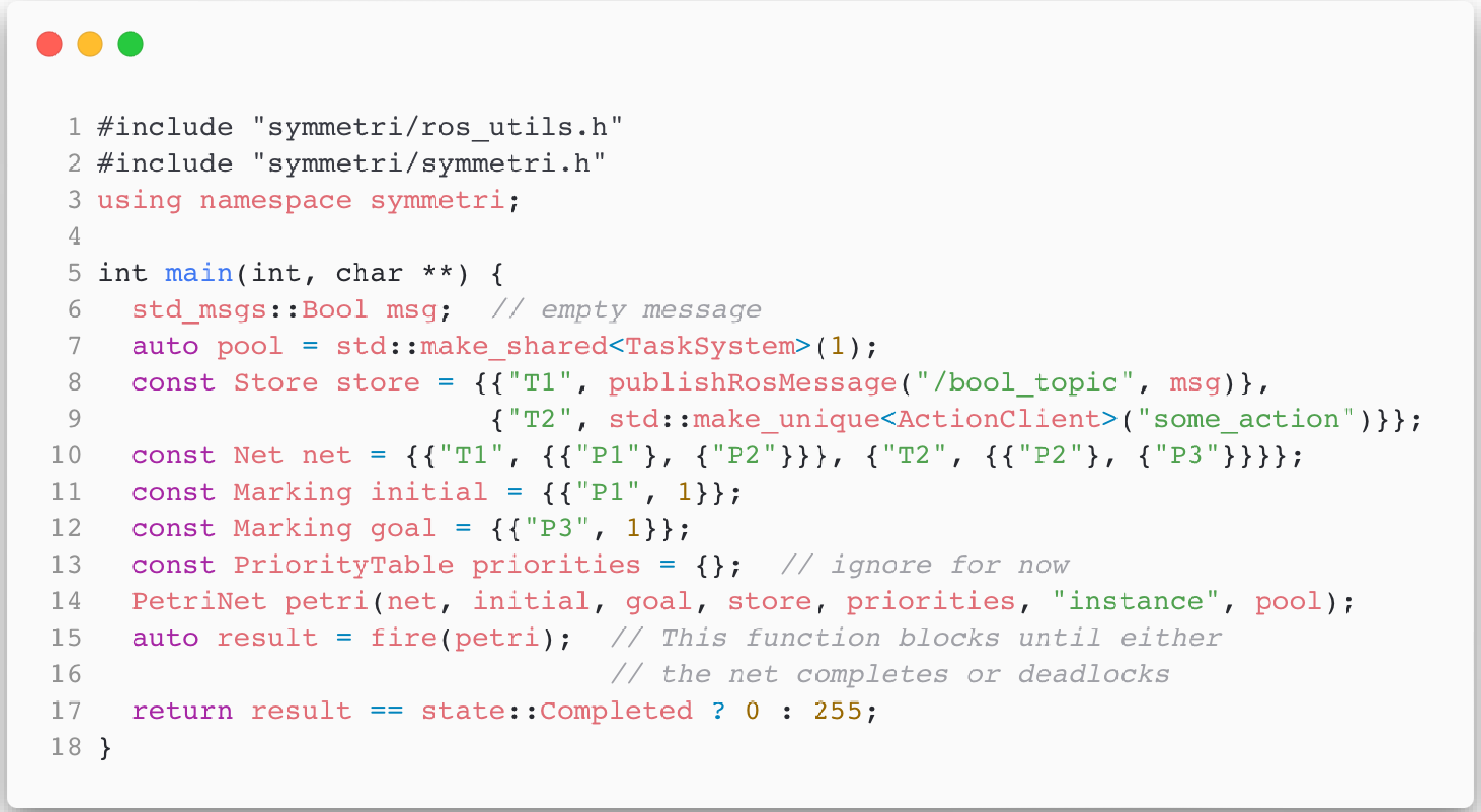


```cpp
using ActionClient = std::unique_ptr<actionlib::SimpleActionClient<example::SimpleAction>>;

Result fire(const ActionClient& ac) {
  example::SimpleGoal goal;
  goal.goal = 1;
  ac->sendGoal(goal);
  ac->waitForResult();
  switch (ac->getState().state_) {
    case actionlib::SimpleClientGoalState::SUCCEEDED:
      return State::Completed;
      break;
    default:
      return State::UserExit;
      break;
  }
}

void cancel(const ActionClient& ac) {
  ac->cancelAllGoals();
}
```

