2022 Future Computing Summer Internship Project: (Creating examples of livelock for the Structural Simulation Toolkit Discrete Event Simulator)

Melissa Jost*, AnotherFirst AnotherLast†

July 7, 2022

Abstract

The focus of this project was the current lack of examples looking into various distributed system design problems, and our ability to detect these issues. In order to address this, our goal within this repository was to take the issue of livelock, simulate it within SST, and come up with a set of metrics to help us analyze other systems to detect these issues. With regards to livelock, we decided to simulate the dining philosopher's problem, and use different iterations of this problem to come up with a set of metrics to decide once livelock had occurred. While coming up with the simulation for this problem was relatively straight-forward, there is still more work to be done in regards to solidifying and expanding on the metrics we chose to define this problem within distributed systems.

1 Project Description

The challenge that is being addressed by this work is that while we have countless examples of what issues existed within distributed systems, as well as various ways to avoid including them within our systems, there is a lack of work that exists with regards to quantifying metrics to detect these issues. In addition, by creating these simulations, we provide additional examples of livelock, which can be helpful to both those working to better understand these issues, as well as those trying to understand how to use SST as a whole.

2 Motivation

The users of this work include two main groups of people: those working on developments for distributed systems, as well as those interested in learning how to use SST. For those working on developments for distributed systems, this is useful because if we are able to develop a truly useful set of metrics for detecting these various design problems, even if in simple simulations, it could potentially save a lot of time in finding them in real systems in the future. For those learning SST, these simulations help give more, simple examples of how to use the toolkit.

3 Prior work

See [**?**] for prior work in this area.

4 How to do the thing

For the simulation, we are able to run it with one laptop, with a total of five philosophers. This is how a majority of our analysis took place. However, by including additional links and components, we could easily scale this simulation to model a bigger system.

The software is available on (https://github.com/lpsmodsim/2022HPCSummer-Livelock)

In order to run it, you simply need to run the Makefile.

5 Result

Throughout the process of simulating the dining philosopher's problem, we were able to come up with a couple different instances of the problem, as well as note down certain parameters that seemed to make livelock more or less likely to occur. When looking into what components made the most sense to model, we had different iterations: one that focused on modelling philosopher components alongside a central dining table component, as well as philosophers alongside individual chopstick components. This allowed us to explore the ways in which different simulations would potentially affect the likelihood, as well as the ways in which livelock would occur. From here, we set a list of parameters in our simulations to see what would induce livelock. These parameters included:

- thinkingDuration: The amount of time a philosopher spends thinking after placing down his chopsticks, whether that was after he finished eating, or while trying to allow another philosopher to eat
- waitingDuration: The frequency at which a philosopher checks the state of his hands, and sees whether or not he needs to place down his chopsticks to allow another philosopher to eat
- eatingDuration: The amount of time a philosopher spends holding both chopsticks when eating

One important thing to note about these parameters is that generally speaking, the more randomization you introduce into the timing, the less likely you are likely to experience livelock, with livelock being defined as the state in which none of the philosophers are ever able to enter their eating states. In addition, you generally want your waitingDuration to be at most, equal to the thinkingDuration if you want to experience livelock. Lastly, a longer eatingDuration will lead to an encourangmenet of livelock, assuming your simulation ever lets you enter the eating condition in the first place.

In terms of the metrics we coined in order to detect livelock, our current findings is that they are heavily reliant on the specific problem at hand. For example, much of the work in this area tends to agree that livelock is dependent on an existance of some infinte loop that contains changing states, as well as the lack of progress in the system. In addition to this, many sources tend to agree that the definition of progress, as well as the definition of infinite is something that is left up to the programmer to decide, as it varies per problem. We can define those criteria in the dining philosophers problem by looking towards how much time one spends switching between thinking and being hungry, and how often one eats. Even so, concretely defining how long we should let the simulation run until we finally declare that the loop is infinite is a bit vague.

6 Future Work

Overall, the lack of concrete research in regards to both livelock as a whole, as well as the specifics of how to quantify livelock issues limited this project. Many papers had differing definitions on what qualified as livelock, and these various definitions were sometimes hard to translate into useful information for distributed systems, such as what would be considered an infinte loop, or what was meant by progress in the system. To further understand this problem, it would probably be most helpful to find more real examples of livelock (ex. not the hallway problem), so that we can come up with metrics that are applicable to a wider set of systems. While we can semi-confidently conclude that we have livelock in the simulations provided above, finding ways to generalize the process in which we found livelock would be useful (ex. a more general definition of progress, or clearer examples of states that don't only apply to dining philosophers).

References

[1] Donald E. Knuth (1986) *The TEX Book*, Addison-Wesley Professional.

[2] Leslie Lamport (1994) *LATEX: a document preparation system*, Addison Wesley, Massachusetts, 2nd ed.