Notes on Parallelizing ATM-TN Cell-level Simulation Models

Technical Report for Telesim Project

Zhonge Xiao
Computer Science Department
University of Calgary
Canada
email:xiao@cpsc.ucalgary.ca

May 1995

Contents

1 Introduction **2**

2 General Study **2**
2.1 Basic Parallel Elements in DES 2
2.2 Parallelism Analysis of ATM-TN model 5

3 Experiments with Single Pipeline **7**
3.1 The Pipeline Model and its Parallelism 7
3.2 Experiment Results 8

4 What can we expect ? **15**

5 Experiments with Two Pipelines **18**
5.1 Parallelism Analysis 20
5.2 Experiments 22

6 Experiments with Multiple Pipelines **27**
6.1 Intersecting Pipelines 27
6.2 Independent Pipelines 29
6.3 Mixed Pipelines 31

7 Experiments with Wnet **33**
7.1 Wnet Overview 33
7.2 Partitioning for Parallel Execution 35
7.3 Experimental results 37

8 Cell Events Flow Control **40**
8.1 Why need Cell Flow Control ? 40
8.2 Cell Flow Control Mechanisms 41
8.3 experiment results 43
8.4 Improve the Mechanism 47

9 Conclusions **49**

1 Introduction

This is a facts finding study for parallelizing ATM-TN simulation models. ATM-TN stands for ATM Traffic and Network. It is a frame work for detailed cell level ATM simulation models. The users are responsible for providing network topology and specifying parameters for each component of the network being modeled. To a large extent, the principles presented in this report also apply to other cell level ATM simulation models. The purpose of the study is to find out what is the available parallelism in this type of ATM-TN models, how to parallelize the models in general, and what problems there are to be addressed for the Time Warp mechanism to get reasonable speedup with the ATM-TN simulation given the very small granularity of the simulation models.

2 General Study

2.1 Basic Parallel Elements in DES

There are three basic parallel elements in general discrete event simulations:

Logical Process

The basic active elements of parallel execution are the Logical Processes (LPs) in the simulation model. It is assumed that the division of LPs is based on the principle that all the events processed by a LP have to be processed one after another in sequence. This is also the basic principle that every DES simulator must observe including Time Warp. The processing of events within a LP can not be further parallelized. If there's no interaction among all the LPs, then we have total parallelism - each LP doing its own job isolated from others. The available parallelism in this case is N, the number of LPs. So N is the maximum possible parallelism in any DES models.

Event Thread

An Event Thread (ET) is defined as a sequence of events that each of them is a result of a previous event in the sequence, or each one is scheduled by the previous event. Each of these events can be processed by any LP, but must be processed in order. So an Event Thread is the basic unit of events that can not be processed in parallel. At any given time, each LP can be assigned at most one ET and each ET can only be assigned to one LP. The realizable parallelism in a simulation model is thus equal to or less than the number of ETs.

Figure 1 shows three LPs, each doing its job and no interaction among them. There are three ETs, one assigned to a LP permanently. We have total parallelism 3. Figure 2 gives a model of three LPs but only one ET. The ET is assigned to one LP at a time and the path of the ET forms a loop. We have no parallelism for this case. In Figure 3, there are more then three ETs, each has the same path - going through three LPs in turn. Assuming it takes exactly the same amount of time to processing an event, the realizable parallelism is also 3, same as the first example.

Real situations are much more complicated with ETs cross with each other which will reduce the parallelism.

Node Weight

The Weight of a node, or LP, is defined loosely as the time it takes to processing an event. In general, the Weight of a LP varies from event to event, but in most cases we are only interested in the average Weight of a LP. For the situation, depicted in Figure 3, if the Weight is different for each LP, the available parallelism will be reduced. For example, suppose the weights are 1, 2, 3 for the three LPs respectively, the actual parallelism would be 2 instead of 3, the two lighter LPs would have to wait for a while before they can pick up next event. It is often the case that we will use normalized Weight as will be discussed in next section.

With the concept of these three basic parallel elements, we can find out forms of parallelism, or causes that reduce the parallelism in the ATM-TN model more easily and in a systematic way.

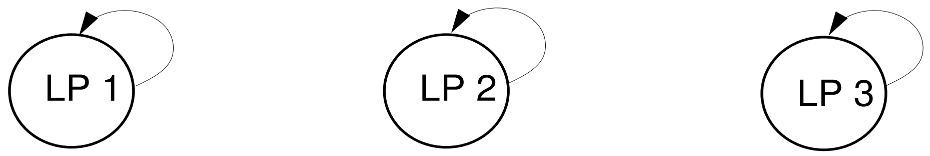


Figure 1: **Independent Logical Processes**

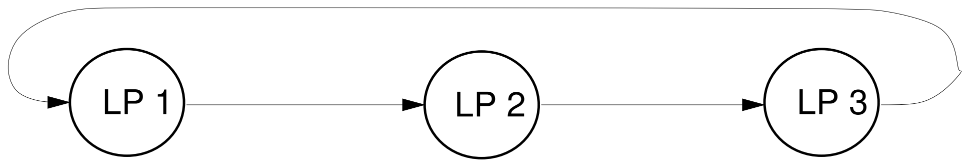


Figure 2: **Single Event Thread**

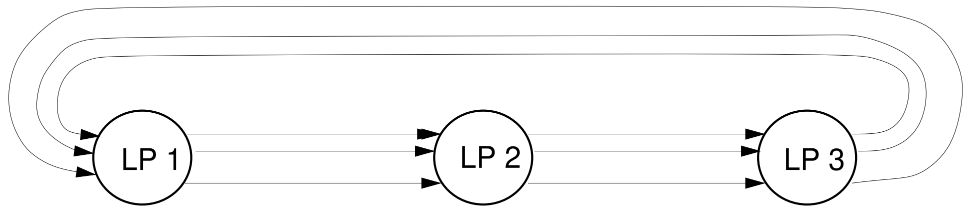


Figure 3: **Multiple Event Threads on a single path**

2.2 Parallelism Analysis of ATM-TN model

The Basic form of cell transfer in the ATM-TN model is pipeline, or Tandem line as some literature put it. A pair of traffic source-sink along with all the switch ports the cells travel through before they reach the sink form a pipeline. Each cell is carried by a sequence of events along the line, so each cell creates an Event Thread. Usually, the traffic sources generate far more cells, or ETs than the number of ports, i.e. LPs in terms of simulation model, along the pipeline. This is exactly the situation described in Figure 3. There are usually enough cells or ETs at any given time for each LP on the pipeline to process. If the Weights are the same for all the LPs, we have tatol parallelism here - the number of of LPs on the line. Unfortunately, this could hardly be the case. The parallelism, which is reduced because of the unequal Weight of the LPs along a pipeline can be calculated by the following formula:

$$parallelism = \frac{\sum_{i=1}^{N} W_i}{W_{max}}$$

where N is the number of LPs along the pipeline;
W_i is the Weight of LP i while W_{max} is the heaviest among all the W_i.

The LP which has the heaviest Weight becomes the bottleneck of available parallelism in a single pipeline. For convenience, we call this LP the Critical Node of the Pipeline. Note that the Weight of a LP refers to the normalized Weight, that is the average processing time for an event multiplied by the number of events scheduled on a LP for a cell. It is possible that a cell causes more than one event on a LP. The simple example is an arrival event and a departure event for a cell.

The intersection of two pipelines will reduce the parallelism. The formula of parallelism calculation for each pipeline is similar to the one given above except that the Weight at the intersection will be the *inflated* weight if the LP at the intersection is a critical node, that is its inflated weight is in deed heavier than the weiths of other LPs on the pipeline in question. The exact definition of *inflated weight* will be discussed later. However, if the inflated weight at an intersection is not the heaviest among the weights of all the nodes on a pipeline, then the parallelism of this pipeline is not affected by that intersection. If a pipeline is intersected by more than one other line, its parallelism will be determined by the intersection where the inflated weight is the heaviest. Weights of LPs other than the critical node should be calculated as if there're no intersections. A simple explanation for this is that all those lighter LPs will have to idle waiting for the heaviest LP to finish processing, the effect of intersections is just filling in the idling time.

The total parallelism of a ATM-TN model can be calculated based on the calculation of each pipeline as outlined above. However, this is not simply the sum of individual amount of parallelism. A similar normalization is required since some pipeline may have heavier traffic going though while others have less cells traveling on them. We need only consider the Critical Nodes of all the Pipelines in this regard. The details of this calculation method will be further discussed by examples in later sections.

This method outlined here has been verified against measured numbers from simulation runs of some ATM-TN models. For the case of single pipeline the calculated results are very close to measured. For a more real model - the WurcNet model (with 12 switches and 12 traffic source-sink pairs), the difference is about 5%. The measured parallelism is calculated by dividing the sum of execution time of all the events with the event execution time along the critical path (excluding system overhead). Both these values are computed on the fly in the Warpkit simulator. The calculations based on the method described here rely on the statistics data reported by Simkit. The data items of interest are the total number of events on a LP and the average processing time of an event for that LP.

What is more significant about this analysis method is to obtain accurate knowledge as to where and how much parallelism is available in a particular application model for a specific configuration. This can be very useful in partitioning, a crucial step in parallelizing ATM-TN models. And it may prove to be possible to develop an automatic partitioning method based on what we have discussed here. These could be equally applicable to both conservative and optimistic PDES.

