A Software Architecture for Transport in a Production Grid

Leo van Moergestel, Erik Puik
Institute for ICT
HU Utrecht University of Applied Sciences
Utrecht, the Netherlands
Email: leo.vanmoergestel@hu.nl, erik.puik@hu.nl

John-Jules Meyer
Intelligent systems group
Utrecht University
Utrecht, the Netherlands
Alan Turing Institute Almere, The Netherlands
Email: J.J.C.Meyer@uu.nl

***Abstract*—In this paper, a software architecture is proposed to implement transport of products being made along production units. In the classical approach, production lines are used where products all follow a similar linear path during production. New production methods require a more agile and flexible path to meet the requirements of different paths to be followed during production to enable the productions of products with different user requirements, as well as a more fault tolerant production system. Starting with results from simulation, the requirements of the software architecture are established. The architecture proposed is inspired by the architecture used in software defined networks, that play a mayor role in complex computer networks.**

***Keywords*–Agent technology; Agile manufacturing; Production software architecture.**

I. Introduction

Today, information technology plays a major role in manufacturing as well as in other aspects of our modern society. In manufacturing, the trend is towards low-cost agile manufacturing of small batch sizes or even one product according to end-user requirements. When looking at the industrial revolutions, the first revolution was the use of steam power to facilitate production. The second revolution was the introduction of production lines based on the use of electrical energy. This resulted in economic and feasible mass production. Computer technology resulted in the third revolution, where many production tasks were automated by the use of Programmable Logic Controllers (PLCs), Distributed Control Systems (DCS) and robots. The latest revolution is the integration of information technology in the production process as a whole. This has been described by the term industry 4.0 [1] or cyber physical systems [2].

One of the ideas behind the concept is production on demand according to end-user requirements. To accomplish this, new production paradigms should be developed. One of the requirements of these new paradigms is the search for alternatives for the so-called production lines, where mass-production is realised by a linear sequence of production units or cells. Every unit offers a single production step in the sequence of steps needed to realise the final product.

In our research group, a set of cheap reconfigurable production machines called equiplets has been proposed as the production platforms that should be combined with a flexible transport system between these equiplets. This resulted in the concepts of a grid of these equiplets that should be capable to produce a variety of different products in parallel [3]. The concept fits in the concepts of Industry 4.0 or cyber physical systems. This paper will focus on the architecture to be used to implement a flexible transport of products during production. Though based on our concept of grid production using equiplets, this model can also be used in situations where a flexible transport between production units or cells is needed. The concept of grid production presented here, does not focus on a specific industry, but should be considered as a generic production concept.

The rest of this paper is organised as follows. Section II is dedicated to related work. Section III discusses the production model in more detail. In Section IV, the simulation model and implementation is presented, followed by Section V showing some results of the simulation. Section VI discusses the architecture that can be used to implement the real transport system and a conclusion will end the paper.

II. Related work

In this section, an overview will be given on agent-based manufacturing. Especially the planning part will be given attention. Important work in the field of agent-based manufacturing has already been done. Paolucci and Sacile [12] give an extensive overview of what has been done. Their work focuses on simulation as well as production scheduling and control [13]. The main purpose to use agents in [12] is agile production and making complex production tasks possible by using a multiagent system. Agents are also proposed to deliver a flexible and scalable alternative for manufacturing execution systems (MES) [14] for small production companies. The roles of the agents in this overview are quite diverse. In simulations agents play the role of active entities in the production. In production scheduling and control agents support or replace human operators. Agent technology is used in parts or sub-systems of the manufacturing process. The planning is mostly based on the type of planning that is used in MES. This type of planning is normally based on batch production. We based the manufacturing process as a whole on agent technology. In our case, a co-design of hardware and software was the basis. The planning will be done on a single product basis and not on batch production.

Bussmann and Jennings [15][16] used an approach that compares in some aspects to our approach. The system they describe introduced three types of agents, a workpiece agent, a machine agent and a switch agent. Some characteristics of their solution are:

- The production system is a production line that is built for a certain product. This design is based on redundant production machinery and focuses on production availability and a minimum of downtime in the production process. Our system is a grid and is capable to produce many different products in parallel;
- The roles of the agents in this approach are different from our approach. The workpiece agent sends an

invitation to bid for its current task to all machine agents. The machine agents issue bids to the workpiece agent. The workpiece agent chooses the best bid or tries again. This is what is known as the contract net protocol. In our system the negotiating is between the product agents, thus not disrupting the machine agents;

- They use a special infrastructure for the logistic subsystem, controlled by so-called switch agents. Even though the practical implementation is akin to their solution, in our solution the service offered by the logistic subsystems can be considered as production steps offered by an equiplet and should be based on a more flexible transport mechanism.

