EXPERIMENTAL EVALUATION OF CUMULATIVE LOOKAHEAD IN CONSERVATIVE PARALLEL SIMULATION

Viliam Solčány[1], **Jiří Šafařík**[2]

[1]Slovak University of Technology, Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovak Republic
[2]University of West Bohemia, Department of Computer Science and Engineering
Univerzitní 8, 306 14 Plzeň, Czech Republic

solcany@fiit.stuba.sk(Viliam Solčány)

Abstract

This work deals with performance improvement of parallel discrete event simulation. In parallel simulation, it is reasonable to focus on large scale models consisting of large number of entities. Such models are usually partitioned for parallel execution so that submodels contain multiple entities. The performance of a conservative parallel simulator is significantly influenced by the lookahead ability of the submodels. The key issue we study is how to combine the lookahead abilities of individual entities residing in a submodel thus enhancing the lookahead of the submodel. The study is conducted in the context of the synchronous time window based synchronization method. The lookahead for this method is constituted by two actions, in particular message pre-sending and future timestamp prediction. Given these actions at the level of model entities, we provide concepts and methods for their combining to get the corresponding cumulative abilities for the level of submodels. In this way the lookahead of the submodels can be enhanced and, consequently, simulation performance increased. We set up a series of experiments to evaluate the performance contribution of this approach. As benchmark models we use a variation of the parallel hold model and queuing networks. The results show that the proposed methods can significantly increase the average size of the time windows, and thus reduce the synchronization overhead.

Keywords: Parallel Simulation, Conservative Synchronization, Lookahead, Compound Submodels, Cumulative Lookahead.

Presenting Author's Biography

Viliam Solčány was born in Handlova, Slovak Republic and went to Slovak University of Technology, where he studied informatics and obtained his M. Eng. degree in 1991. He works part time as researcher at Faculty of Informatics and Information Technologies of Slovak University of Technology in Bratislava, Slovak Republic, and also at Austrian Academy of Sciences in Vienna, Austria. He is a member of ACM since 2004.

1 Introduction

Parallel discrete event simulation (PDES) is an appropriate technology when running a model sequentially takes too long to be acceptable [1]. Such situations occur when simulating complex models consisting of a large number of *entities* with high degree of detail. In PDES, the model is partitioned into *submodels* which are executed on multiple processors. The *simulation processor* denotes an abstraction including hardware and software needed for interpretation of a submodel. Simulation processors communicate via *timestamped event messages* and need to be synchronized in order to avoid *causality errors*. To tackle this issue, a wide variety of *synchronization* approaches has been developed [1]. In our work, we use the *conservative* method based on time windows [2]. The method is briefly described in section 2. As in any conservative method, it is supposed that the entities have the ability known as *lookahead* [1].

In large models, which are of primary interest in parallel simulation, the number of entities can easily exceed the number of available processors. In that case, submodels consist of multiple entities. The execution within a submodel runs sequentially, thus submodels are the units which need to be synchronized and the lookahead of which is needed. In such *compound submodels*, each or some of the entities possess some amount of lookahead. Then a natural question arises about the possibility of combining these amounts into a larger one for the whole submodel. In our previous work we elaborated this issue in detail [3, 4]. The first experimental results appeared in [5]. They indicated a good potential of the approach to improve simulation performance. The purpose of this paper is to provide a more deep experimental evaluation of the methods for another model and an extended parametric space. Moreover, another performance metric has been added to get better insight into the simulation behavior.

The paper is structured as follows. Sec. 2 provides a brief description of the synchronization method used. In Sec. 3 we review the issues of cumulative lookahead. Sec. 4 describes the models used in our experiments, and Sec. 5 is devoted to the experiments themselves. Finally, conclusions are given in Sec. 6.

2 The synchronization method

For submodel synchronization, we use the conservative method based on synchronous time windows [2]. The method assumes the ability of entities to pre-send event messages. Further, it is assumed that the entities can predict at any point in simulation time the timestamp of the next output message. More precisely, an entity c can compute the *conditional* timestamp $\delta_c(t)$ of the next message output message (or a lower bound on it). The timestamp is conditional, because it is computed under the assumption that c will receive no new input messages. Message pre-sending and conditional future timestamp prediction are the two actions which constitute the lookahead necessary for the synchronization protocol.

For now, suppose that each entity is assigned to a distinct simulation processor. The synchronization protocol works as follows: assume that all events with timestamps less than t have already been processed and that the processors are globally synchronized at time t. Each entity c provides its future conditional timestamp $\delta_c(t)$. Then the processors cooperatively compute the minimum $\delta(t) = min_{\forall c}\{\delta_c(t)\}$. The interval $\langle t, \delta(t))$ defines the time window within which all simulation processors may execute events independently. Such construction of the time window ensures that there will be no message with timestamp within the window [2].

3 Cumulative lookahead

Model entities are the basic building blocks for model construction. It is desirable to allow re-use of entities, including their lookahead properties. In the case of compound submodels, the lookahead of a submodel can be derived from the lookahead of the entities in several ways. For conditional timestamp prediction, there are two common approaches. One of them is to compute global minimum of $\delta(c)$ values of all entities of a submodel. Since every entity of the model is directly involved in time window computation in this case, we refer to this method as the "every entity" (EE) method.

The other method considers only inter-submodel messages. Entities which can send such messages provide their lower bound on timestamp increment. This is the minimum delay a message experiences when passing through the entity. The global minimum of these values provides the time window size which is constant throughout the simulation. The window size must be strictly positive, so must be the minimum delays supplied by the entities. We call this approach of time window computation the "minimum delay" (MD) method.

Neither of the two methods exploits the presence of multiple entities in a submodel. Such method has been proposed in [3], and is reviewed next.