Symmetri & ROS

Example: putting it together



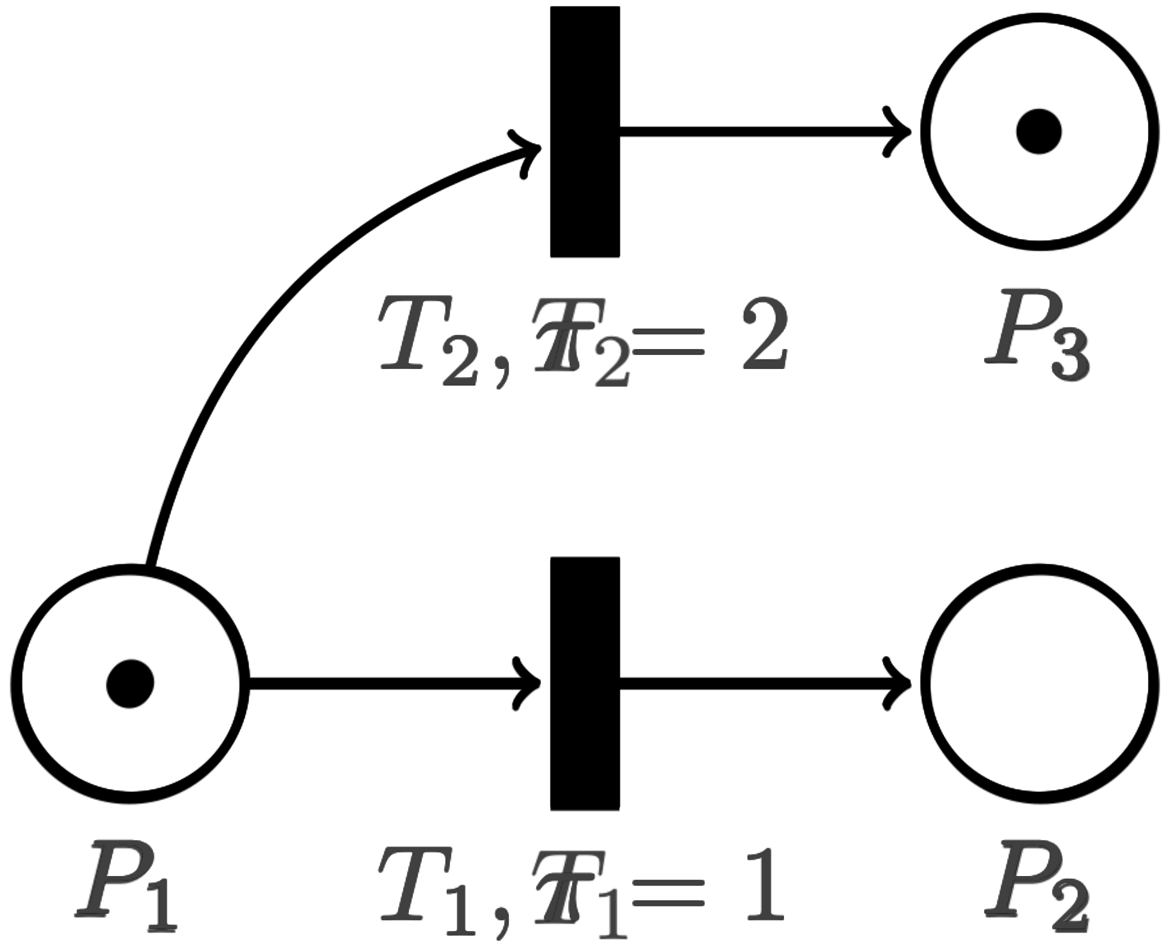


```cpp
#include "symmetri/ros_utils.h"
#include "symmetri/symmetri.h"
using namespace symmetri;

int main(int, char **) {
  std_msgs::Bool msg;  // empty message
  auto pool = std::make_shared<TaskSystem>(1);
  const Store store = {{"T1", publishRosMessage("/bool_topic", msg)},
                       {"T2", std::make_unique<ActionClient>("some_action")}};
  const Net net = {{"T1", {{"P1"}, {"P2"}}}, {"T2", {{"P2"}, {"P3"}}}};
  const Marking initial = {{"P1", 1}};
  const Marking goal = {{"P3", 1}};
  const PriorityTable priorities = {};  // ignore for now
  PetriNet petri(net, initial, goal, store, priorities, "instance", pool);
  auto result = fire(petri);  // This function blocks until either
                              // the net completes or deadlocks
  return result == state::Completed ? 0 : 255;
}
```