So, there are important differences between the approach of Bussmann and our approach. The solution presented by Bussmann and Jennings has the characteristics of a production pipeline and is very useful as such, however it is not meant to be an agile multi-parallel production system as presented here. Their system uses redundancy to overcome the problem that arises in pipeline-based production when one of the production systems fails or becomes unavailable. The planning is based on batch processing.

Other authors focus on using agent technology as a solution to a specific problem in a production environment. In [17], a multi-agent monitoring is presented. This work focusses on monitoring a manufacturing plant. The approach we use monitors the production of every single product. The work of Xiang and Lee [18] presents a scheduling multiagent-based solution using swarm intelligence. This work uses negotiating between job-agents and machine-agents for equal distribution of tasks among machines. The implementation and a simulation of the performance is discussed. We did not focus on a specific part of the production but we developed a complete production paradigm based on agent technology in combination with a production grid. This model is based on two types of agents and focuses on agile multiparallel production. The role of the product agent is much more important than in the other agent-based solutions discussed here. In our model, the product agent can also play an important role in the life-cycle of the product [19]. The design and implementation of the production platforms and the idea to build a production grid can be found in Puik [20].

III. PRODUCTION MODEL

Industry 4.0 is also characterised as a cyber physical system. In this section, these two parts will be explained starting with the physical aspect.

A. Physical aspect

As stated in the introduction, the actual production is done by so-called equiplets. An equiplet is a reconfigurable production machine [4]. Every equiplet is capable to perform one or more production steps. A definition of a production step is: *A production step is an action or group of coordinated or coherent actions on a product, to bring the product a step further to its final realisation. The states of the product before and after the step are stable, meaning that the time it takes to start the next step can be short or long for the production as a process (not for the production time) and that the product thus can be transported or temporarily stored between two steps.* A sequence of production steps should be performed to create a product. To accomplish this, a set of equiplets should be used in a certain order. This is done by moving platforms that can transport components, as well as the product itself from equiplet to equiplet. In Figure 1, this setup is shown. The arrows show a global path a product has to follow [5]. The equiplets are placed in a grid. The transport platform is first loaded with components needed for the production and will enter the grid where the components are handled by the equiplets to create a product (or product part or a product that is used as a component for the final product, a so-called half product). When this is done, the product will be finished or in case of product parts or half products, the grid can be re-entered to handle different product parts to make the final product. Different products need specific production steps

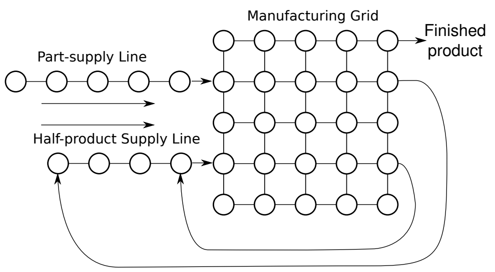


Figure 1. Grid production setup

in their own perhaps different order. The transport between the equiplets for a certain product will look like the path depicted by arrows in Figure 2. This particular path is actually a production line for that specific product mapped on the grid. The strength and versatility of the system is that every product can have its own path in the grid, resulting in a unique tailor made product. Complex products consisting of a set of half products can be built using the same principle. In that case, multiple paths should be followed to create the half products and the grid should be re-entered.

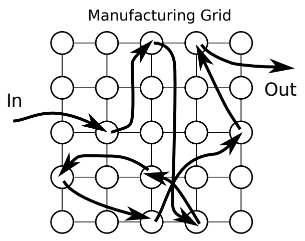


Figure 2. Path for a certain product

B. Cyber aspect

The software entities that control the production are software agents [6]. An equiplet is represented by an equiplet agent and every product to be made is represented by its own product agent. The robot platforms or moving production platforms are part of the transport infrastructure. The architecture of this infrastructure is discussed in Section VI.