3.1 Cumulative timestamp prediction

Event processing within a submodel runs sequentially, and submodels are the units which need to be synchronized. Thus the task for a compound submodel is to provide the conditional timestamp of next message, let us denote it $\delta_{SM}(t)$. In [3] we developed an algorithm for computing the $\delta_{SM}(t)$ value, called *DeltaSM* algorithm.

The essence of the algorithm is to compute for each entity c the unconditional earliest timestamp $\gamma_c(t)$ it will send after time t. It is computed as the minimum of two cases. First, if c does not receive new messages, its earliest output timestamp will be $\delta_c(t)$. Second, if c will receive an input message with timestamp $\gamma_{PRED}(t)$, and the message will be delayed by c for time interval $\sigma_c(\gamma_{PRED}(t))$, the next output timestamp will be $\gamma_{PRED}(t) + \sigma_c(\gamma_{PRED}(t))$. Then

$$\gamma_c(t) = min\left[\delta_c(t), \gamma_{PRED}(t) + \sigma_c(\gamma_{PRED}(t))\right] \quad (1)$$

The timestamp $\gamma_{PRED}(t)$ coming from the predecessor is computed recursively. The recursion has its bottom because of the assumption that the submodel receives no input messages. Thus the timestamps coming from other submodels can be considered infinite. The desired $\delta_{SM}(t)$ value is determined as the minimum of timestamps sent out of the submodel.

3.2 Cumulative message pre-sending

From a message flow viewpoint, model entities fall into two categories. The *self-initiating* ones can themselves initiate simulation activity, while in the *message-initiating* the simulation activity is caused by incoming messages. An example of the former is a source of transactions in a queuing network, the latter is typically represented by a node serving transactions.

Consider a submodel consisting of message-initiating entities where a message travels through the entities. Its timestamp gradually increases as it is delayed by the entities. If the timestamp exceeds the current time window, the message will be further forwarded within the next window. However, in such case it will cut the δ value of the entity where it stops during the present time window. Such message then restricts the size of the next time window. The cumulative message pre-sending attempts to avoid such behavior by "pushing" the message out of the submodel in the current time window.

It is achieved by *immediate message forwarding* (IMF) meaning that an event message is forwarded as early as it gets scheduled. This is feasible for certain entities when constructed in a special way. In addition, to enable IMF, messages sent to the entity on each link must follow the non-decreasing timestamp order. There is an exception for some entities when this condition can be ignored. In that case, the simulation does not yield the same results as without IMF, but the results are still statistically correct. More detail about the concepts and issues involved in cumulative message pre-sending can be found in [4] and [5].

3.3 Entities with non-zero minimum delay

If an entity c forwards messages immediately, its conditional future timestamp, i.e. its $\delta_c(t)$ value is always infinite. If, in addition, the entity supplies the lower bound on message delay instead of its actual $\sigma_c(t)$ value, then both $\delta_c(t)$ and $\sigma_c(t)$ do not vary with simulation time t. These values are used as inputs of the DeltaSM algorithm, thus the window size computed by the algorithm is constant. The advantage is that the algorithm needs to run just once, instead of a new run for each time window. The disadvantage is that the time windows are smaller. Exact details about this technique are given in [3].

4 Experimental models

To evaluate the effect of cumulative lookahead, we use two kinds of models, namely *queuing networks* and the *parallel hold* model. The former ones are similar to closed queuing networks (QNets) used to study the impact of lookahead itself, e.g. in [6]. The latter, parallel hold (*PHold*, for short) model has been first introduced in [7], and is included as a standard benchmark and test model in several parallel simulators, e.g. GTW [1], or DSIM [8].Both of the models consist of interconnected nodes. The nodes do not generate messages, they are message-initiating. The models are homogeneous in the sense that each submodel contains the same number of identical nodes, and the submodels are of the same structure. This allows us to exclude the influence of load balancing on measured results.

Further details about the models are given in the following subsections.

4.1 Queuing Networks

A node of the QNet model consists of a tandem of queue entities. We use two configurations of the nodes, in particular a tandem of two entities (c_1 and c_2), and a tandem of four of them (c_1 through c_4). The former is referred to as *QNet2*, and the latter as *QNet4* model. In both cases, entity c_1 ensures that messages arrive to c_2 in timestamp order. Then IMF can take place in c_2 as mentioned in Sec. 3.2. In the case of QNet4, entities c_3 and c_4 are similar to c_2 with respect to lookahead and IMF. Whether or not IMF actually takes place in c_2, c_3, and c_4 is controlled by model parameters, see Sec. 5.1. A certain number of transaction (job) arrivals are initially scheduled for each node. Varying the number can control the queue load.

4.2 The Parallel Hold Model

In the PHold model, there is just one event type. Processing that event by a node results in scheduling a single new event for another node selected according to some rules. The timestamp of the new event is calculated as the timestamp of the original event plus a certain increment. The rules for selecting the target node, and the probabilistic distribution of the timestamp increment are among the parameters of the model. As a consequence of this event processing, there is a constant number M of scheduled events in the entire model. The event occurrences of the entire simulation can be decomposed into M threads based on their causal relationships. The simulation run can be viewed as the M event threads moving through the model.

A PHold node in our experiments contains a tandem of two entities c_1 and c_2. Therefore, similarly to QNet2, it will be referred to as PHold2. Entity c_1 processes event messages coming to the node and forwards them to c_2. The timestamp of a forwarded message is the result of adding an increment to the timestamp of the incoming message. Similarly, c_2 adds an increment to the message and forwards it to some other node.

5 Performance Measurements

The most common performance measure in PDES experimental studies is the overall simulation speedup. Unlike this, our primary performance measure in this study is the *average time window size* (ATWS). Beside that, we also measure the *average number of events per window* (ANEW). The reason is that the cumulative lookahead is aimed at increasing the time window size, thus comparing the ATWS and ANEW values obtained with and without the cumulative lookahead gives us the most appropriate picture about the contribution of the cumulative lookahead itself. Unlike speedup, the results in terms of ATWS and ANEW are isolated from factors such as characteristics of the underlying parallel machine, algorithms employed in the simulator (e.g. for global min reduction), and efficiency of their implementation. Such performance measures can be found in the literature, e.g. [9].