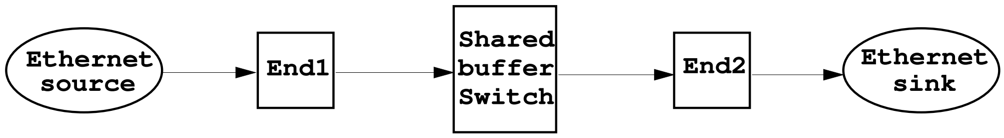


Figure 4: **Simple One Switch Network**

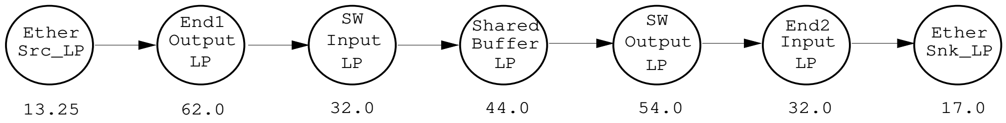


Figure 5: **The Pipeline model**

3 Experiments with Single Pipeline

3.1 The Pipeline Model and its Parallelism

As discussed above, an ATM-TN simulation model can be decomposed into a set of Pipelines intersecting with each other. Each Pipeline consists of a number of LPs. By normalizing the Weights at LPs of a Pipeline, where it intersects with other Pipelines, we can treat it as a stand-alone Pipeline. So studying the behavior of single Pipeline becomes the basis for further experiments with more complicated models.

The first ATM-TN model used for this study is a single switch network with an Ethernet Traffic Source-Sink (TSS). The Ethernet TSS is connected to the switch through Endnodes (Figure 4). The Switch is a single stage switch with shared buffer.

Figure 5 shows the single Pipeline model for this simple ATM-TN model. It consists of 7 LPs. The Weights of LPs are normalized Weights which are based on the data collected from running the simulation on one processor. All the output ports LPs need two events for each cell, so the Weight of these LPs equals to twice the time it takes to process an event. In Figure 5 the numbers under each LP are the normalized weights, or the processing time for a cell to pass through a LP. The available parallelism is :

$$parallelism = \frac{\sum_{i=1}^{7} W_i}{W_{max}} = \frac{254.25}{62} = 4.10$$

It is very close to the measured value :

$$parallelism = \frac{< sequential - exec - time >}{< critical - path - exec - time >} = \frac{11.569}{2.839} = 4.08$$

3.2 Experiment Results

The experiments were carried out on SGI Power Challenge using Simkit_par 1.0 as underlying simulator. Three processors (PE) were executing in parallel on this model. The following are the summary results of 5 different partitioning. The number of initial event buffers was 20,000 for all the tests.

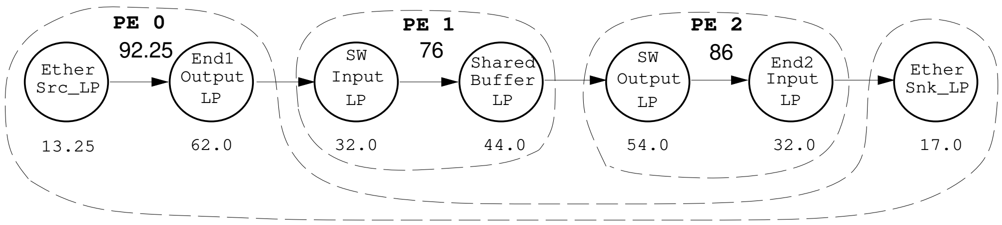


Figure 6: **Partitioning 1**

```
========================= TEST 1 ===========================
 Partitioning --  PE0: Ether-Src, End1-Out, Ether-Snk. - 92.25
                  PE1: SW-In, Shared-Buf. - 76
                  PE2: SW-Out, End2-In. - 86

 per committed event time: 22.41 uS
 ----------------------------------
         forward overhead:  7.79 uS
            send overhead:  2.40 uS
                   idling:  1.06 uS
          user exec. time:  9.33 uS
        rollback overhead:  0.24 uS
  global control overhead:  1.59 uS

 efficiency: 99%
  rollbacks: SW-In - 4.24%
             Shared-Buf - 2.28%
             SW-Out - 0.33%
             End2-In - 0.91%
  rollback distance: 1 (ave. & max.)

 global control: per PE GVT computations - 41
                 no-fossils - 0
                 spins - 0
```

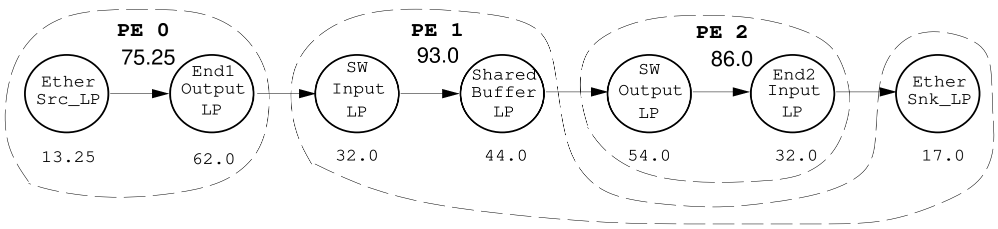


Figure 7: **Partitioning 2**

```
========================= TEST 2 ===========================
 Partitioning --  PE0: Ether-Src, End1-Out. - 75.25
                  PE1: SW-In, Shared-Buf, Ether-Snk. - 93
                  PE2: SW-Out, End2-In. - 86

 per committed event time: 26.03 uS
 ----------------------------------
         forward overhead:  7.78 uS
            send overhead:  2.08 uS
                   idling:  0.80 uS
          user exec. time:  8.64 uS
        rollback overhead:  1.77 uS
  global control overhead:  5.17 uS

 efficiency: 91.82%
  rollbacks: SW-Out - 15%
             End2-In - 31%
  rollback distance: 1 (ave. & max.)

 global control: per PE GVT computations - 36608
                 no-fossils - 32114
                 spins - 28686
```

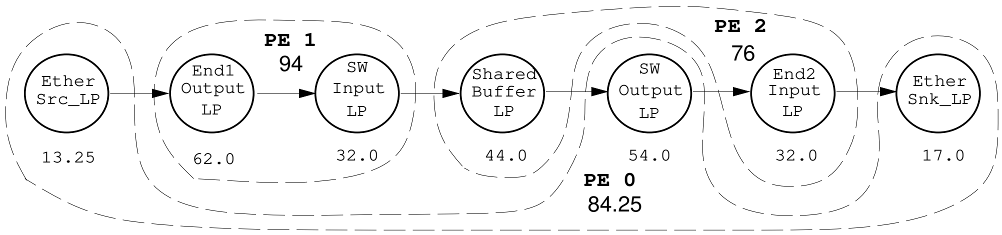


Figure 8: **Partitioning 3**

```
========================== TEST 3 ===========================
 Partitioning --  PE0: Ether-Src, SW-Out, Ether-Snk. - 84.25
                  PE1: End1-Out, SW-In. - 94
                  PE2: Shared-Buf, End2-In. - 76

 per committed event time: 21.51 uS
 ----------------------------------
         forward overhead:  7.54 uS
            send overhead:  2.82 uS
                   idling:  0.18 uS
          user exec. time:  9.42 uS
        rollback overhead:  0.03 uS
  global control overhead:  1.53 uS

 efficiency: 99.87%
  rollbacks: Ether-Snk - 0.00025%
             SW-In - 0.0002%
             Shared-Buf - 0.0033%
             SW-Out - 0.00009%
             End2-In - 0.0044%
  rollback distance: 1 (ave. & max.)

 global control: per PE GVT computations - 46
                 no-fossils - 0
                 spins - 0
```

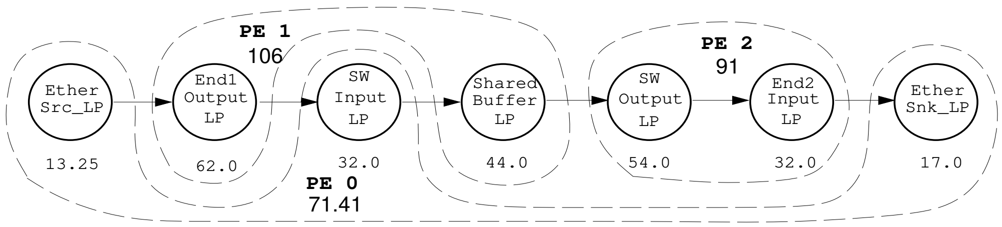


Figure 9: **Partitioning 4**

```
========================= TEST 4 ==========================
 Partitioning --  PE0: Ether-Src,  SW-In, Ether-Snk. - 62.25
                  PE1: End1-Out, Shared-Buf. - 106
                  PE2: SW-Out, End2-In. - 86

 per committed event time: 25.54 uS
 ----------------------------------
         forward overhead:  8.66 uS
            send overhead:  2.49 uS
                   idling:  0.93 uS
          user exec. time: 11.44 uS
        rollback overhead:  0.46 uS
  global control overhead:  1.57 uS

 efficiency: 98.26%
  rollbacks: SW-In - 4.5%
             Shared-Buf - 1 rollbak
             SW-Out - 1.3%
             End2-In - 2.6%
             Ether-Snk - 5.8%
  rollback distance: 1 (ave. & max.)

 global control: per PE GVT computations - 57
                 no-fossils - 0
                 spins - 0
```