As a start to create a product, a product agent is generated. This agent knows what production steps should be taken and the components to be used. The product agents allocates a transport system and collects the components. Next, it will pass along the equiplets required to perform the production steps. The product agent can discover the equiplets needed by looking at a blackboard system where the equiplets have published the production steps they are cappable to perform. Before an equiplet is chosen, the product agent will first investigate if the equiplet is really capable to perform the specific production step needed, given the parameters involved. To do so the equiplet will run a precise simulation of the step with the parameters given to discover the possibility and the time needed to bring the production step to an end. It will then inform the product agent about success or failure. When the actual real production step is performed, the product agent will also be informed about success or failure, but also the production parameters that had been used. This might be the exact temperature, or the amount and type of adhesive used, etc. Finally, the product agent has a complete production log of the product it represents.

In our model, the creation of a product agent can be done by using a webinterface where the end-user can specify his/her product to be made [7].

IV. Simulation model and implementation

The simulation model presented in this paper opens the possibility to explore the behaviour of the production system as a whole, taking into account, the transport as well as the time to perform a production step. The model is based on the production model described in the previous section. This means that at random times an agent enters the grid with a list of steps to perform, resulting in a list of equiplets to be visited. This situation is comparable to a group of people shopping in a shopping center, where they need to buy items available in different shops. Everybody is doing this autonomously and according to their own specific shopping list.

To make the simulation versatile, a decision was made to use a graph approach for the description of the grid. The advantage is that all kinds of interconnected nodes can be simulated including a grid so this approach is more powerful and can also be used in a grid where some of the interconnections are obstructed or impossible to use.

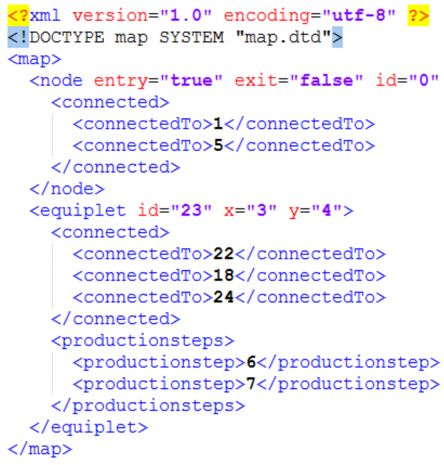
The simulation is driven by three different information files. These file are XML-files so human- and computer-readable.

1) the file maps.xml describes the structure of the grid, actually the structure of the graph;
2) the steps needed for a certain product;
3) the products to be made.

In Figure 3 an example of a map is shown. A map consists of nodes and equiplets, where an equiplet is actually a node offering production steps. Both nodes and equiplets have an unique id, an x-coordinate and y coordinate. A node can also be an entry point and/or exit point of the grid. Equiplets have a set of at least one production step. This way, all kind of production infrastructures fitting in our production model, can be expressed.

The simulation is controlled by a central clock. The simulation is not a realtime simulation, but by using this clock as the central heartbeat, a lot of concurrency problems could be prevented.

A path finding solution is in case of this particular simulation one of the challenges. The production system is based on



```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE map SYSTEM "map.dtd">
<map>
  <node entry="true" exit="false" id="0" x="0" y="0">
    <connected>
      <connectedTo>1</connectedTo>
      <connectedTo>5</connectedTo>
    </connected>
  </node>
  <equiplet id="23" x="3" y="4">
    <connected>
      <connectedTo>22</connectedTo>
      <connectedTo>18</connectedTo>
      <connectedTo>24</connectedTo>
    </connected>
    <productionsteps>
      <productionstep>6</productionstep>
      <productionstep>7</productionstep>
    </productionsteps>
  </equiplet>
</map>
```

Figure 3. XML content describing the grid

autonomous entities, actually the product agents, that share the production grid, each having a specific goal, and each making the product it represents. The way this goal should be accomplished should fit in the common goal of the system, a versatile agile production system. The path finding solution used was based on a special map that was generated, telling for every node how far the distance to a certain production step (equiplet) was. A moving platform would choose a direction towards the production step node. If this was not possible it would choose a node having the next lowest distance to the production step node. The reason for making this choice can be found in [8].

The simulation has been implemented as two components. First, there is the core system that actually performs the simulation. The second component is a graphical user interface (GUI) that will show in detail the working of the production system. It is possible to use the core system without the GUI if a lot of simulation runs should be made to generate data that can be studied afterwards.