Conflict and scalability

Practical limitations and workarounds

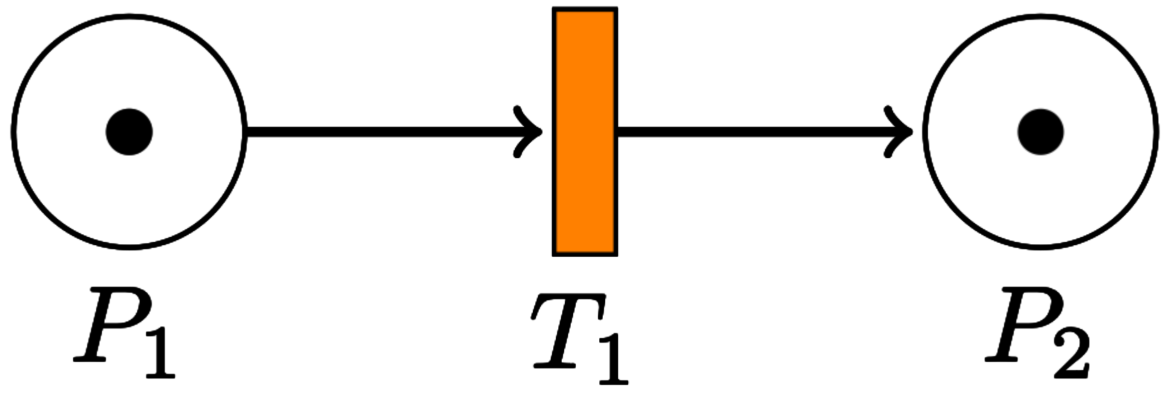
- Prioritisation[2]
- Clutter
- Hierarchy

