FCT NOVA

Project Report

An Automated Factory in Go

Filip Brna and Leonard Storcks

Course by Dr. Bernardo Toninho

October 18, 2023

Abstract

In this project we designed a concurrent factory in Go with buffered channels at the heart of managing availability.

Contents

1 Introduction

With increasing automatization it is ever more important to efficiently manage and control factories with robot workers. We consider an industrial factory which does work on components. The processes offered are

- Welding
- Assembly
- Painting

which can be chained together to a task-set the factory works on. To perform those tasks, the factory includes

- I pickup-stations, N transport robots, D dropoff-stations
- N welding robots, W welding stations (each welding station needs two welders and one transporter to start the welding)
- N assembly robots, A assembly stations (each assembly station needs one assembler and one transporter to start the assembly)
- N painting robots, P painting stations (each painting station needs one painter and one transporter to start the painting)

which are managed by a central control station to which robots come back after finishing tasks (there is also a wireless connection to the robots).

The factory at hand is modeled as a concurrent system in go with channels at the heart of the communication and management processes, as detailed out in section 2. Test results of our system are provided in section 3.

2 Architecture and Implementation

Data Structures

We model resource-types as data structures, e.g. a worker (robot) is a struct with fields

- identity (integer, uniquely identifying the worker among workers of the same type)
- specialization (reference to the set of workers the worker belongs to)
- inbox (channel for receiving task-sets)
- next_facility (channel to receive the reference of the specific facility the worker should go to next)
- task_completed (channel on which a station can notify individual workers that a process has been completed)

where each worker is specified by this data-type and runs as a go-routine. Facilities (stations) are modeled similarly, each with a channel to which a worker can signal arrival and a channel on which tasks are received. Note that all workers and facilities run concurrently, waiting for, handling and finishing tasks. The correct messaging channels (e.g. channel for painter 6 to signal arrival to painting facility 5) are given to the workers (and stations) as soon as available / necessary.

Worker arrivals do not have to be in a specific order. The respective stations wait for all necessary workers (e.g. two welders and one transporter for the welding station) to arrive by means of draining its arrival channel. As of task assignment it is assured that the arriving workers are of the correct type, but this is - as a safety measure - also checked on the side of the facility.

Ensuring Correct Availability Management by Buffered Channels

The base of the architecture is that availability of each resource-type (e.g. welding robots, painting stations) is handled using buffered channels with the respective capacities given by the number of resources of that type. When a resource is needed, it is drained from the channel and added in when it is released (a station has finished processing a component, a robot has returned to the control station).

Overview on the Factory and the Workflow

The overall factory is illustrated in figure 2. When a task-set arrives (imagine a truck coming to the factory with a part that has to be welded first, then assembled, then painted), the

control center is notified (e.g. by the truck), and - as soon as available via the buffered channels - assigns a pickup station and a transportation worker. The task-set is handed to the transportation worker, that carries the component through the different processing stations (as specified by the task-set) and requests assignment of the respective specific station (and implicitly workers) from the control system which organizes everything and sends out the workers (see figure 1 for this workflow). More specifically in the initial task-set handed to the transporter only the facility types and processes are described (e.g. first pickup, then paint in blue, ...) and the specific facilities used and workers involved are only specified upon request along the way. Two different approaches to workflow organization are presented in table 1.

Central Approach (as illustrated in figure 1)	**Distributed Approach**
When the transportation worker gets to the next process in the task-set, it sends a request for a facility of the needed type to the control center which assigns a facility when available and the respective workers and directs the transporter to the correct facility. Requests for different facility types are handled concurrently but multiple requests from multiple transporters on the same facility type are handled sequentially.	Alternatively, the transportation worker could directly draw resources from the buffered (facility and worker) channels and send out the notifications. This way, multiple requests to one resource type (e.g. painting facilities & workers) can be handled in parallel. The welding station - which needs two welders - has to be paid special attention to in this approach, as transporters rush on the channel with welders which can easily lead to deadlocks if not properly avoided.

Table 1: Comparison of Central and Distributed Approaches

We employ the more central approach as it seems a bit simpler and more power is in the hand of the control system. Also note, this is only about sending out the tasks (which should be very quick) and e.g. different painting jobs still run concurrently.