Java has been used as the language for implementation. It is not considered to be the fastest language, but it fits well in modern software engineering concepts. The fact that many multiagent platform implementations are also based on Java was a second reason to use this language, because this simulation can also become part of the production software that is actually a multiagent system based on Jade. Jade is a Java-based multiagent programming environment.

V. Some simulation results

The implementation resulted in a simulation system that can be used with or without a GUI. The grid consists of nodes that are connected in a certain way. It is actually a graph as mentioned earlier. The edges of the graph can be unidirectional or bidirectional. A node can host an equiplet, but also be empty.

The first result that will be shown is the behaviour of the grid under different loads. By load is meant:

$$LOAD_{Grid} = \frac{Numberofproductsinthegrid}{Numberofnodes} \times 100\%$$

In Figure 4, a grid is shown that is not fully connected. The grid has five equiplets (denoted by the extra square connected to the node), one entry-point at the top left corner and two exit points at the right side (top and bottom node of the group of three nodes). We used this grid to simulate the production of 20 products. In this case we have four times a similar product, thus making one specific product five times. The test was run with several different loads of the grid. The results of the test

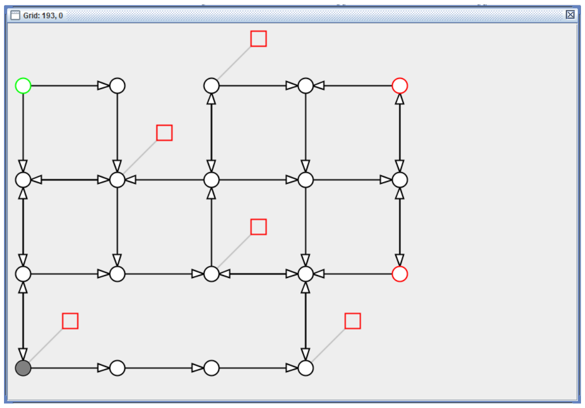


Figure 4. Example grid setup

are shown in Table I. The top row shows the load and the other numbers are the time ticks for a product to complete. When the production is not possible due to deadlock in the overcrowded grid, this is denoted by $DEAD$.

TABLE I. INCREASING GRID LOAD

10	25	50	75	90	100
114	130	128	143	DEAD	DEAD
114	130	161	256	DEAD	DEAD
114	168	202	398	DEAD	DEAD
114	168	237	404	DEAD	DEAD
113	174	279	446	DEAD	DEAD
120	174	296	664	DEAD	DEAD
121	169	317	672	DEAD	DEAD
121	200	313	683	DEAD	DEAD
121	167	367	681	DEAD	DEAD
120	174	284	723	DEAD	DEAD
143	192	347	717	DEAD	DEAD
143	208	328	777	DEAD	DEAD
143	183	302	775	DEAD	DEAD
142	181	376	771	DEAD	DEAD
143	184	357	712	DEAD	DEAD
190	213	362	595	DEAD	DEAD
190	200	370	488	DEAD	DEAD
190	215	315	481	DEAD	DEAD
190	232	335	472	DEAD	DEAD
189	256	306	251	DEAD	DEAD

Table II is partly generated from Table I and shows the average production time for all products under a certain load. The load is shown in the first row, the average production time in the second row. The last row shows the total production time for all products. This is actually the total time of the simulation.

TABLE II. CALCULATED VALUES FROM THE SIMULATION

10	25	50	75	90	100
149	196	315	585	0	0
2879	1064	824	936	0	0

The data from the second row (average production time) in Table II are plotted in Figure 5 and Figure 6 shows the total production time (the last row in Table II). As might be expected, the average time increases when the grid is working under a heavy load. A load of 75% is still feasible. The total production time of all products will at first decrease, because a higher load means also more parallelism in the production. However, the total production time for a 75% load is higher than the time for a 50% load. This is due to the crowded grid traffic and the availability of equiplets that are working under a heavy load in the 75% load situation.