5.1 Model Parameters

All experimental models contain 64 nodes and are partitioned into 8 submodels. The configurable parameters of the models are as follows:

1. Model topology (TY). To achieve various fan-in/fan-out of the nodes, four different topologies are used. The fan-in/fan-out is 2 for a bidirectional `ring`, 4 for a 2D `mesh`, k for a $N = 2^k$ node `hypercube`, and $N - 1$ for an N node `fully connected` network.
2. Node type (NT). In order to allow an unambiguous parametric description of each of the simulation experiments, we include this parameter into the set of parameters. Possible values of the parameter are `QNet2`, `PHold2`, and `QNet4`.
3. Time distribution (TD). This is the probabilistic distribution of service time in QNet2 or timestamp increment in PHold2. Three distributions with the same mean but different variance are used, in particular `Const(1.0)`, `0.5+Exp(0.5)`, and `Exp(1.0)`.
4. Time distribution ratio (TR). The above time distribution parameter TD expresses the service time/message delay of a node in total. This time is divided between the two entities c_1, c_2 contained in the node. Two ways of division are used, first with ratio $TR =$ `7:3`, and then $TR =$ `3:7`. In QNet4, the ratios are `35:35:15:15` and `15:15:35:35`, but, for the sake of simplicity of the notation, we refer to them as `7:3` and `3:7`, respectively.
5. Message population (IA, standing for initial arrivals). This is the number of messages circulating in the model. It is determined by the number of message arrivals scheduled initially before the simulation begins. Three different levels are used with 1, 4, and 16 initial arrivals per node, denoted as $IA =$ `1`, `4`, and `16` respectively.
6. The lookahead level (LL). This parameter controls the message pre-sending and future timestamp prediction in the submodels. There are two non-cumulative levels, in particular `EE` and `MD` (see Sec. 3), and several cumulative levels. The `DeltaSM` level uses the cumulative timestamp prediction, but no IMF. The reason for this level is that the DeltaSM algorithm is generally applicable, but IMF is not feasible in all models. Therefore we are interested in the effect of the DeltaSM algorithm alone. In the `MDIMF` level, messages are forwarded immediately. The time window has constant width that is computed by the DeltaSM algorithm using minimum delays of entities as its input. This is the approach mentioned in Sec. 3.3. The `IMF` level in QNet2 and PHold2 means that entity c_2 of the nodes uses IMF. In QNet4, the `IMF` levels denotes immediate message forwarding in entity c_4 only. In this model, additional two levels are used, denoted as `IMF2` for the case of IMF in c_3 and c_4, and `IMF3` when c_2 through c_4 do employ IMF. In the PHold2 model, the level denoted as `IMFFRC` (forced IMF) is used for the case when all entities in a submodel are involved in IMF. This is the statistically correct approach mentioned in Sec. 3.2.

5.2 Results

We performed a measurement for every point in the parametric space defined by the model parameters. The results for QNet2 are summarized in Fig. 1 and Fig. 2, for QNet4 in Fig. 3 and Fig. 4, and for PHold2 in Fig. 5 and Fig. 6. The QNet models with time distribution `Exp(1.0)` cannot be simulated using the `MD` nor `MDIMF` time window computing method. This is because the lower bound of the distribution is zero and, consequently, the time window size is zero. For the PHold2 with this time distribution, any window constructing method leads to zero-size time window. Thus PHold2 with timestamp increment `Exp(1.0)` cannot be simulated at all.

The measurement consists of 10 replications yielding a set of 10 samples. The result of the measurement is the sample mean of the 10 values. A replication runs until each entity processes at least 20000 transactions.

A number of interesting observations can be made from the figures:

- When not considering the `IMFFRC` lookahead level, for time distribution `Const(1.0)`, both ATWS and ANEW values are essentially insensitive to the topology. For the exponential distributions, the results for ring are somewhat larger than those for other topologies. Thus, from the viewpoint of fan in/fan out, starting with value of 4, the performance measures become steady. This behavior can be observed in all models.
- Increased variance means smaller time windows in all models. This fact is known for non-cumulative lookahead, and can be observed for the cumulative one too.
- In QNet2, the delay a message experiences at an entity increases with increasing message density. The `IMF` lookahead level is able to utilize the larger delays to increase ATWS. In other words, for the `IMF` lookahead level, ATWS always increases with increasing message density. The same holds for ANEW in QNet2. Both performance measures show this behavior also in QNet4 with the maximum cumulative lookahead level

`IMF3`. In PHold2, on the other hand, the delay of a message is independent of messages the node has received recently. Therefore, ATWS is almost insensitive to message density in this model with the cumulative lookahead level `IMF`.

- The cumulative lookahead involving immediate message forwarding performs always better than any of the non-cumulative levels in all three models. This holds for the `IMF` lookahead level in QNet2 and PHold2, and also for all the three levels `IMF` through `IMF3` in QNet4. Both performance measures show this behavior.

- In two-entities-per-node models with the `IMF` lookahead level, the performance is (almost) insensitive to the time distribution ratio (`3:7` versus `7:3`). In the case of QNet4, this holds for the largest, `IMF3` level. Both performance measures show this behavior.

- The DeltaSM algorithm alone outperforms the non-cumulative methods in QNet2 if time distribution ratio is `3:7`. In PHold2, this holds only for the constant time distribution. In QNet4, on the other hand, DeltaSM in more successful. It outperforms the non-cumulative approaches almost always, and often with a significant margin.

- The `IMFFRC` level in PHold2 shows the potential of cumulative lookahead in larger scale. All entities are involved in IMF in this case. The values of ATWS are approximately proportional to the minimum average number of entities a message traverses between its entry to and departure from a submodel. In this extreme case of IMF, ATWS is insensitive to time distribution.