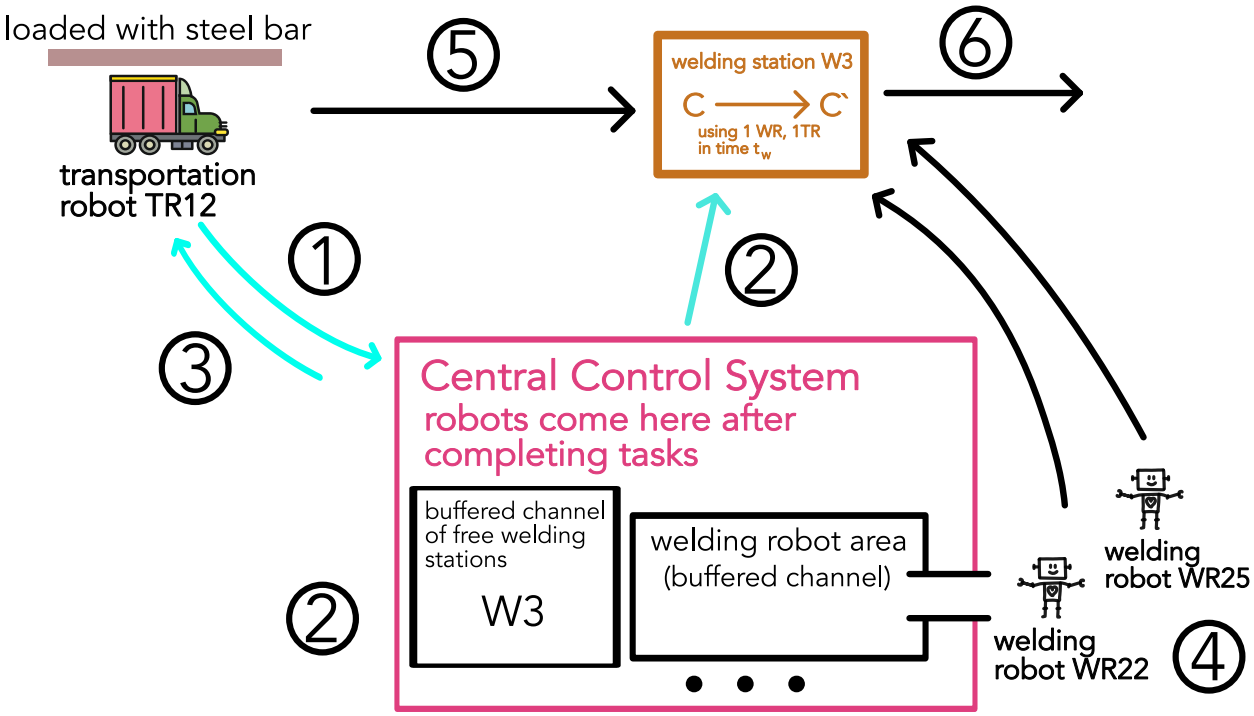

Figure 1: Example workflow for one process. 1: The transportation robot requests a welding station from the control center; 2: If there is an available welding station, it is assigned to the task and notified; 3. After a welding station is assigned, the transporter is notified and commutes there (communication channel to the station is also transmitted); 4. As soon as available, the system sends out welding robots equipped with the correct communication channel.; 5. When all robots have notified the station about their arrival, welding can start; 6. When welding is complete, all robots involved are notified, the welders go back to control, the transporter continues the task-set.