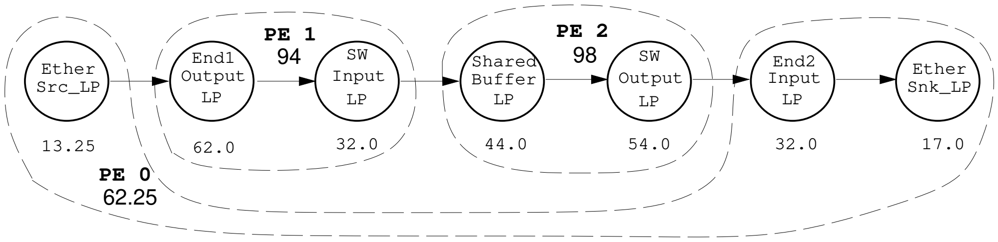


Figure 10: **Partitioning 5**

```
========================== TEST 5 ===========================
 Partitioning --  PE0: Ether-Src, End2-In. Ether-Snk. - 62.25
                  PE1: End1-Out, SW-In. - 94
                  PE2: Shared-Buf, SW-Out. - 98

 per committed event time: 25.64 uS
 ----------------------------------
         forward overhead:  8.08 uS
            send overhead:  1.96 uS
                   idling:  0.99 uS
          user exec. time: 11.81 uS
        rollback overhead:  0.38 uS
  global control overhead:  2.42 uS

 efficiency: 98.5%
  rollbacks: Shared-Buf - 0.69%
             SW-Out - 0.73%
             End2-In - 3.5%
             Ether-Snk - 6.8%
  rollback distance: 1 (ave. & max.)

 global control: per PE GVT computations - 13382
                 no-fossils - 18612
                 spins - 0
```

The partitioning is based on the Weights of the LPs. The load of a PE equals to the sum of the Weights of all the LPs on that PE. The numbers given in the partitioning are the load of the PEs for the particular partitioning. With equal load, the PEs will process events at roughly the same rate, so the load are balanced.

Normally, one would expect no rollbacks in the case of single Pipeline, but idling of PEs which have lighter load. The rollbacks reported in these tests are the result of an artifact of the design of this ATM-TN model. There's a external events handler LP which schedules the report events - determined future events - at the initialization phase. These future events sit in the schedule list of every LP right from time zero and wait for every possible chance to be processed. On those PEs, like PE1 , PE2 in test1 and PE2 in test2, that have a lighter load will be able to process events faster than its previous PE, so there will be times when no cell events available and these PEs will grab the determined future events for processing. When new cell events arrive, these future events will be rolled back. This will happen periodically and these events keep rolling forth and back until the time is due for them. In the test model, there is only one future event , final report, for each LP, so the rollback distance is exactly 1. This type of rollbacks happens at those PEs that process event faster than their predecessors, as a result there will be times when no cell events are available and the future events will be picked up.

The results of these tests indicate that the performance is very sensitive to the load balance. In test2 and test5, the PE (PE 0) on the traffic source side has 20% to 30% less load and will process events faster than its successors PE 1 and PE 2. As a result, PE 0 consumes its event buffers faster and engages itself in GVT at much higher frequency. The situation in test2 is worse where PE 0 never receives cells, so it will execute as fast as it can pushing the global control mechanism to its limit. It is interesting to notice that even under such an extreme condition, the overall performance degradation is about 20%. This is due to the asynchronous global control mechanism in Warpkit, the Time Warp kernel of the Simkit simulator.

In test4, the load balance is similar to test2. The average number of GVT computations has gone down as one more LP, which sends out a cell only when it receives one, is added to the traffic source side PE, PE0. The global

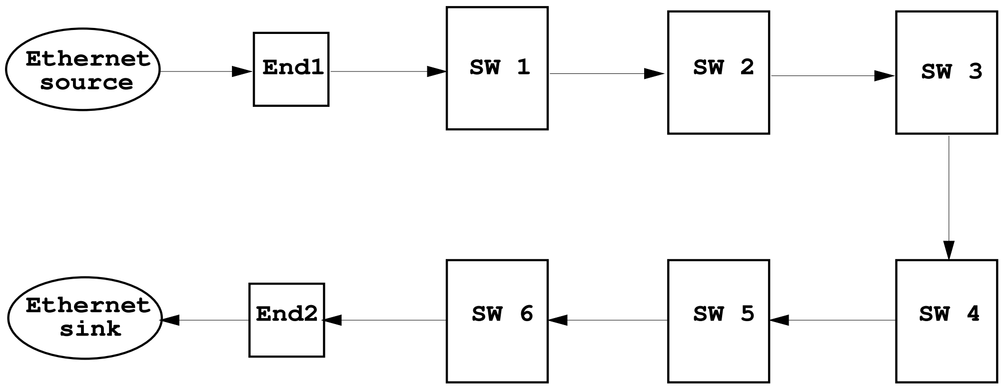


Figure 11: **A Single Path Network**

control overhead is much smaller compared to test2, but the forward execution overhead and the user execution time have increased quite a bit. The explanation for this might be the cache effect. The number of PEs a cell travels from the source to the sink is 6 (PE0-PE1-PE0-PE1-PE2-PE0), it changes the PE almost for each LP.

Test3 has the best performance because the loads of the PEs are very close and the rollbacks are brought under control by arranging the LPs on PE2 (which has the lighter load) intermittent with those on PE0.

4 What can we expect ?

To find out what is the best speedup we can expect from the ATM-TN model with current implementation of parallel Simkit, we used a ATM-TN model of 6 simple Per-Port swithes, connected to form a single Pipeline as Figure 11 shows. This is an almost ideal case, so we expect to see the best speedup we can get.

Figure 12 gives the Pipeline model. The calculated available parallelism is 11.59 while the measured one is 11.60. The partitioning is almost perfect in the sense that all the PEs have very close load (around 95) except PE7 (36). The experimental results obtained before the SGI at UofC was updated to new O.S. 6.01 are as follows.

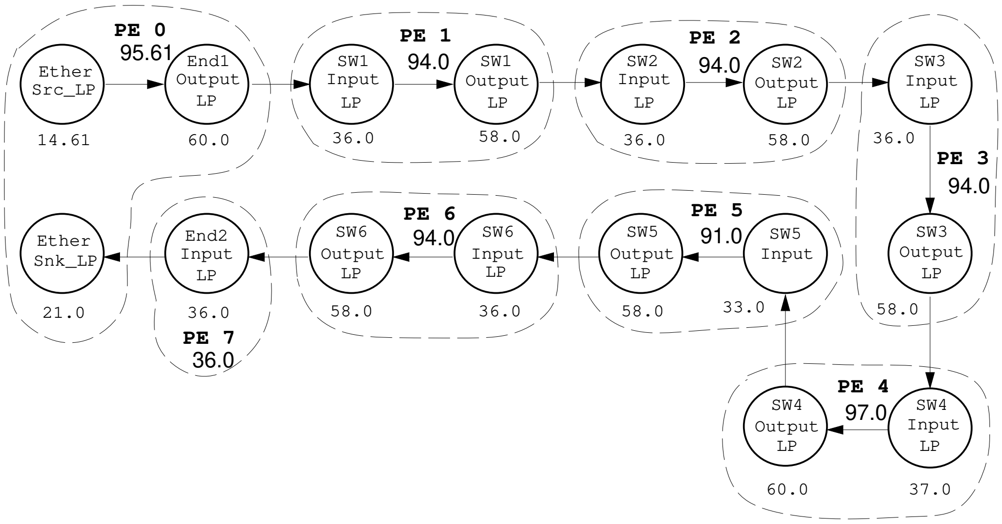


Figure 12: **The Pipeline model of Figure 6**

```
                      average per event time

sequential            21.90 uS
(Simkit_seq 1.0)

parallel on 8 PEs      9.83 uS
(Simkit_par 1.0,       (user time 3.64, system time 5.74)
 non-streamlined)

parallel on 8 PEs      7.39 uS
(Simkit_par 1.0,
 streamlined)

speedup : 21.90/7.39 = 2.96
relative granularity : G = 3.64/5.74 = 0.63
```

Assuming that user processing time remains the same on multi-processors as that on single processor, the maximum speedup over sequential simulator one can expect will be less than:

$$speedup_{max} = N * \frac{1}{1 + \frac{1}{G}} = 8 * 0.39 = 3.10$$

We are close to the limit here. However, the maximum possible speedup is only 3.10 on 8 PEs. The reasons for this are:

poor scalability
: On an ideal parallel platform, the system overhead and the user execution time would be the same no matter on how many processor a program is executing. Under such ideal case, we would have linear speedup, i.e. speedup 8 on 8 PEs (with the same simulator). The closer a machine is to the ideal one, the better its scalability. It has been found that SGI has very poor scalability for the Simkit. In this test case, the overall system overhead on 8 PEs is twice as much as that on one PE. On one PE the relative Granularity G is 1.15, so the maximum possible speedup would be 4.24 on 8 PEs if the multiprocessor machine were perfectly scalable. The actual speedup from one PE to 8 PE with the same Simkit parallel is 5.11 instead of something much closer to 8.

high system overhead
: The Simkit system overhead is still too high compared to the sequential Simkit and the user processing time. The scalability of Simkit on SGI is not adequate. If rollback rate is high, system overhead could become unbearable. Simkit sequential version is highly optimized with just four C statements in the main loop of the simulator scheduler including one call to get the next event.

small granularity
: The nature of ATM-TN transfer mode determines the small user processing time as the majority of events will be cell events. The processing of cells in real switches is done by hardware and minimum processing is required. Together with high system overhead, the relative granularity is generally low. In more realistic network models where switches are fully loaded, user event processing time will be higher as more buffering

operations are required. This will increase the granularity and lead to higher speedup.