- In QNet4, increasing the number of entities involved in immediate forwarding of messages, i.e. moving from lookahead level `IMF` to `IMF2` and `IMF3`, leads to improvement of ATWS for non-constant distributions. For the constant time distribution, the model behaves differently for these three lookahead levels. There seems to be a saturation value of ATWS for each message density. In some cases this saturation value is reached, in others not.

- The `MDIMF` lookahead level, although involving IMF, does not lead to any improvement of time window size over `MD` in QNet2 and PHold2. For `MDIMF` in QNet4 the following can be observed. `MDIMF` significantly outperforms the non-cumulative levels. At the same time, it is in some cases outperformed by the `DeltaSM` level that does not require to forward messages immediately. Another interesting fact is that with time distribution ratio `7:3`, the `IMF` level results in smaller time windows than `MDIMF`. This is because in the former case, only the actual delay of c_4 (which is small in this case) is involved in cumulative lookahead, while in the latter case, the minimum delays of c_2 through c_4 are accumulated. Yet another observation that holds for all models is that ATWS for the `MDIMF` level does not depend on message density.

6 Conclusion and future work

We have presented an experimental study of performance of cumulative lookahead in synchronous conservative parallel simulation. As the main contribution of the paper, we have specified experimental models, and have carried out experimental simulations for a large parametric space. The results show that cumulative lookahead can bring significant performance improvement. The increase of average time window size is up to 57-fold when using IMF, and up to 16-fold with the DeltaSM algorithm. The latter value is encouraging, since the DeltaSM algorithm, in contrast to IMF, is generally applicable.

Having shown the potential of the cumulative lookahead, our future work includes the study of its applicability to several specific simulation models, such as wireless network models.

7 Acknowledgment

This work was supported by the Scientific Grant Agency of the Ministry of Education of Slovak Republic, grant No. VG 1/3103/06, and by the Ministry of Education, Youth and Sport of the Czech Republic, "University spec. research - 1311".

8 References

[1] Richard M. Fujimoto. *Parallel and Distribution Simulation Systems*. Wiley & Sons, 1999.

[2] David M. Nicol. The cost of conservative synchronization in parallel discrete event simulations. *Journal of the ACM*, 40:304–333, 1993.

[3] Viliam Solčány and Jiří Šafařík. The lookahead in a user-transparent parallel simulator. In *Proc. 16th Workshop on Parallel and Distributed Simulation*, pages 11–16, Washington D.C., USA, May 2002.

[4] Viliam Solčány and Jiří Šafařík. Cumulative lookahead in conservative parallel simulation. In *5th EUROSIM Congress on Modeling and Simulation*, Marne la Valle, Paris, France, September 2004.

[5] Viliam Solčány and Jiří Šafařík. The effect of cumulative lookahead on conservative parallel simulation performance. In *ESM 2005 European Simulation and Modeling Conference*, pages 512–519, Porto, Portugal, October 2005.

[6] Bruno Richard Preiss and Wayne Mervin Loucks. The impact of lookahead on the performance of conservative distributed simulation. In *Proc. 1990 European Multiconference—Simulation Methodologies, Languages and Architectures*, pages 204–209, Nuremberg, Germany, June 1990. Society for Computer Simulation.

[7] R. M. Fujimoto. Performance of Time Warp under synthetic workloads. In *Proc. SCS Multiconference on Distributed Simulation*, volume 22, pages 23–28, January 1990.