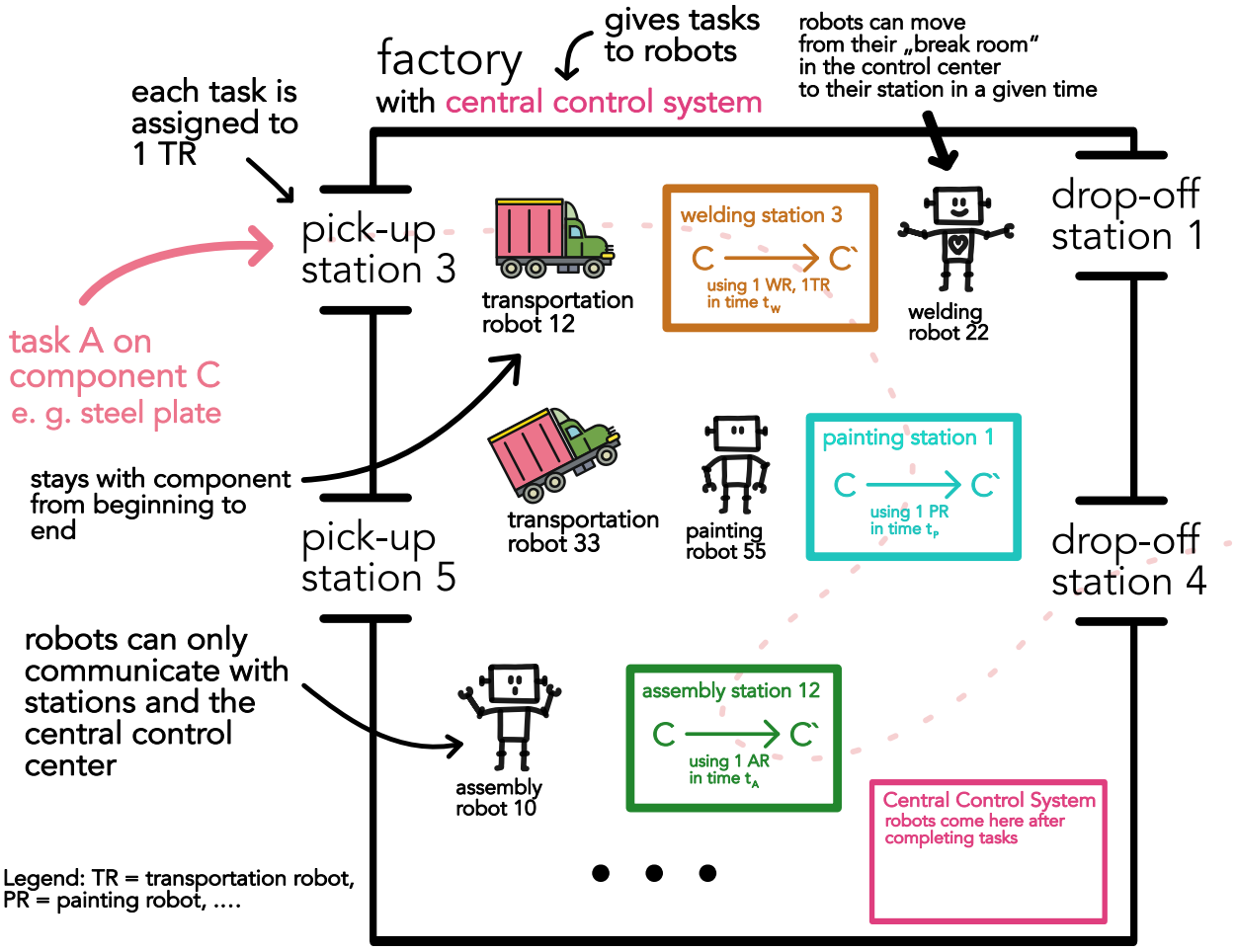

Figure 2: Overview on the factory.

3 Tests

TestNoPickup

Description: Test checks the system's response when a pickup station is missing. It sends a task that is impossible to complete without a pickup station and ensures that the task is not completed. Factory is booted with 1 dropoff station and 1 robot of each kind.

Expected outcome: The test expects that the number of completed tasks should be 0.

TestNoDropoff

Description: This test checks the system's response when a dropoff station is missing. It sends a task that is impossible to complete without a dropoff station and ensures that the task is not completed. Factory is booted with 1 pickup station and 1 robot of each kind.

Expected outcome: The test expects that the number of completed tasks should be 0.

TestPickupAndDropoffStation

Description: This test checks the functionality of both pickup and dropoff stations by sending a task set that involves only picking up and dropping off components. It ensures that the task is completed. Factory is booted with 1 pickup station, 1 dropoff station, and 1 robot of each kind.

Expected outcome: The test expects that the number of completed tasks should be 1.

TestWeldingStation

Description: This test checks the functionality of the welding station by sending a task set that involves pickup, welding, and dropoff operations on components. It ensures that the task is completed. Factory is booted with 1 pickup station, 1 dropoff station, 1 welding station and 2 robots of each kind.

Expected outcome: The test expects that the number of completed tasks should be 1.

TestPaintingStation

Description: This test checks the functionality of the painting station by sending a task set that involves pickup, painting, and dropoff operations on components. It ensures

that the task is completed. Factory is booted with 1 pickup station, 1 dropoff station, 1 painting station and 1 robot of each kind.

Expected outcome: The test expects that the number of completed tasks should be 1.