From the numbers above it can be seen that poor scalability and high system overhead (compared to sequential simulator) contribute equally to the limited speedup over sequential execution. One can not expect big speedup unless the Time Warp system overhead can be further reduced significantly or some other low overhead scheme can be implemented.

Ironically, after the SGI at UofC got updated to new O.S., the situation becomes further in favior of sequential execution. The execution time for sequential now has dropped to 14 uS per event while on multi-processors the improvement is very small, so we ony have speedup of close to 2 , compared to previous 21.9 uS, for the test model shown in Figure 12. Fortunately, the situation did not get that worse with more complicated models like Wnet as will be presented later in this report.

From the above experiments and what we have discussed, it is predicable that with current ATM-TN and Simkit implementation, the best speedup we may expect is around 2.5 to 3.5 out of 8 PEs.

5 Experiments with Two Pipelines

The parallelism of ATM-TN models will be affected by the intersection of pipelines. We use a model consisting of one switch and two Ethernet traffic source-sinks for analysis and experiments. Figure 13 shows the network model. Two Ethernet traffic sources reach the switch at different input ports, but merge to the same output port. Thus the corresponding pipeline model has a form of two pipelines intersecting at a output port of the switch and the input port of endnode 3 as depicted by Figure 14. The number in microseconds printed by each LP is the normalized weight of that LP which is based on the data collected in the simulation run on single processor.

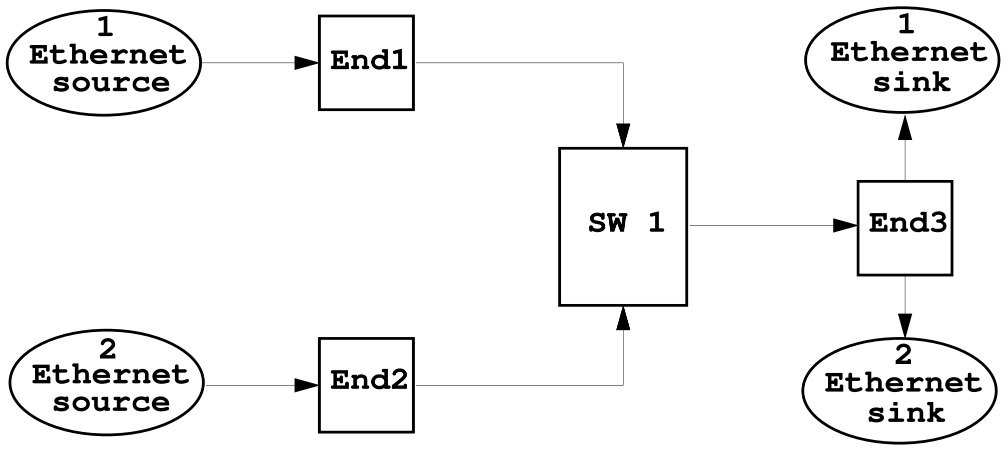


Figure 13: **Two TSSs and one Switch Network**

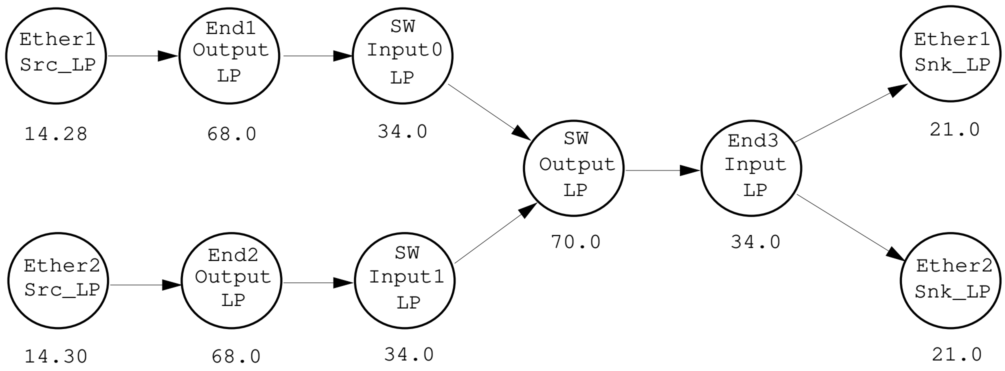


Figure 14: **Two Pipeline model**

5.1 Parallelism Analysis

We analyze the available parallelism for two intersecting pipelines first. When two pipelines intersect with each other, the LPs at the intersection have to process events from both pipelines, so the rate of cells passing though those LPs for each pipeline will be reduced, or the time it takes to pass a cell down to next LP will be longer and the parallelism will be reduced. To take into account of this negative effect, we introduce *inflated weight*. To distinguish it from the normalized weight which we have referred so far. We call the latter *intrinsic weight*. Suppose at the intersection of two pipelines, the events processed for pipeline 1 is $N1_{ev}$ and the events for pipeline 2 is $N2_{ev}$. The *inflated weight* of the LP at the intersection for pipeline 1 is defined as:

$$W1_{infl} = W1_{intr} * \frac{N1_{ev} + N2_{ev}}{N1_{ev}}$$

Similarly, the *inflated weight* for pipeline 2 is

$$W2_{infl} = W2_{intr} * \frac{N1_{ev} + N2_{ev}}{N2_{ev}}$$

In general, the inflated weight for intersecting pipelines are different. It is also possible that W_{intr} could be different for different pipelines. For the ATM-TN models, however, the intrinsic weight is always the same for all the pipelines at intersection LPs. The available parallelism for intersected pipeline can be calculated by:

$$parallelism = \frac{\sum W_i}{W_{max}}$$

$$W_{max} > W_i$$

where $\sum W_i$ is the sum of intrinsic weight of all the LPs in the pipeline including the LP at the intersection. If W_{infl} is lower than any of the W_i, then the parallelism of this pipeline is not affected by the intersection and the W_{max} should be replaced by the highest of W_i in calculating the parallelism.

For the model shown in Figure 14, the inflated weight of Traffic Source-Sink 1 is:

$$W1_{infl} = W1_{intr} * \frac{N1_{ev} + N2_{ev}}{N1_{ev}} = 70 * \frac{46794 + 46934}{46794} = 140.0$$

where 46794 is the number of cells on TSS 1 and 46934 is the number cells on TSS 2.

The inflated weight for traffic source-sink 2 is:

$$W2_{infl} = W2_{intr} * \frac{N1_{ev} + N2_{ev}}{N2_{ev}} = 70 * \frac{46794 + 46934}{46934} = 140.0$$

The two TSS have equal load, so their inflated weights are very close. The sum of intrinsic weights for both TSS are almost the same for this specific example:

$$\sum W_i = 14.0 + 68 + 34 + 70 + 34 + 21 = 241.0$$

The parallelism for both are:

$$parallelism = \frac{\sum W_i}{W_{infl}} = \frac{241}{140} = 1.72$$

The total parallelism is the sum of the two pipelines (general formula will be presented later, this is a special case), 3.44. The measured parallelism is:

$$measured\ parallelism = \frac{18.45}{5.38} = 3.43$$

The validity of the formula for calculating parallelism of individual pipeline can be easily proven. $\sum W_i$ is the time it takes to finish one cell on the pipeline in sequential execution while W_{infl} is the least time needed to complete one cell in parallel execution.

5.2 Experiments

Here are the results of experiments with the two pipeline model depicted in Figure 14. Since the available parallelism is merely 3.44, execution on 4 PEs is a waste of computing power and will result in un-even load. Using three PEs seems more efficient. Three different partitioning have been tested. None of them could have well balanced load. The partitioning is again based on the weight of the LPs, for those LPs at intersection the inflated weight is used instead of intrinsic weight. The load of a PE is the sum of the weight of all LPs assigned to it. The target of partitioning is to have the load of all the PEs as close as possible, plus some other considerations. One of the guide lines is always to put the Ethernet Source LP and Sink LP on the same PE. As a result, No event buffer migration would normally be necessary as the Source sends a cell along with an event buffer and the Sink will finally receive the cell carried by another event buffer. With both Source and Sink LP on the same PE, the event buffer lost in sending a cell can be compensated by the event buffer which carries the cell to the Sink. However, the need for buffer migration is a characteristic of Warpkit which picks up buffers from local pool for sending messages.