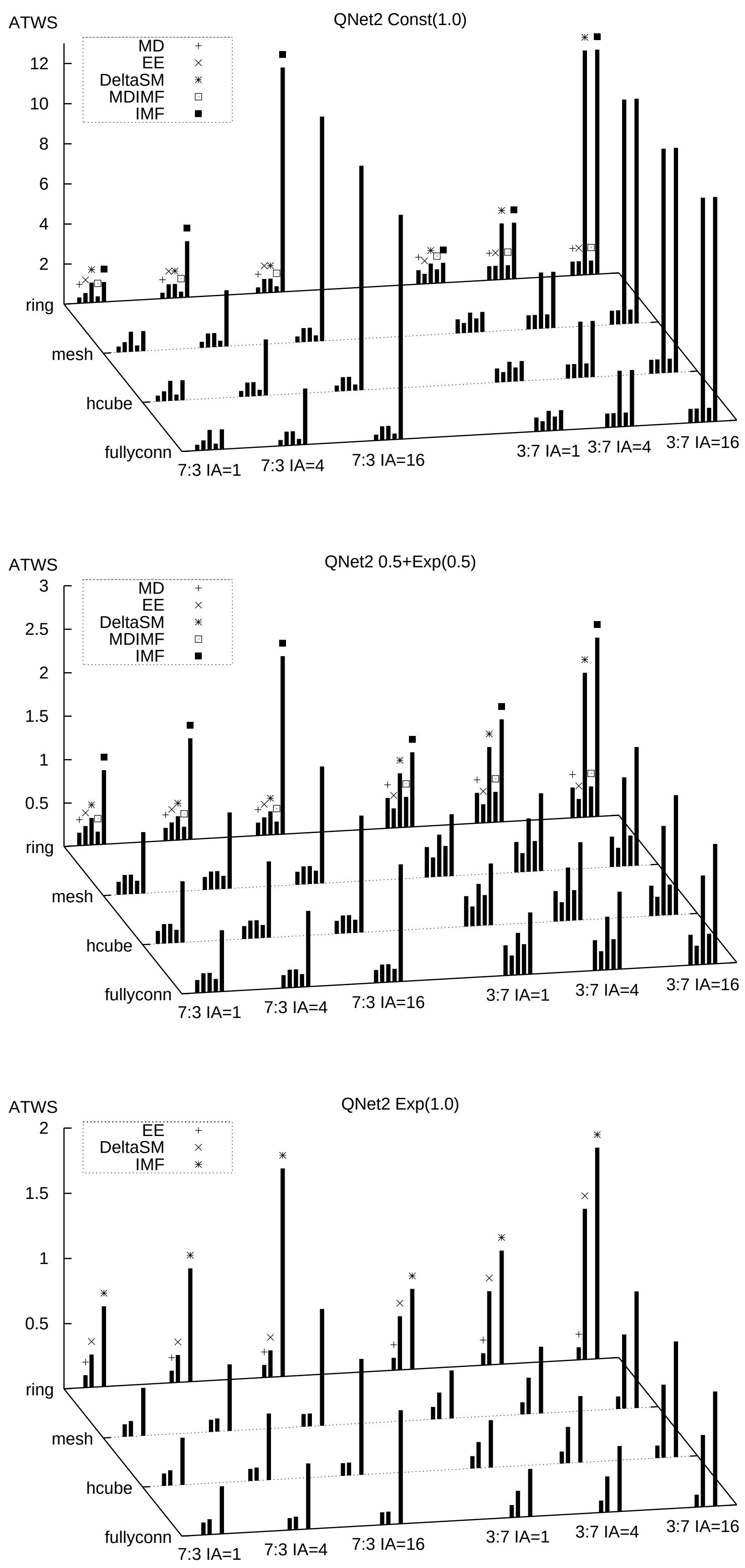
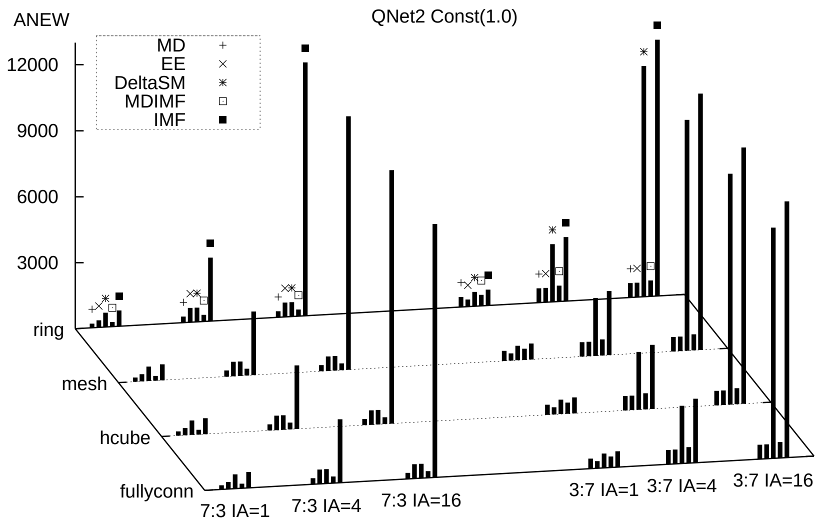
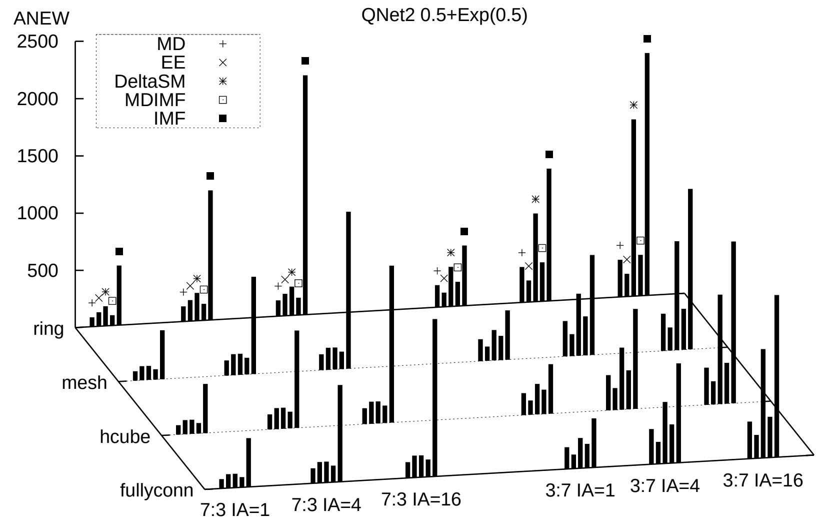