TestAssemblyStation

Description: This test checks the functionality of the assembly station by sending a task set that involves pickup, assembly, and dropoff operations on components. It ensures that the task is completed. Factory is booted with 1 pickup station, 1 dropoff station, 1 assembly station and 1 robot of each kind.

Expected outcome: The test expects that the number of completed tasks should be 1.

TestFree

Description: This test checks if the system properly frees workers and facilities after completing a task set that involves multiple stations. It sends a task that consists of pickup, welding, painting, assembly, and dropoff operations on components. It ensures that the task is completed. Factory is booted with 2 stations of each kind and 2 robots of each kind.

Expected outcome: The test expects that all workers and facilities should be free after the task is completed and the number of completed tasks should be 1. Example output is shown in figure 3.

TestAllCompletedTaskSets

Description: This test checks if the system can handle and complete multiple task sets simultaneously. It sends 3 task sets with pickup, welding, painting, assembly, and dropoff operations on components. It ensures that all task sets are completed. Factory is booted with 2 stations of each kind and 2 robots of each kind.

Expected outcome: The test expects that all requested task sets are completed, and all workers and facilities are free and the number of completed tasks should be 3.

TestComplexTasks

Description: This test involves a more complex scenario where multiple task sets are sent to the system. It sends 5 task sets with pickup, welding, painting, assembly, and dropoff operations on components. It ensures that all task sets are completed. Factory is

booted with 2 pick up stations and drop off stations and 1 welding station, painting station, assembly station. It also has 2 robots of each kind.

Expected outcome: The test expects that all requested task sets are completed, and all workers and facilities are free and the number of completed tasks should be 5.

```
booted factory with 2 pick-up stations, 2 assembly stations, 2 welding stations, 2 painting stations,
2 drop-off stations, 2 assembly workers, 2 welding workers, 2 painting workers and 2 transport workers

: taskset 1 received.

: taskset 1 arrived at transportation worker 0
[ 1 ] : task pickup steel bar arrived at pickup station 0
[ 1 ] : transportation worker 0 arrived at pickup station 0
[ 1 ] : Pickup station 0 is free again
[ 1 ] : task weld steel bar arrived at welding worker 1
[ 1 ] : task weld steel bar arrived at welding station 0
[ 1 ] : task weld steel bar arrived at welding worker 0
[ 1 ] : next facility of transportation worker 0 is welding number 0
[ 1 ] :  welding  worker 0 arrived at welding station 0
[ 1 ] :  welding  worker 1 arrived at welding station 0
[ 1 ] :  transport  worker 0 arrived at welding station 0
[ 1 ] : welding task finished
[ 1 ] : Welding station 0 is free again
[ 1 ] : task assemble steel bar arrived at assembly worker 0
[ 1 ] : task assemble steel bar arrived at assembly station 0
[ 1 ] : next facility of transportation worker 0 is assembly number 0
[ 1 ] :  transport  worker 0 arrived at assembly station 0
[ 1 ] :  assembly  worker 0 arrived at assembly station 0
[ 1 ] : welding worker 1 arrived at control center
[ 1 ] : welding worker 0 arrived at control center
[ 1 ] : assembly task finished
[ 1 ] : Assembly station 0 is free again
[ 1 ] : task paint steel bar in blue arrived at painting worker 0
[ 1 ] : next facility of transportation worker 0 is painting number 0
[ 1 ] : task paint steel bar in blue arrived at painting station 0
[ 1 ] : assembly worker 0 arrived at control center
[ 1 ] :  transport  worker 0 arrived at painting station 0
[ 1 ] :  painting  worker 0 arrived at painting station 0
[ 1 ] : painting task finished
[ 1 ] : Painting station 0 is free again
[ 1 ] : next facility of transportation worker 0 is dropoff number 0
[ 1 ] : task dropoff steel bar arrived at dropoff station 0
[ 1 ] : transportation worker 0 arrived at dropoff station 0
[ 1 ] : painting worker 0 arrived at control center
[ 1 ] : dropoff task finished
[ 1 ] : Dropoff station 0 is free again

taskset 1 was completed

[ 1 ] : transport worker 0 arrived at control center
```

Figure 3: Printscreen of example output for test 3.

4 Summary and Outlook

In conclusion, using the channel-based approach in Go we were able to design a somewhat elegant factory (without using any locks or similar techniques) where the next steps could be to optimize task distributions so that a set of arriving components is processed as fast as possible.