7 Conclusion

In this thesis, a dynamic load balancing solution for data stream processing in DDS-based systems is presented, which is named *Data Processing Slice Load Balancing Solution*. The proposed solution consists of two distinct types of nodes, *Processing Nodes* and a *Load Balancer*. The Processing Nodes (PN) are homogenous nodes that perform identical processing on each item of the data stream, which is application dependent. The Load Balancer is a node responsible for monitoring the current load of all Processing Nodes, defining the actions required to redistribute the system's workload (i.e. reassignment of DPS) when an work unbalance is detected and synchronizing the actions executed by PNs to move DPSs between them. The data stream flows directly from Client Nodes, which publish data on the DDS Domain, to Processing Nodes. Thus, the Load Balancer does not behave as a dispatcher which has to choose a Processing Node for each single data published by Client Nodes, but instead just controls the amount of the total data stream that is to be delivered to each Processing Node. The Load Balancer has a module that effectively analyzes the workload at each PN and decides whether any load distribution should be done. Moreover, many load balancing algorithms may be employed on the Load Balancer.

The proposed load balancing solution fulfills all the requirements that have been listed in Chapter 1. It accords for the *dynamic* nature of the DPSLB solution as it initiates the Load Balancing Process whenever it becomes necessary, i.e. a workload unbalance is detected. The capability of plugging – and replacing – the Load Balancing Algorithm Module into the Load Balancer enables the implementation of customized load balancing algorithms, and thus gives the solution its *adaptive* nature. Furthermore, all load balancing actions are *transparent* to the application – both on the publisher side and the Processing Node – since all necessary actions are performed and coordinated by DPSLB components at the Load Balancer or Processing Nodes. *Heterogeneity* with respect to the managed resources and its load parameters is achieved through the use of MAPE-SDDL, which is able to monitor any type of resource at PNs, and a Load Balancer – depending of the Load Balancing Algorithm that is plugged into the Load Balancer – that can take into account nodes with different resource capacities. Finally, the DPSLB solution is *generic* since it is not dependent on the DDS topics defined by the application. Instead it just manages how the DDS samples (data items) are delivered to the application Processing Nodes.

The test experiments have already shown encouraging results. In particular, we could check that during the Load Balancing Process there was neither any data item loss nor duplication. It could also be noticed that the addition of new Processing Nodes effectively enhances the system´s processing capacity and does not drive to an rise of the overhead. For example, the overall throughput could be augmented from 40 DI/s for one PN, to 323 DI/s, when five PNs are used. While new Processing Nodes help to increase the throughput, more PNs reduce the RTD – or at least to reduce the growth – because the load is divided across the available PNs since the system is able to promptly process more data items in the same unit of time. Most importantly, the proposed DPSLB solution allows a stream processing system to scale in the number of PNs with acceptable overhead. The overhead introduced by the DPSLB prototype represents only 1,4% and 1,58% of the throughput and RTD, respectively, which is a low cost compared to the benefit of having a load balancing mechanism. With a data stream rate of 10 MB/s and 1.000 *Slices*, the DPSLB prototype was able to move 500 *Slices* from a single PN to a second PN in less than 500 ms (milliseconds), which it is believed to be fast enough for many stream processing application.

7.1
Future Work

Although all the proposed objectives were met, there is some future work in short term that would enrich the proposed solution. One of the most important next steps is to verify whether new versions of CoreDX DDS will remove the limitation of not being able to dynamically update filter expression parameters of Content Filtered Topic, which was faced during the development of the DPSLB prototype. If this limitation is removed, then we could go back to the original approach of using Content Filtered Topic, instead of performing the filtering of data items in the PN software layer, after delivery by the DDS layer. Another improvement could be made in regard to the Load Balancing Process, reducing its overhead: the process of sending the data item cache could be made more efficient by sending all cached data items in a single *CacheTopic* sample instead of sending one *CacheTopic* sample for each DPS. Yet another interesting enhancement of the Load Balancing Process would be adding support for the transfer of the application´s state from *Slice-giving* PNs to *Slice-taking* PNs.

In medium term, the implementation and evaluation of more complex Load Balancing Algorithms and Assignment Functions would be valuable to study how the DPSLB solution behaves in face of uneven Assignment Function and heterogeneous PN resource capacities. In particular, it is believed that collecting statistical data about the data items received in each *Slice,* and the corresponding workload associated to each Slice will certainly enable much better Load Balancing Algorithms, as these would be able to take into account the different data stream rates assigned to each *Slice,* and to decide which specific Slices, and not only how many, should be moved among the PNs.