Fig. 1 ATWS for the QNet2 model





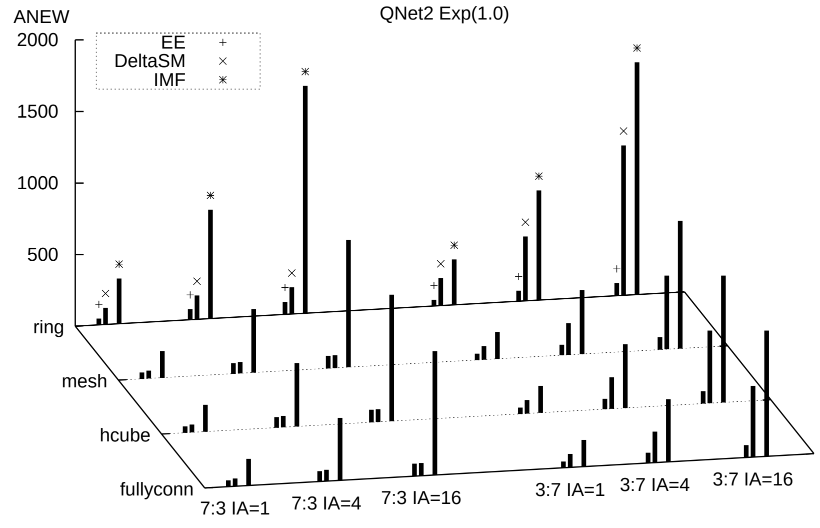
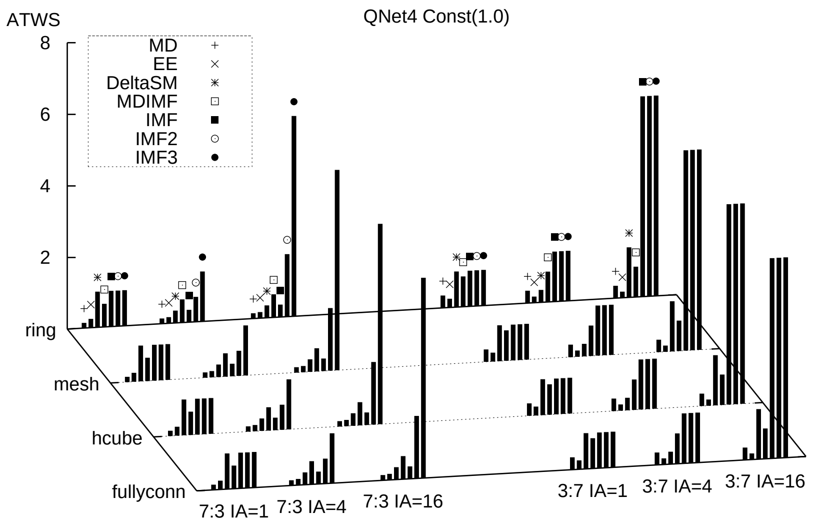
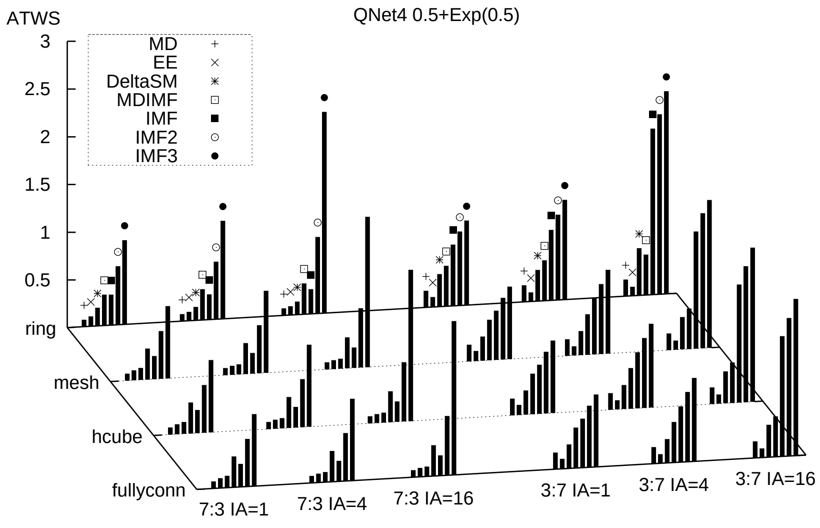


Fig. 2 ANEW for the QNet2 model





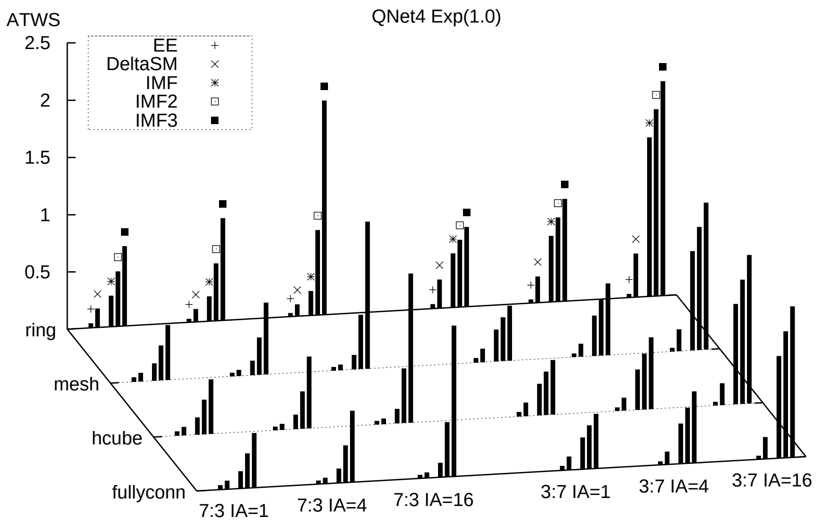
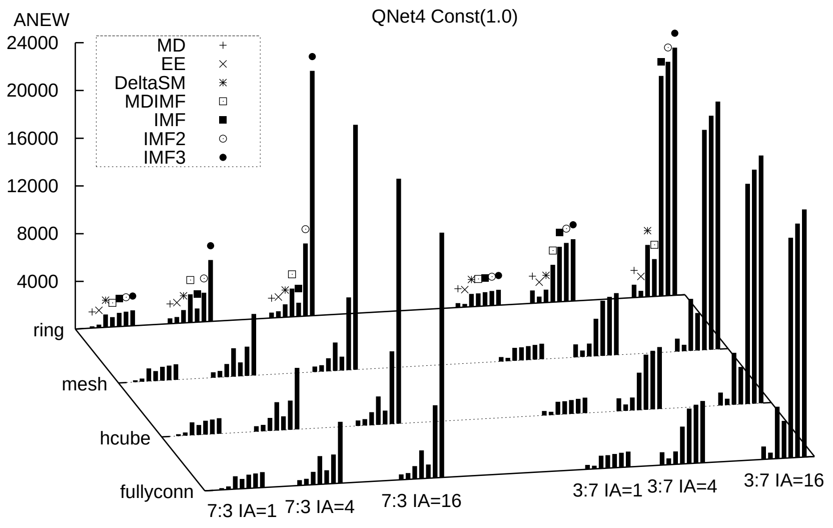
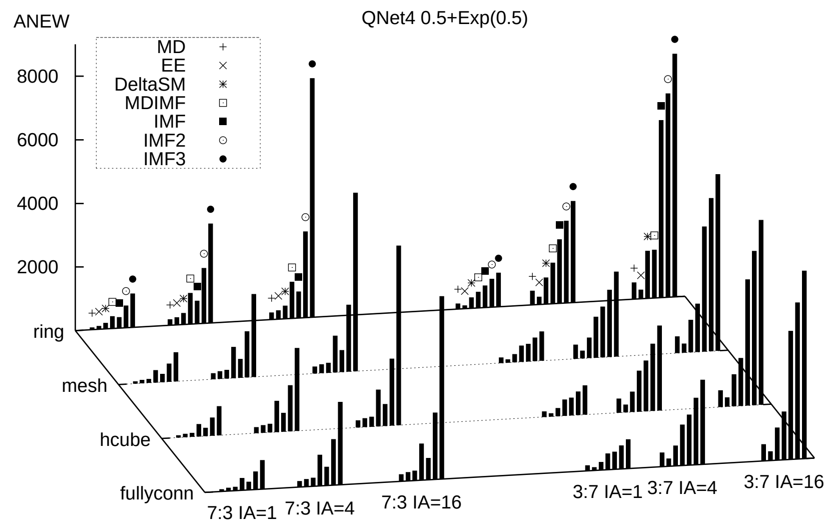