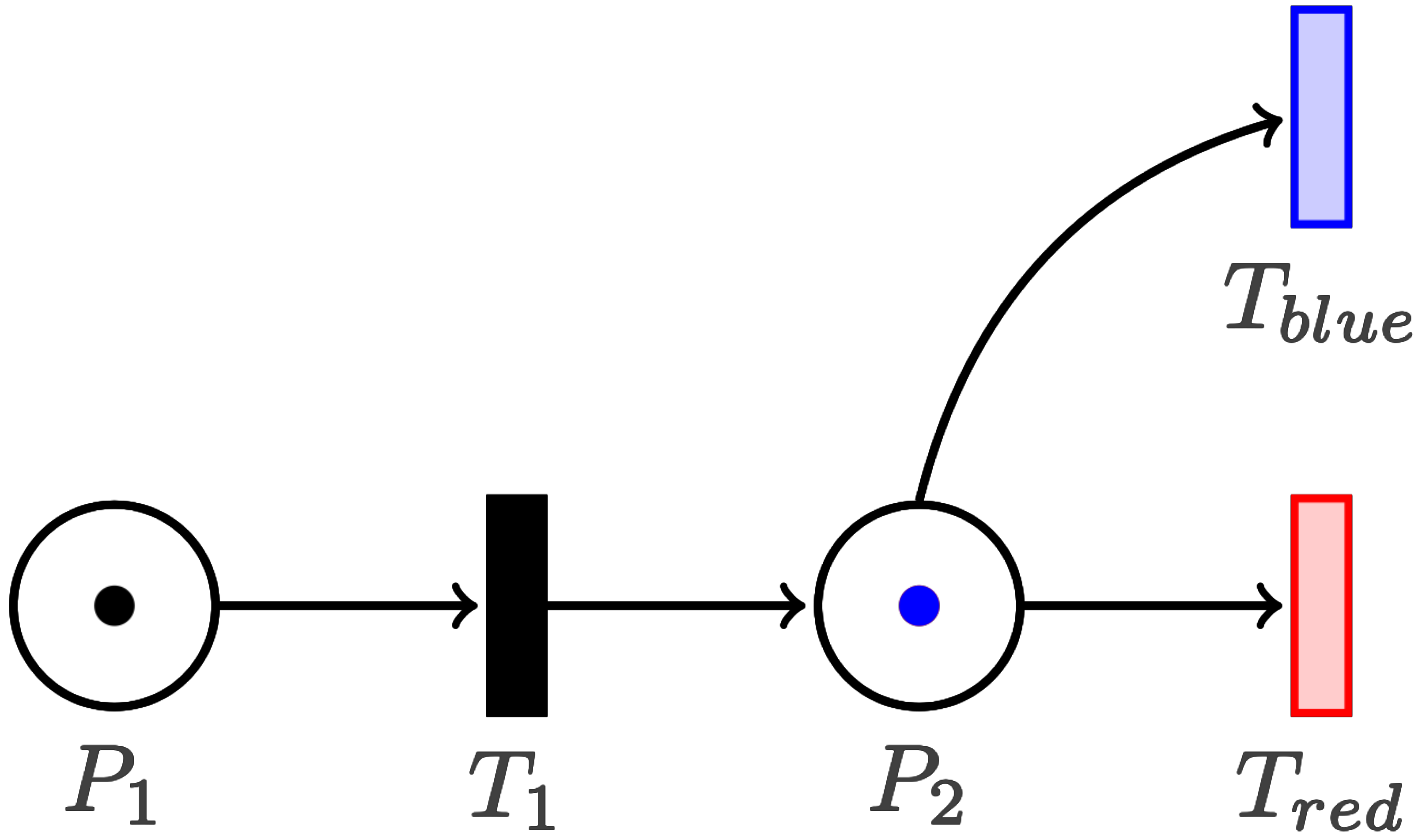


From black & white to Colours

Future functionality for Symmetri

- Tokens are black
- state::Error prevents token production
- Coloured tokens[3]





Summary

Symmetri & Petri nets

- Petri nets are
  - A modelling language
  - An execution protocol
- Symmetri is
  - A C++ library that executes Petri nets
  - Used in production by Mainblades
  - Almost API stable

Symmetri?