The experiments were carried out on SGI Power Challenge using Simkit_par 1.0 as underlying simulator. The following are the summary of results of 3 different partitioning. The number of initial event buffers was 10,000 for all the tests.

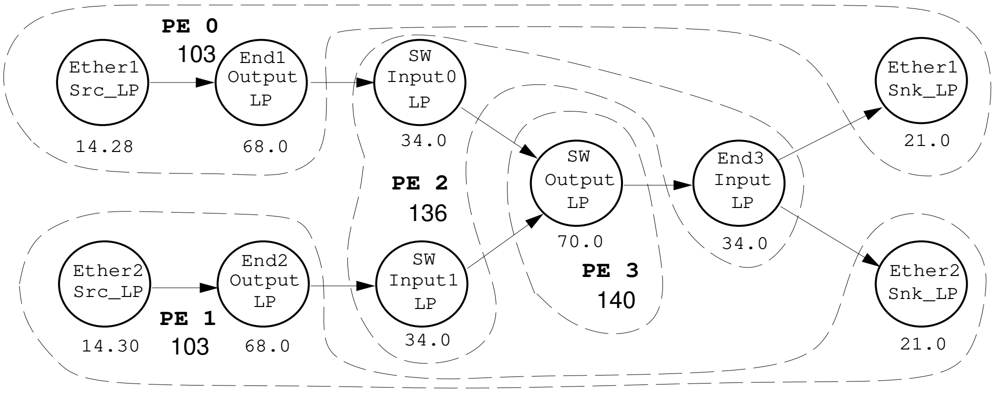


Figure 15: **Partitioning 1**

```
======================== TEST 6-4-1 =========================
 Partitioning --  PE0: Src1, End1-Out, Snk1. - 103.28
                  PE1: Src2, End2-Out, Snk2. - 103.30
                  PE2: SW-In0, SW-In1, End3-In. - 136
  PE3: SW-Out. - 140

 per committed event time: 26.17 uS (abs. 104.70 uS)
 ---------------------------------------------------
         forward overhead: 10.51 uS
            send overhead:  0.71 uS
                   idling:  1.62 uS
          user exec. time:  7.00 uS
        rollback overhead:  0.29 uS
  global control overhead:  6.03 uS

 efficiency: 99.71%
  rollbacks: SW-In - 4.5%
  abortions: 17685

 global control: per PE GVT computations - 91043
                 no-fossils - 173936
                 spins - 33528
```

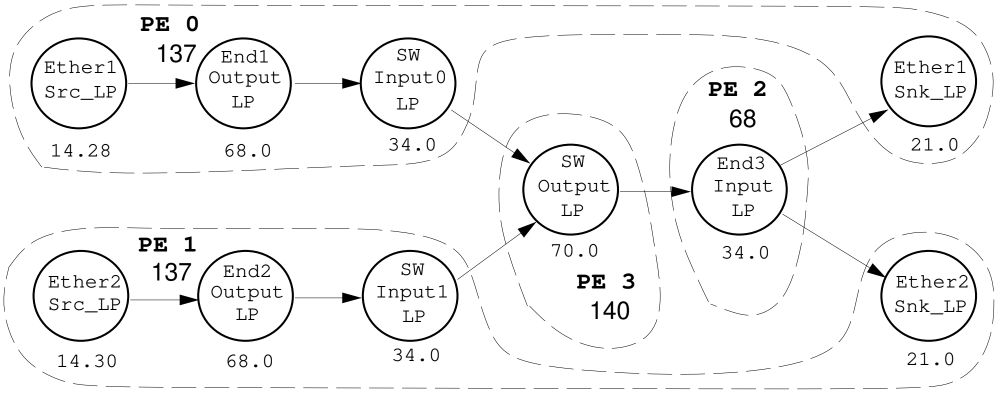


Figure 16: **Partitioning 2**

```
======================= TEST 6-4-2 =========================
 Partitioning --  PE0: Src1, End1-Out, SW-In0, Snk1. - 137
                  PE1: Src2, End2-Out, SW-In1, Snk2. - 137
                  PE2: SW-Out. - 140
  PE3: End3-In. - 68

 per committed event time: 21.92 uS (abs. 87.70 uS)
 --------------------------------------------------
         forward overhead:  7.68 uS
            send overhead:  2.16 uS
                   idling:  2.08 uS
          user exec. time:  7.23 uS
        rollback overhead:  0.00 uS
  global control overhead:  2.77 uS

 efficiency: 100%
  abortions: 0

 global control: per PE GVT computations - 37260
                 no-fossils - 62413
                 spins - 0
```

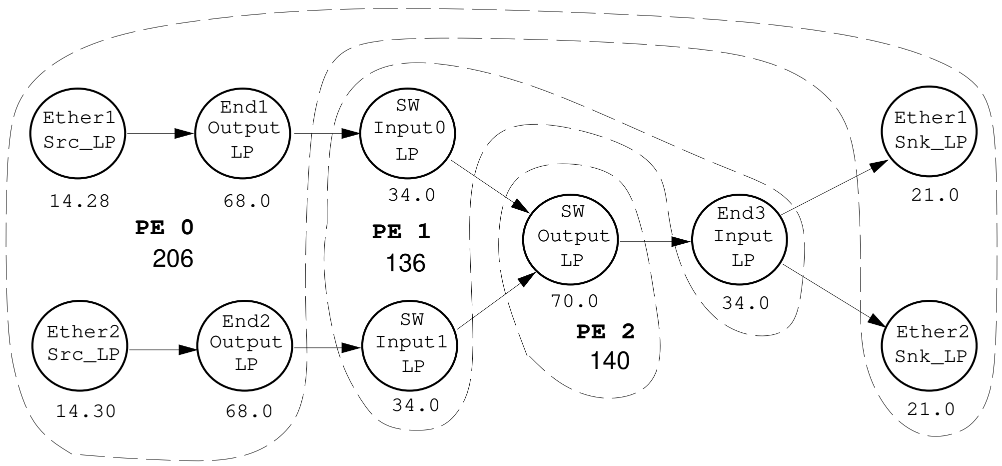


Figure 17: **Partitioning 3**

```
======================= TEST 6-3-1 ========================
  Partitioning --  PE0: Src1, End1-Out, Snk1,
Src2, End2-Out, Snk2. - 206.58
                   PE1: SW-In0, SW-In1, End3-In. - 136
                   PE2: SW-Out. - 140

  per committed event time: 25.73 uS (abs. 77.19 uS)
  --------------------------------------------------
          forward overhead:  8.86 uS
             send overhead:  2.89 uS
                    idling:  2.74 uS
           user exec. time:  9.20 uS
         rollback overhead:  0.26 uS
   global control overhead:  1.99 uS

  efficiency: 99.74%
   abortions: 0

  global control: per PE GVT computations - 24
                  no-fossils - 0
                  spins - 0
```

Test 6-4-1 is the worst case where two PEs on the traffic source side are lightly loaded. These two PEs (PE 0 and PE 1) are able to pump cells into the network at much higher rate than the two PEs down the pipeline. As a result, the traffic sources will generate cells far ahead until exhaust the event buffers and will constantly resort to fossil collection to get buffer for events. At this point the load condition on source side has changed because of the significantly higher system overhead being spent on GVT computation and fossil collection. The traffic sink LPs, which reside on the same PE as their source LP's, will have to suffer serious buffer shortage too. Large number of GVT computation are involved and the chance of no fossil to collect is very high. What is worst is the high number of event abortion. The abortion occurs when a PE runs out of buffer and could not reclaim any because some of its LPs is executing far in the future and used all the buffers, at the same time some of its LPs (the sink LP or endnode 3 in this case) is holding the GVT. In such a situation, the processor has no choice but to abort the current event.

In Test 6-4-2, the PE 0 and PE 1 on the source side get higher load, very close to PE 2 where the critical node - the output port of the switch resides. PE3 has only half of the load, compared to the other three PEs. Global control cost is much lower because the tension on it has been eased. Yet quite high number of GVT computation has been reported due to the load on the source side is slightly lower. This demonstrates once again that Time Warp simulator is very sensitive to the load balance in ATM cell level models. But the global control of Warpkit is not costly in less tense situations like this one. On the other hand, lighter load on the source side has certain advantage. It will result in *cell buffering* at the output port of the switch which is where rollback originates as cells from different input ports merge onto the same outgoing link. This type of *cell buffering* is very effective in reducing statistical rollbacks caused by fluctuation of the time stamp of incoming cells. This can be justified by few rollbacks (two in one particular run) eported with this partitioning. Lighter load on PE 3 results in higher idling. This, however, does not affect other PEs as endnode 3 sends out a cell to one of the Sink LPs only if it receives one, there are no traffic sources on it.

Test 6-3-1 uses three PEs instead of four and more load are put on the PEs on the Source side. Now the global control cost becomes very low as source side PE 0 executes slower than the rest and the Source LPs will never have the chance to rush ahead and to exhaust the event buffer. The absolute per event time (77.19 uS) is much lower than the other two partitioning (104.7 uS and 87.7 uS respectively). The higher idling is inevitable due to the small available parallelism. Idling does not have negative global affect compared to puting tension on global control, therefore is preferable over the latter. This partitioning represents the best arrangement for the particular model. Compared to test 6-4-2, quite a number of rollbacks are recorded. This is due to the fast event processing rate at the switch output port and less chance of *cell buffering* as mentioned above.