Fig. 3 ATWS for the QNet4 model





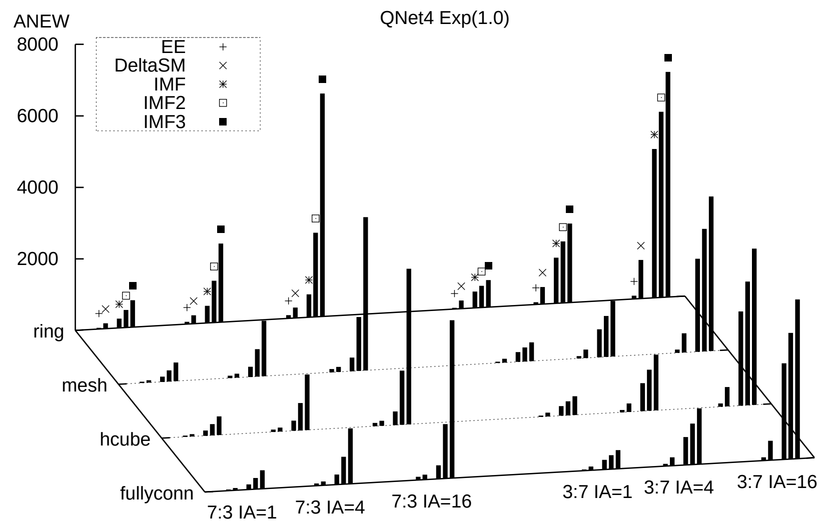
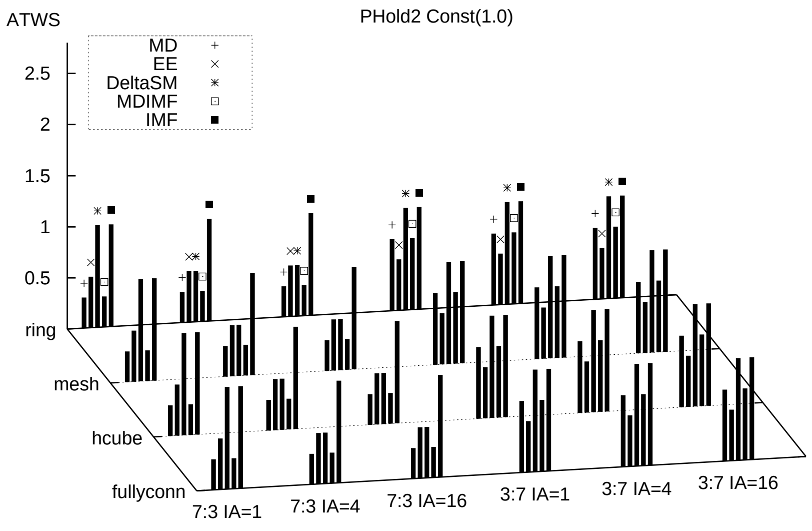
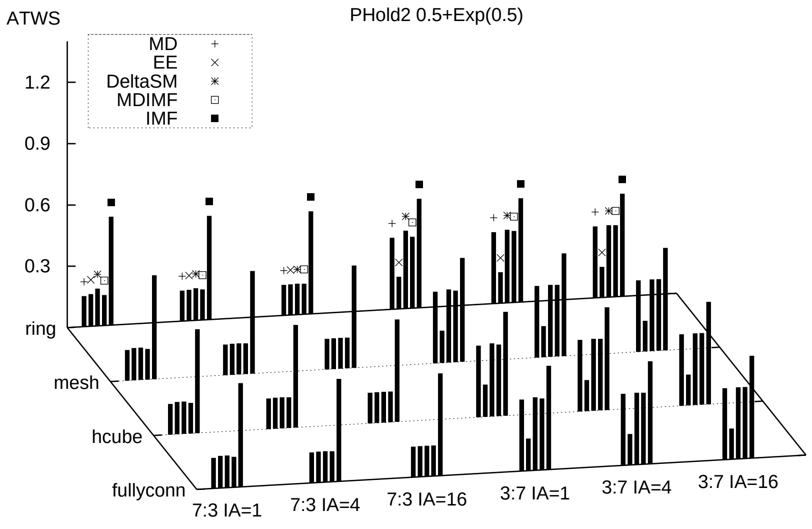