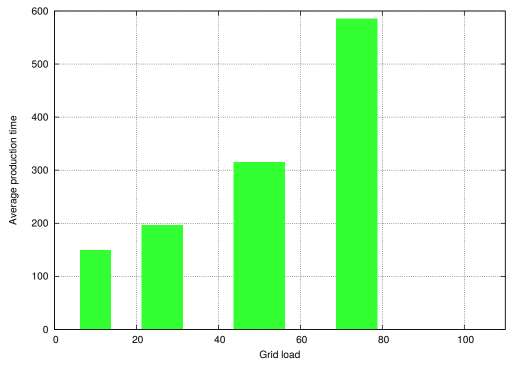


Figure 5. Average time for all products

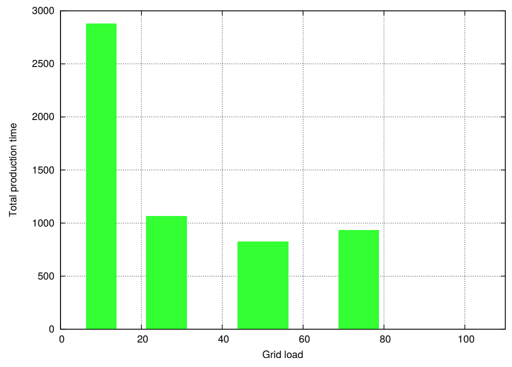


Figure 6. Total time for all products

By exporting the data to an open standard spreadsheet format, spreadsheet tools can be used to generate graphs or calculate additional data. An example of a graph is shown in Figure 7. A nice way to show the busiest node in a certain simulation. A third result shows the effect of making a modification in the path finding method. Observing the simulation, it turned out that a production platform was moving around an equiplet while the equiplet was busy with another product. If another equiplet with the same production step was available, it would be better to head for that equiplet. This was implemented and the results are shown in Table III. Three types of grids are used. They have the same paths and number of nodes, however, type A uses unidirectional paths, type B unidirectional vertical paths and bidirectional horizontal paths, while type C is using bidirectional paths. The static approach shows the time for the situation where moving to an alternative equiplet is not supported, while dynamic supports this option. In the last column the percentage of decrease in production time is shown.

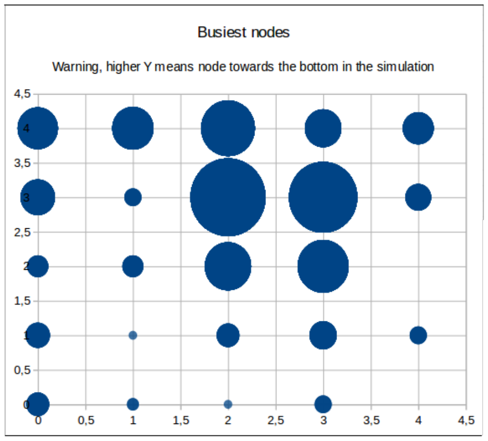


Figure 7. Simulation results showing busiest nodes

TABLE III. CALCULATED VALUES FROM THE SIMULATION

Type	Load	Static	Dynamic	Diff.
A	25%	5473	4302	-26%
A	50%	4425	3542	-20%
A	75%	4237	3604	-15%
B	25%	5023	3288	-35%
B	50%	3676	2933	-20%
B	75%	2550	2942	-17%
C	25%	4796	3290	-31%
C	50%	3374	2929	-13%
C	75%	3297	2938	-11%

Another important result is that our simulation proves that the path finding approach works well in our production system. However, under heavy loads the system will block as shown in Table I. This means that a way finding an architecture for implementing this in the production multiagent system is the next challenge. The next section will discuss this issue.

VI. From Simulation to an Implementation Architecture

The simulation was a tool to study the transport system and might play a role in the implementation architecture. In real life, the following situations should be taken care of:

- it is a concurrent system, so there should be a solution for the concurrency problem.
- a moving production platform could fail and block a path in the graph;
- a production step might take longer or shorter than predicted;
- a production unit might fail or become unavailable.

This means that the graph containing the paths for the transport robot will change in time and that the system should be prepared for the unexpected. In our first architecture proposal, only the first item mentioned is covered.