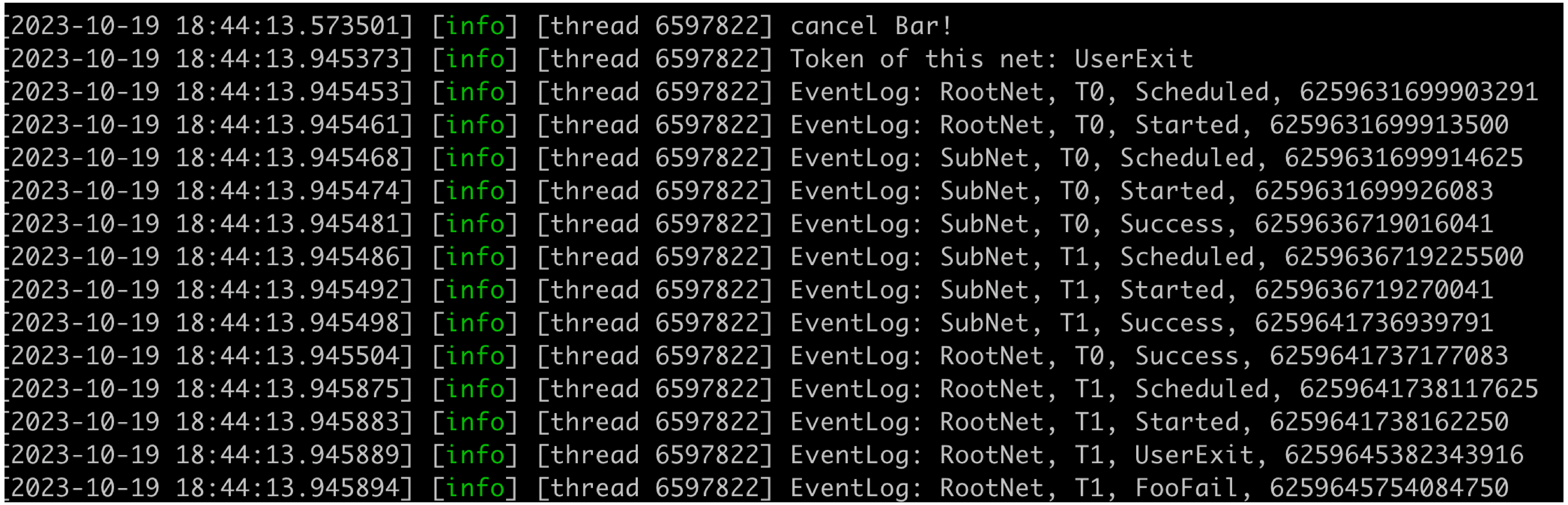
A Petri net library for controlling your ROS Application

Thomas Horstink (thomas@mainblades.com)
https://github.com/thorstink/Symmetri
ROSCon ’23, October 20th

MAINBLADES

Petri net logs are Event logs

- Business Process Mining inspired *event logs*
    - Case ID, Activity and Timestamp
- Also an execution trace



```
[2023-10-19 18:44:13.573501] [info] [thread 6597822] cancel Bar!
[2023-10-19 18:44:13.945373] [info] [thread 6597822] Token of this net: UserExit
[2023-10-19 18:44:13.945453] [info] [thread 6597822] EventLog: RootNet, T0, Scheduled, 6259631699903291
[2023-10-19 18:44:13.945461] [info] [thread 6597822] EventLog: RootNet, T0, Started, 6259631699913500
[2023-10-19 18:44:13.945468] [info] [thread 6597822] EventLog: SubNet, T0, Scheduled, 6259631699914625
[2023-10-19 18:44:13.945474] [info] [thread 6597822] EventLog: SubNet, T0, Started, 6259631699926083
[2023-10-19 18:44:13.945481] [info] [thread 6597822] EventLog: SubNet, T0, Success, 6259636719016041
[2023-10-19 18:44:13.945486] [info] [thread 6597822] EventLog: SubNet, T1, Scheduled, 6259636719225500
[2023-10-19 18:44:13.945492] [info] [thread 6597822] EventLog: SubNet, T1, Started, 6259636719270041
[2023-10-19 18:44:13.945498] [info] [thread 6597822] EventLog: SubNet, T1, Success, 6259641736939791
[2023-10-19 18:44:13.945504] [info] [thread 6597822] EventLog: RootNet, T0, Success, 6259641737177083
[2023-10-19 18:44:13.945875] [info] [thread 6597822] EventLog: RootNet, T1, Scheduled, 6259641738117625
[2023-10-19 18:44:13.945883] [info] [thread 6597822] EventLog: RootNet, T1, Started, 6259641738162250
[2023-10-19 18:44:13.945889] [info] [thread 6597822] EventLog: RootNet, T1, UserExit, 6259645382343916
[2023-10-19 18:44:13.945894] [info] [thread 6597822] EventLog: RootNet, T1, FooFail, 6259645754084750
```