Fig. 4 ANEW for the QNet4 model





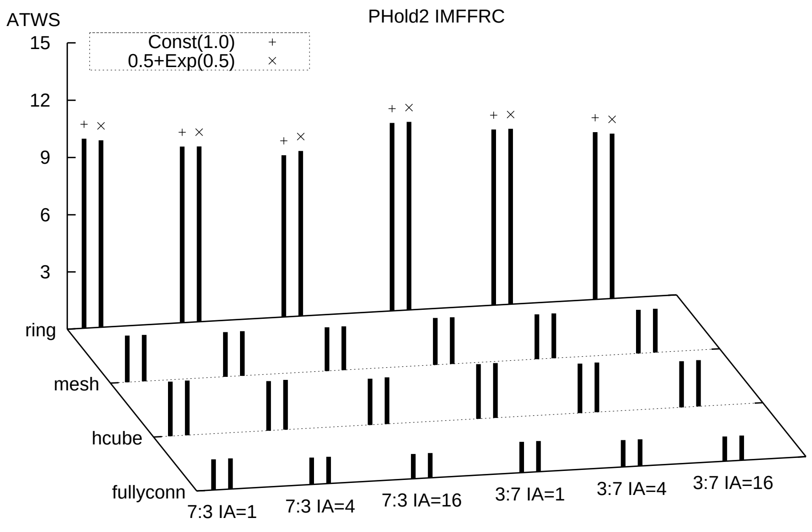
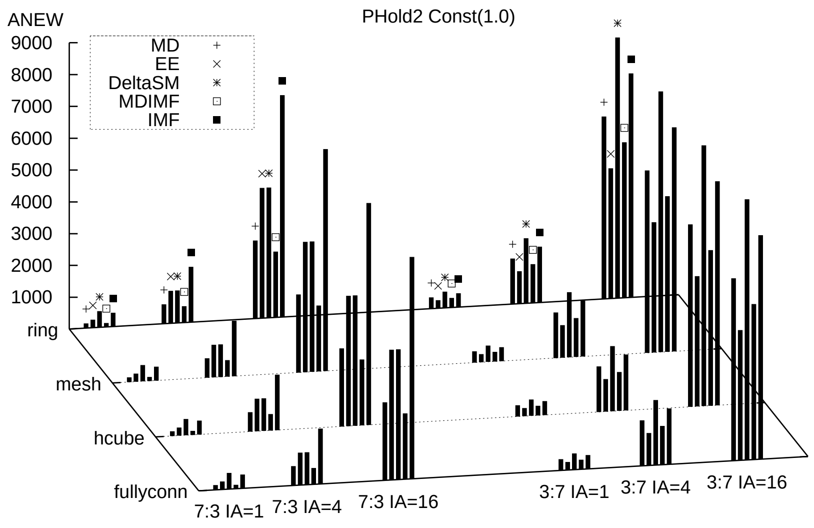
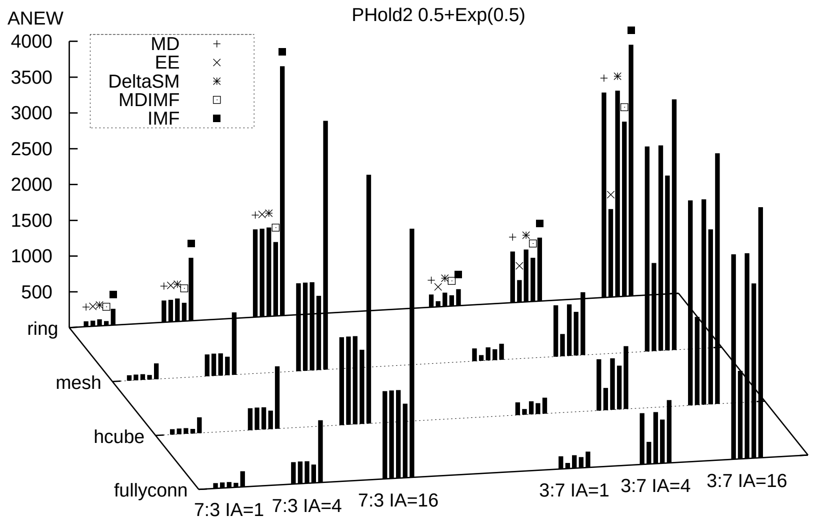


Fig. 5 ATWS for the PHold2 model





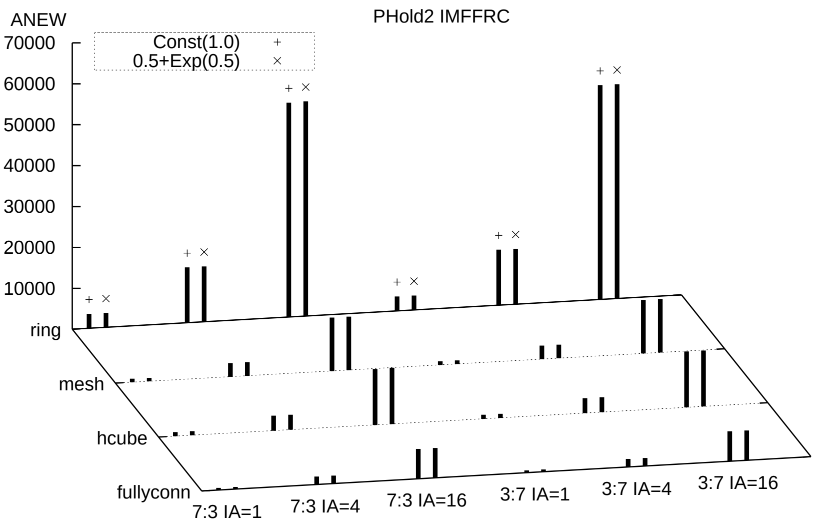


Fig. 6 ANEW for the PHold2 model

[8] Gilbert Chen and Boleslaw Szymanski. Dsim: A distributed optimistic parallel discrete event simulator. http://www.cs.rpi.edu/ cheng3/dsim, accessed July 3, 2007.

[9] Rassul Ayani and Hassan Rajaei. Parallel simulation using conservative time windows. In *Proc. 24th Winter Simulation Conference*, pages 709–717, New York, NY, USA, 1992. ACM Press.