6 Experiments with Multiple Pipelines

A set of three tests presented in this section are for the purpose of verifying parallelism calculation formulas. We will give the formula for each of these three cases and compare the calculated result with the measured value.

6.1 Intersecting Pipelines

For each individual pipeline of a set of pipelines that intersect with each other, its available parallelism can be calculated by:

$$parallelism = \frac{\sum W_i}{W_{max}}$$

$$W_{max} = \{W_{intr}, W_{infl}\}$$

W_{max} is the weight of its critical node. The critical node of a pipeline is the LP in the pipeline that has the highest weight among all the LPs in the pipeline. If the critical node is at a intersection, then W_{infl}, its inflated weight, is taken as its weight, i.e. $W_{max} = W_{infl}$. Otherwise $W_{max} = W_{intr}$. The critical node of a pipeline may not be the same as the critical node of other pipelines which intersect with it.

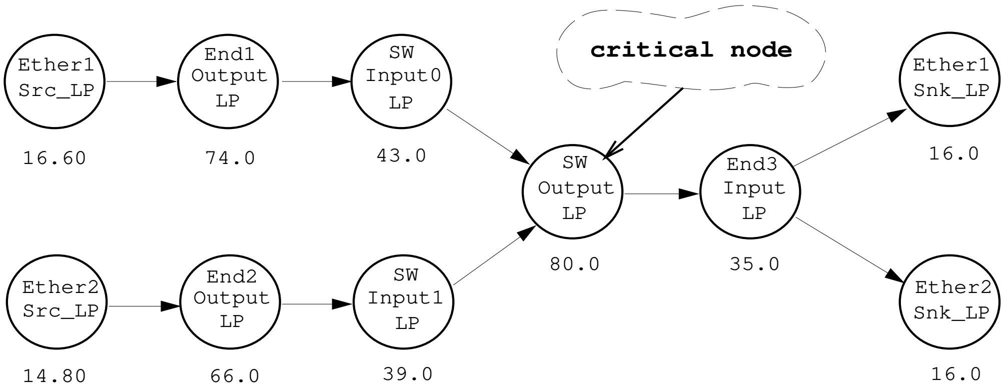


Figure 18: **Intersecting Pipelines**

Figure 18 shows a two pipeline model and the intrinsic weight of each LP. This model is the same as that used in last section for test6 except that the load of Ethernet 2 is much higher than the load on Ethernet 1. The number of cells processed by Ethernet Source-Sink 1, $N1_{cev}$, is 46794 over some time period and the number of cells for Ethernet Source-Sink 2 over the same period, $N2_{cev}$, is 141776. Switch output LP is the critical node for both pipeilines and the inflated weight of pipeline 1 is:

$$W1_{infl} = W_{intr} * \frac{N1_{cev} + N2_{cev}}{N1_{cev}} = 80 * \frac{46794 + 141776}{46794} = 322.38$$

and its available parallelism is:

$$\frac{\sum W1_i}{W1_{infl}} = \frac{16.60 + 74 + 43 + 80 + 35 + 16}{322.38} = 0.82$$

Similarly for pipeline 2 we have:

$$W2_{infl} = W_{intr} * \frac{N1_{cev} + N2_{cev}}{N2_{cev}} = 80 * \frac{46794 + 141776}{141776} = 106.40$$

$$parallelism = \frac{\sum W2_i}{W2_{infl}} = \frac{14.80 + 66 + 39 + 80 + 35 + 16}{106.40} = 2.36$$

The available parallelism for the whole model is the sum of individual pipeline's parallelism, i.e. $0.82 + 2.36 = 3.18$. The measured value from the same simulation run where the statistics data for LP weights and cell numbers were collected is:

$$measured\ parallelism = \frac{35.9}{11.30} = 3.18$$

6.2 Independent Pipelines

The notion of *inflated weight* has been used to account for the effect caused by pipeline intersection. It may not be concluded that the parallelism of a pipeline is not affected by another independent pipeline. If a pipeline takes less time to finish processing than another pipeline, then its parallelism will be reduced because the parallelism is available only before it finishes processing. Therefore the general formula for calculating parallelism for multi-pipelines is based on the processing time of the critical nodes, called *critical time* and denoted by T_{crtc}. The critical time of a pipeline is defined as:

$$T_{crtc} = W_{max} * N_{ev}$$

where W_{max}, N_{ev} are the weight and total number of events (or cells) on the critical node of a pipeline respectively.

The general formula for calculating actual available parallelism of a pipeline (P_k) is:

$$P_k = \frac{\sum W_i}{W_{max}} * \frac{T_{crtc}}{T_{max}}$$

where T_{max} is the maximum T_{crtc} of all the pipelines in a model. The total parallelism is the sum of individual values.

$$P_{totl} = \sum P_k$$

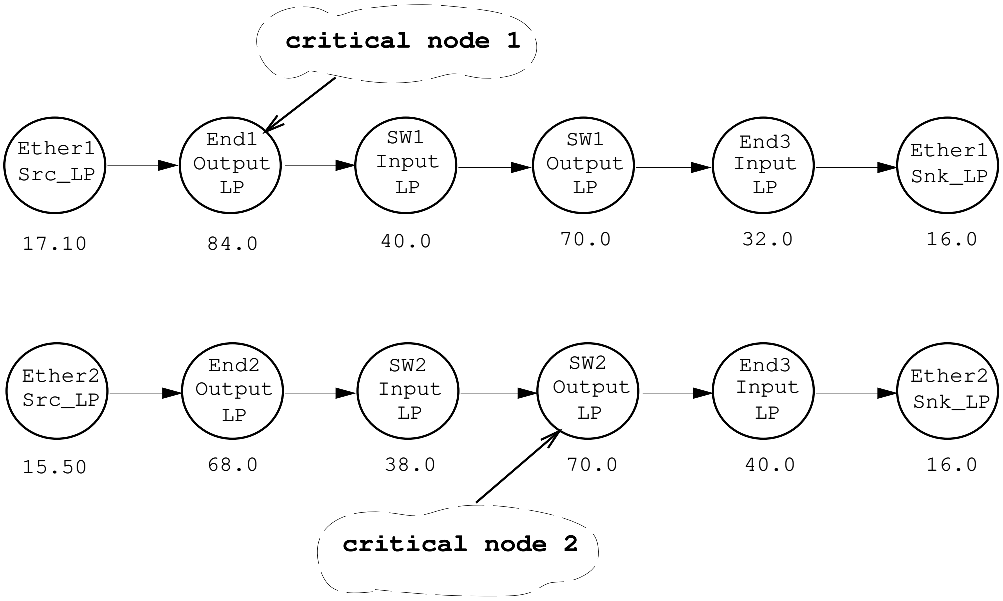


Figure 19: **Independent Pipelines**

Figure 19 gives a simple two independent pipeline model. For Ethernet 1 we have:

$$T1_{crtc} = W1_{intr} * N1_{ev} = 84 * 141886 = 11.92\ Sec.$$

For Ethernet 2, its $T2_{crtc}$ is the maximum,

$$T_{max} = T2_{crtc} = W2_{intr} * N2_{ev} = 70 * 420748 = 29.45\ Sec.$$

The parallelism of the two pipelines are:

$$P_1 = \frac{\sum W1_i}{W1_{intr}} * \frac{T1_{crtc}}{T_{max}} = \frac{17.1 + 84 + 40 + 70 + 32 + 16}{84} * \frac{11.92}{29.45} = 1.25$$

$$P_2 = \frac{\sum W2_i}{W2_{intr}} * \frac{T2_{crtc}}{T_{max}} = \frac{15.5 + 68 + 38 + 70 + 40 + 16}{70} * \frac{29.45}{29.45} = 3.54$$

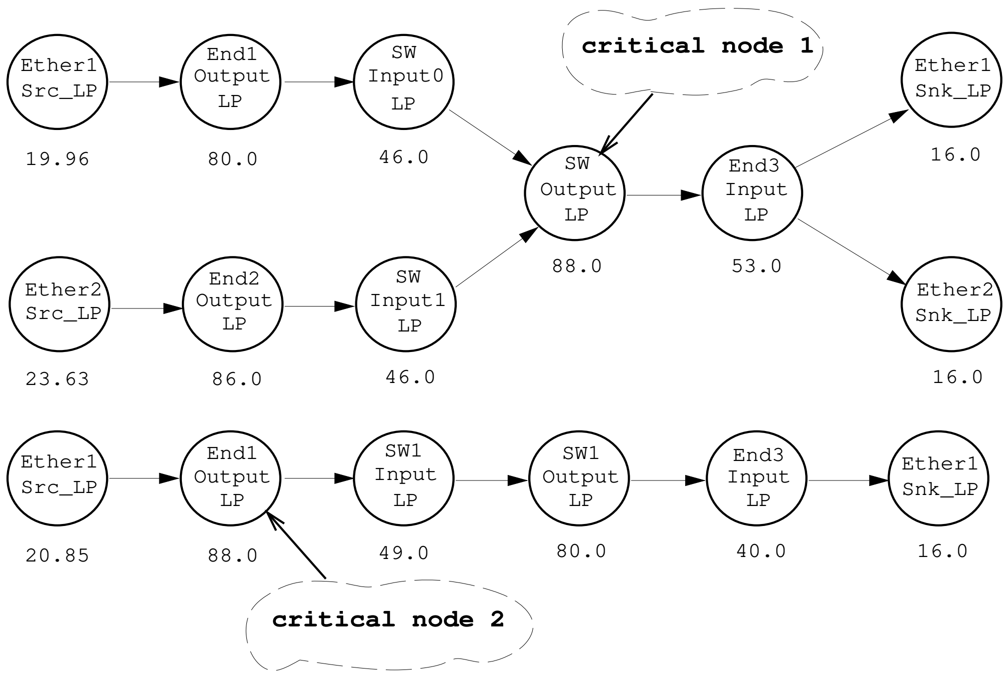


Figure 20: **Mixed Pipelines**

The parallelism is not affected for pipeline 2 because it holds the global critical node. The total available parallesim for the model is

$$P_{totl} = P_1 + P_2 = 1.25 + 3.54 = 4.79$$

The measured value is 4.65.