A. *Pure autonomous agents based architecture*

A way to mimic the situation of the simulation and thus overcoming concurrency problems might be a token passing system where the transport is based on timeslots (comparable with the clock ticks in the simulation). A timeslot is the time needed to reach a nearby node in the grid. An overview or list of active product agents should be available. The situation of the grid G at the beginning of the timeslot having N active product agents can be described by:

$$G(p1(t), p2(t)...pN(t)$$

Where $p1(t)$ is the position of the product agent $p1$ at time t, $p2(t)$ the position at time t of product agent $p2$ and so on. At the start the token is given to agent $p1$ that calculates its path according to the weighted path algorithm described in this paper. This will generate a new state for the grid, given by:

$$G(p1(t+1), p2(t)...pN(t)$$

Now the token is passed to agent $p2$ that will calculate its path based on this new state and so on until all N agents have a path and the new grid state will be:

$$G(p1(t+1), p2(t+1)...pN(t+1)$$

This concept can be implemented in a multiagent system by sharing the state of the grid G on a blackboard. Every agent is only interested in a small part of this information and will update only the state of two nodes, the node that becomes free and the node it will occupy at $t+1$. The overhead of communication in the distributed system will be small. There are also some disadvantages involved. There are no concepts included to overcome some of the situations mentioned in the beginning of this section. A token passing system is also vulnerable to loss of token, resulting in the whole system failing. There are solutions to this problem, like letting the token passing agents check the agent it will send the token to and a token timer to check if the token passing continues, but this requires extra overhead and complexity of the system. So an alternative solution should be investigated.

B. *Logically centralized control*

The concepts of autonomous agents seems to fit in the pure academic view on multiagent systems, but in our situation it might also be a pitfall as described in Wooldridge [9]. A central control of the transport system might be a suitable concept, but central control could become a single point of failure. A proper solution was found in the latest developments in the realisation of complex computer networks and is known by the term *Software Defined Networks* (SDN) [10].

1) Concepts used in Software Defined Networks: The concepts that are used in our proposal are inspired by the concepts of software defined networks. This paragraph will explain in a nutshell these concepts that have to do with the network infrastructure used in complex computer networks as used by Internet Service Providers and Content Providers. The core of the network is based on routers that receive packets from source hosts and forward it to other routers to deliver it to the destination hosts. This is the situation as shown in Figure 8. In the classic situation, all routers had the capability to compute the output the received packet should be forwarded to, based on the destination address in the packet and the information in the routing table. The routing table is built by the cooperation of routers, sending information to each other by a routing protocol like Open Shortest Path First (OSPF) or Border Gateway Protocol (BGP). The actual situation is that a router consists of two parts: a part that forwards packets from certain inputs to certain outputs and a part that is responsible for maintaining and building the routing tables based on information received from other routers. One failing router can spoil the system

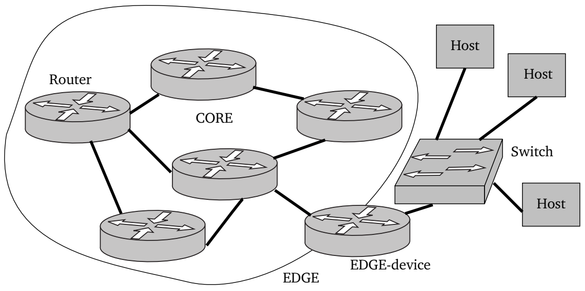


Figure 8. network core with routers

by corrupting the routing tables of other routers. All routers are also involved with two separate tasks being complex path finding algorithms as well as fast forwarding.

In the software defined network approach, routers are not involved with setting up the routing tables. They receive these tables from a server that computes the routes for them. In the SDN approach a router plays its primary role by forwarding packets based on a table containing pattern action combinations. This routing table (actually called a flow table in SDN) is not built by the router itself but by a logically centralised system that functions as the network operating system. Normally, the system is called the SDN controller. This logically centralised system can receive event messages from the routers (like the status and speed of the links it is connected to) and send messages to the routers. On the other hand as shown in Figure 9, it can also communicate with other servers to implement things like access control, routing computation and so on [11].