6.3 Mixed Pipelines

More general form of ATM-TN models consists of both intersecting and independent pipelines as exemplified by Figure 20. We refer pipeline 1 through 3 to Ethernet Source-Sink 1 through 3. The parallelism and critical time for each of them is as follows, Pipeline 1:

$$W1_{infl} = W1_{intr} * \frac{N1_{cev} + N2_{cev}}{N1_{cev}} = \ 88 * \frac{94687 + 280251}{94687} = 348.46$$

$$T1_{crtc} = W1_{infl} * N1_{cev} = 32.99\ Sec.$$

$$\frac{\sum W1_i}{W1_{infl}} = \ \frac{19.96 + 80 + 46 + 88 + 53 + 16}{348.46} = 0.87$$

$$P_1 = \frac{\sum W1_i}{W1_{infl}} * \frac{T1_{crtc}}{T_{max}} = \ 0.87 * \frac{32.99}{32.99} = 0.87$$

Pipeline 2:

$$W2_{infl} = W2_{intr} * \frac{N1_{cev} + N2_{cev}}{N2_{cev}} = \ 88 * \frac{94687 + 280251}{280251} = 117.73$$

$$T2_{crtc} = W2_{infl} * N2_{cev} = 32.99\ Sec.$$

$$\frac{\sum W2_i}{W2_{infl}} = \ \frac{23.63 + 86 + 46 + 88 + 53 + 16}{117.73} = 2.66$$

$$P_2 = \frac{\sum W2_i}{W2_{infl}} * \frac{T2_{crtc}}{T_{max}} = \ 2.66 * \frac{32.99}{32.99} = 2.66$$

Pipeline 3:

$$T3_{crtc} = W3_{intr} * N3_{cev} = 88 * 180634 = 15.90\ Sec.$$

$$\frac{\sum W3_i}{W3_{intr}} = \frac{20.85 + 88 + 49 + 80 + 40 + 16}{88} = 3.34$$

$$P_3 = \frac{\sum W3_i}{W3_{intr}} * \frac{T3_{crtc}}{T_{max}} = \ 3.34 * \frac{15.90}{32.99} = 1.61$$

The global critical node is output LP of Switch 1. So the parallelism of pipeline 1 and 2 are actually not affected by pipeline 3, but for pipeline 3 its parallelism is reduced significantly. The total parallelism is simply the sum:

$$P_{totl} = P_1 + P_2 + P_3 = 0.87 + 2.66 + 1.61 = 5.14$$

while the measured value is 4.99.

7 Experiments with Wnet

7.1 Wnet Overview

Wnet is a simplified ATM-TN model of Wurcnet, an experimental ATM network across Alberta, Saskachewen, and Manitoba of Canada. Artificial traffic sources are attached to the network model to simulate aggregate ATM traffic. The traffic models are MPEG model and Ethernet model, both are part of the general ATM-TN simulation package. Figure 21 shows the topology and traffic sources of the Wnet. The traffic sources consist of two way JPEG video between University of Alberta and University of Calgary, four inter-LAN Ethernet traffic between University of Calgary and other three institutions (UofA, HPCC and University of Manitoba). The traffic loads are set to saturate the link between Calgary_AGT switch and UofC switch so that about 1% to 2% CLR are recorded for Ethernet traffic coming to and out of UofC. There is another inter-LAN Ethernet traffic between University of Saskatoon and University of Regina, which does not mix with other traffics. The traffic arrangement is not particularly in favor of parallel execution as the loads of the switches are extremely un-even. Random partitioning which works well in many cases does not work here.

Figure 21: **Wnet map**

7.2 Partitioning for Parallel Execution

The partitioning method used for previous simple models is based on the normalized weight of a LP, or inflated weight if the LP is at a intersection. For more general case, the multiplication of the number of events and the average per event processing time (denoted as T_{lp}) should be used as measure of the load on a LP. T_{lp} represents the proportion of time spent on a LP in a simulation session. We may consider two PEs are of approximately equal load if the total T_{lp} on each of them are equal. This may not reflect the dynamic nature of the network, but experiments have indicated that the partitioning based on T_{lp} gives good parallel performance for network models using our MPEG and Ethernet traffic models. This has been verified by simple pipeline models presented previous sections. The experiments with Wnet were designed to find out if the notions and methods discussed in this report are also justified in more realistic network models.

There are some other facts that need to be considered in the model partitioning. These facts could have significant performance consequences. The following is the list of the facts that have been used in the partitioning of Wnet. The list may grow longer as more understanding is obtained about the ATM simulation, the ATM-TN model and the Simkit simulator.

- LP is the basic active element in parallel execution. Time Warp system will force the events for a LP to be processed in non-decreasing receive time order. So one important principle in the design of parallel version of ATM-TN is to form a separate LP for each logically related (implying time ordered) stream of events. In the sequential version of ATM-TN, the traffic source and sink are handled by a single LP. As a result, two independent streams of events, i.e. the sending of cells at the source side and the receiving of cells at the sink side, are forced to be ordered resulting unnecessary rollback caused by cells arriving at the sink. This type of rollback may propagate further along the source-sink pipeline which forms a loop at the closure of source and sink. In the parallel version of ATM-TN, the traffic models have separate LPs for the source and sink.
- When a cell travels in the network, it is carried by an event buffer at

each point it stops for processing and the buffer remains there so that in case of rollback it can be tracked down. A new buffer is picked up from local buffer pool to carry the cell to next stop. We see a picture of "relay" of buffers here which may cause undesirable cache effect. So the assignment of LP to PE in partitioning should result in minmum number of PEs a cell has to visit on its way from source to sink.

- As we have found in the simple pipeline experiments, the simulation of ATM-TN models with Time Warp based simulator is very sensitive to load balance which has to do with cell flow in the network. We will discuss this in more detail in the following section. Without proper flow control, either the network will be flooded by cells or it becomes "dried" out. In former case, the buffer pool at the source side will be persistently exhausted and global control mechanism stressed resulting in much higher system overhead on the PEs where traffic sources reside. In the latter case, there will be few cells pending for processing in the network, the statistical fluctuation of cell receive time will cause greater number of rollback at the output ports of switches. Experiments show that the former has more adverse affect on the performance, therefore the load should be shift toward the traffic source side. That is higher load should be placed on those PEs where traffic sources reside.

- As a specific requirement for Warpkit, the underlying Time Warp kernel of Simkit, the source LP and sink LP of the same traffic model should be assigned to the same PE. The buffer flow can then be naturally balanced as the source sends out a cell on a buffer will normally be compensated by receiving a cell along with the buffer that carries it to the sink. At the switch ports, the common case is to send a cell only when receiving one which does not result in neither loss nor gain of buffers. Otherwise, massive buffer redistribution could be the cause of higher system overhead.

- In ATM-TN model, the traffic sources usually generate cells in batch and they never receive event from outside except for external events (warmup, report, etc.). To prevent traffic sources from generating cells too fast, always put a traffic source LP and the very next port LP the cells from the source will stop on the same PE. The traffic source will not generate next batch of cells until its neighbour LP finishes processing all or almost all the cells generated in last batch.