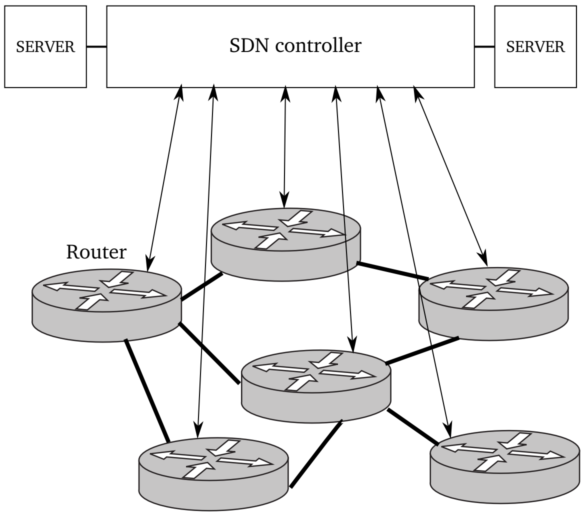


Figure 9. network with SDN controller

So summarized, one can say that there is an "unbundling" of network functionality. The result of this unbundling is that the routers are less complicated as a system and that the behaviour of the network can be easily controlled and changed by the software in the SDN controller and its related servers. Though the name suggests that this SDN controller is a single server, in practice, to prevent a single point of failure, it will be implemented as a distributed server with fail-over and high availability capabilities.

2) Using the SDN concept: The lesson learned from the previous paragraph is that it might be a good solution to simplify the agent controlled robot platform and to introduce a traffic control agent or system that is logically centralized like the SDN controller. Of course a moving robot platform is not a router, but there are also similarities. A moving production platform is in the field and like a router can explore its direct neighbourhood. This information can be sent to the traffic agent, that can update its view on the production grid as a whole and inform other platforms if they need this information to reach their next destination. The advantages of this approach are quite similar to the advantages of software defined networking being:

- All information about the traffic in the grid is available at a central place.
- Easy maintenance and control is possible. If a change in path planning is needed, only the traffic agent is involved.
- Simplification of the software on board of the moving platforms.
- No direct communication between the moving platforms.
- Computing power to solve the routing is not needed on board of every moving platform, but can be done by a special server of set of servers.

Considering the fact that the approach of a multiagent-based system is still adequate, the roles of the agents and their communication should be specified. The traffic agent knows the status of the grid. That means, the available paths, the position of the equiplets and the status of the equiplets as well as the status of all moving production platforms. Based on this information, it will guide every production platform to its next destination. The knowledge about the status of the grid is kept up-to-date by information received from the field where the moving platforms and the equiplets live. The situation is depicted in Figure 10. The traffic agent will not plan the whole path for the production, but only the path between two production steps. This is done to prevent a roll-back of plans already made, if a production step takes longer or shorter than expected. The step by step planning is also used in the simulation described before. The transport agent is the software entity that lives in the production platform and its goal is to bring the product from equiplet to equiplet according to the production steps needed. To meet its goal this agent needs to know the position of the equiplets in the grid and a path to reach the next equiplet in the set of equiplets to be visited. This information will be received from the traffic agent. Events that will generate a message from transport agent to the traffic agent are:

- entering the grid at a certain entering node
- change of edge in the grid
- starting a production step at a certain equiplet
- completion of a production step
- failure of the platform
- failure to enter a certain path (an edge in the graph) because of an obstacle

Summarized: the product agent has the list of steps and will build a list of equiplets to be visited. The product agent

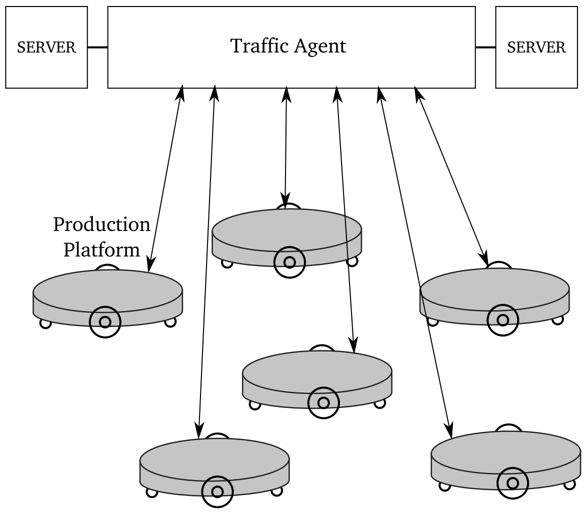


Figure 10. Traffic agent and transport platforms

hands over the list to the traffic agent. The traffic agent will allocate a production platform and guide it along the equiplets to be visited. To accomplish this, the traffic agent will tell the platform (i.e. the traffic agent in the platform) where to move to, while the platform itself is informing the traffic agent about the actual situation of its neighbourhood in the grid.

VII. CONCLUSION

An important conclusion from the simulation was, that a change in path finding could result in a significant improvement of the working of the production grid. In practice with autonomous path finding software in all platforms, this would mean that all software in the production platforms should be replaced. From SDN was learned that the moving platforms could contain a simpler type of software and the path finding could be done remotely by a traffic agent and sent to the platform. The platforms could send significant information to the traffic agent. This way, the traffic agent has an accurate view on the status of the grid at a certain moment and can use this status to generate paths for the moving platforms in the grid. The simulation system developed so far can be used to implement the path finding in the traffic agent controlled production system. In that case, calculations for different production approaches can be simulated resulting in the selection of a path planning possibility with the best result for the production as a whole.

Future work will be to implement the architecture as proposed in this paper.

REFERENCES

[1] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, “How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective,” International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, vol. 8, no. 1, 2014, pp. 37–44.

[2] R. Rajkumar, I. Lee, Insup, S. L., and J. Stankovic, “Cyber-physical systems: The next computing revolution,” Proceedings of the 47th Design Automation Conference (DAC), Anaheim, California, 2010, pp. 731–736.

[3] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, “Decentralized autonomous-agent-based infrastructure for agile multiparallel manufacturing,” Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2011) Kobe, Japan, 2011, pp. 281–288.

[4] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, “Reconfigurable manufacturing systems: the state of the art,” International Journal of Production Research, vol. 46, no. 4, 2008, pp. 599–620.

[5] L. v. Moergestel et al., “A simulation model for transport in a grid-based manufacturing system,” The Third International Conference on Intelligent Systems and Applications (Intelli 2014), Seville, Spain, 2014, pp. 1–7.

[6] M. Wooldridge, An Introduction to MultiAgent Systems, Second Edition. Sussex, UK: Wiley, 2009.

[7] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, “Implementation of manufacturing as a service: A pull-driven agent-based manufacturing grid,” Proceedings of the 11th International Conference on ICT in Education, Research and Industrial Applications (ICTERI 2015), Lviv, Ukraine, 2015, pp. 172–187.

[8] L. v. Moergestel et al., “A multiagent-based agile work distribution system,” Proceedings of the Intelligent Agent Technology (IAT 2013), 2013, pp. 293–298.

[9] M. Wooldridge and N. N. R. Jennings, “Software engineering with agents: Pitfalls and pratfalls,” IEEE Internet Computing, May/June 1999, pp.175–196.

[10] J. Kurose and K. Ross, Computer Networking, A Top-Down Approach, 7th ed., ISBN 978-1-292-15359-9. Pearson, 2017.

[11] D. e. Kreutz, “Software-defined networking: A comprehensive survey,” Proceedings of the IEEE, vol. 103, no. 1, 2015, pp. 14–76.

[12] M. Paolucci and R. Sacile, Agent-based manufacturing and control systems : new agile manufacturing solutions for achieving peak performance. Boca Raton, Fla.: CRC Press, 2005.

[13] E. Montaldo, R. Sacile, M. Coccoli, M. Paolucci, and A. Boccalatte, “Agent-based enhanced workflow in manufacturing information systems: the makeit approach,” J. Computing Inf. Technol., vol. 10, no. 4, 2002, pp. 303–316.

[14] J. Kletti, Manufacturing Execution System - MES. Berlin Heidelberg: Springer-Verlag, 2007.

[15] S. Bussmann, N. Jennings, and M. Wooldridge, Multiagent Systems for Manufacturing Control. Berlin Heidelberg: Springer-Verlag, 2004.

[16] N. Jennings and S. Bussmann, “Agent-based control system,” IEEE Control Systems Magazine, vol. 23, no. 3, 2003, pp. 61–74.

[17] D. Ouelhadj, C. Hanachi, and B. Bouzouia, “Multi-agent architecture for distributed monitoring in flexible manufacturing systems (fms),” ICRA 2000 proceedings, 2000, pp. 2416–2421.

[18] W. Xiang and H. Lee, “Ant colony intelligence in multi-agent dynamic manafacturing scheduling,” Engineering Applications of Artificial Intelligence, vol. 16, no. 4, 2008, pp. 335–348.

[19] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, “Embedded autonomous agents in products supporting repair and recycling,” Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2013) Mexico City, 2013, pp. 67–74.

[20] E. Puik and L. v. Moergestel, “Agile multi-parallel micro manufacturing using a grid of equiplets,” Proceedings of the International Precision Assembly Seminar (IPAS 2010), 2010, pp. 271–282.