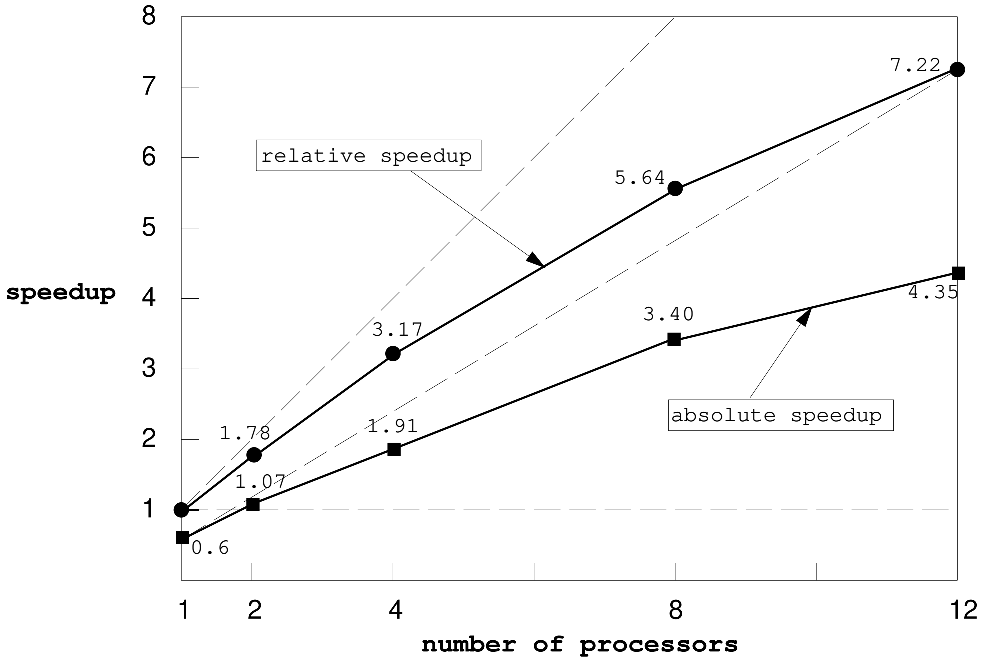
- It is could be a good choice to have same type of LPs held on a PE, if this is not in conflict with other more important considerations. For example the traffic source-sink LP pairs, LPs upstream in pipelines and LPs downstream in pipelines could be three different types of LPs. So the PEs could be arranged in a way that themselves form a pipeline.

- The output ports of switches are where most rollback originate. One effective way to reduce the chance of rollback at these output ports is to have the incoming cells wait for a while in kernel's event lists before being processed by the output ports. The statistical fluctuation of cell's receive time can be eased and less straggler cells will appear at the output ports. To achieve this effect, we need to control cell flow by putting less load on PEs that are located upsteam from the PE where output port LPs exist so that those upstream PEs will be able to send out cells at a higher rate and the cells will be accumulated at the output port LPs. This is obviously in conflict with what we discussed above for easing the stress on global control mechanism.

7.3 Experimental results

In this section we present the summary results of experiments with Wnet. The report facility was turned off in the tests as Simkit_par 1.0 does not have the output mechanism implemented. The output design is a problem as mentioned earlier in this report, but not very bad. The shere number of cell events simply obscures the high cost of output. Tests with raw output indicate up to 5% rollback rate and 10% increase in per event time.

The available parallelism for Wnet is around 12, so there is no point to run the simulation on more than 12 processors. We would also to know what is the maximum possible speedup for Wnet running under the parallel simulator, Simkit_par 1.0. The relative granularity of Wnet is a bit higher than 1 on single processor, that is:



Parallel execution:

number of processors	average per event time	rollback rate
1	34.88 uS	0%
2	19.60 uS	0%
4	11.01 uS	0.48%
8	6.18 uS	1.68%
12	4.83 uS	0.74%

* simulator: Simkit_par 1.0
* number of buffers: 20,000 – 25,000
* simulation time : 20 Sec.

Sequential execution: average per event time: 21.01 uS.

* simulator: Simkit_seq 1.0 (a simple "bare–bone" sequential simulator)
* simulation time: 20 Sec.

Figure 22: **Wnet experiment results**

$$G = \frac{user\ event\ processing\ time}{system\ overhead} = 1$$

The upper limit of maximum possible speedup can be estimated by:

$$N * \frac{1}{1 + \frac{1}{G}} = 0.5 * N$$

where N is the number of processors involved in a parallel simulation. The actual G when executing on more than one processors will be lower because the system overhead generally increases with the number of processors depending on the architecture of the machine and the design of the simulator. While Warpkit demonstrates good scalability on KSR within the same ring of processors (less than 32), on SGI Power Challenge its system overhead increases significantly with the number of processors. As a result, the maximum possible speedup on multi-processors will be lower than the value given by the above formula.

The partitioning used in the Wnet experiments was done manually based on the method outline above. Figure 22 gives the results of Wnet simulation runs on 1, 2, 4, 8, and 12 processors. Relative speedup is measured against the execution time on one processor with the same parallel simulator while absolute speedup is compared to the execution time with a optimized sequential simulator which has no extra cost other than handling event list. The dashed line going toward upper right in the chart represents linear relative speedup if the hardware and the simulator were perfectly scalable, that is the system overhead remained the same as that on a single processor. The discrepancy between the real relative speedup and the ideal one is caused mainly by the increases in forward execution overhead, send overhead and global control cost (by 50%, 90% and 250% on 8 PEs respectively). The global control cost has the largest increase, but still it accounts for just 25% of the total system overhead. The relatively high overhead in global control indicates the stress experienced by the global control mechanism because the lack of an effective flow control. There is a potential to reduce the global control cost by installing some efficient flow control mechanism. We will revisit this topic in the next section. In general, rollback cost is neglectable, idling time is low (implying good load balance), and the speedup is satisfactory given the granularity and the level of overhead incurred in the underlying Time Warp simulator.

8 Cell Events Flow Control

All the previous experiments have revealed that flow control is more a problem than rollback. In this section we discuss the problem and present a flow control scheme. Preliminary tests with a implementation of this flow control scheme indicate that effective flow control can have significant overall performance gain for ATM cell level simulation models.

8.1 Why need Cell Flow Control ?

In an ATM network, cells are generated at traffic sources, transfered over the network, eventually reach their destination and come out of the network. If the traffic sources generate too much cells, some where in the network the traffic will be jammed and cells dropped. The cell flow problem we are addressing here is not the cell flow in the real network that the model simulates, but a unique phenomenen which occures with Time Warp simulators. In a Time Warp simulator, more than one processor executs in parallel and asynchronously. It is possible that some processors may send out too many events to be handled by other processors that are to receive and to process these events. In such a situation, events will be jammed inside the simulator at those slow processors infinitely if without any restrictions. This is common in ATM-TN simulation models as demonstrated by previous tests programs.

In ATM-TN models, cell events are generated at traffic sources and come out of the network at traffic sinks exactly as what is happening in the real ATM network. If the traffic sources generate cell events at a fast rate, the system will be filled with increasing number of events until the processors, where the traffic source reside, run out of local event buffer pool. The global control mechanism will restrict the traffic sources. Unfortunately, this is expensive as global control mechanism is designed to handle buffer reclaim in normal circumstance - periodic invokation, not for a situation where the processors are in a persistent status of lacking event buffers. By putting more load on processors that hold traffic sources can prevent such situation from happening. However, this could lead to few cell events pending processing in the system, therefore increase the chance of rollback at switch output ports LPs and more frequent idling of processors with lighter load. It is impossible to get the load of all the PEs perfectly balanced in real simulation models.

The cell events flow problem is caused by the nature of the network model - it is an open system. There is no logical link with other entities in the ATM-TN model that can restrict the traffic source LPs. A traffic source schedules cell generation on its own, it never receives any messages which could be linked to the cell generation. Using the concept of Event Thread introduced in section 1, the cell Event Thread is open-ended. The number of cell ETs in the system is not stable, so is the simulation system itself. It is clear that we need to control the cell event flow to make the parallel ATM-TN simulation execute in a stable condition.

8.2 Cell Flow Control Mechanisms

It is very difficult if not impossible to implement flow control at the simulation kernel level, the Simkit. The simulator has no knowledge as to how and where the cell flow becomes a problem. It would be much more efficient and possibly very simple to do flow control at the application level within the simulation control part of the model.

Several schemes have been tried and tested. They fall into two categories. One is to build the flow control mechanism into the model itself with little or no help from Simkit. The other one relies on the collaboration between the model and Simkit. A wait/signal facility has been tested for the second approach. It has been found that "wait" is not in harmony with the optimism of Time Warp and is deaklock-prone, besides, major revision is required for Simkit. The advantage is that it is more general and could be a supplement to existing global control mechanism in Simkit. We will not discuss this subject further.

The requirements of cell event flow control are:

- to stabilize the number of cell events in the system so that the global control mechanism will not become stressed because of the "runaway" of cell events and the cost associated with it eliminated.
- to keep reasonable level of events pending processing in the system, that is the system is neither "flooded" nor "dried", to reduce the chance of statistical rollback and processor ideling.

The basic concept of flow control within the ATM-TN model is to artificially make the cell Event Thread a loop instead of open-ended. The simulation model will then become a closed system. The flow control mechanism is based on the following ideas assuming fixed traffic source-sink pair:

- When the first cell arrives at the the sink side of a traffic model, the pipeline on which the cells travel must be filled to some extent, or there are enough cells for all the LPs involved in that pipeline to process. The flow control mechanism should start functioning at least after first cell events have been seen by the sink LP.
- At initial stage of the simulation, The sink LP keeps monitoring the receive time of the arriving cells and the send time of next cell at the source. If it is found that the source has gone far enough in the future, i.e. the time of the next cell to be sent by the source is greater than the time of last cell received by the sink, the sink LP will take over the sending on behalf of the source and the source LP will stop sending cells. The sink LP will send a new cell for the source only if it receives one or it finds that a cell has been lost in the network. Starting at the moment the sink takes over, the number of cell events in the network becomes constant for the specific traffic source.
- Two parameters are used to regulate the flow control mechanism. They are the maximum and minimum *send distance*, denoted by D_{max} and D_{min} respectively. *Send distance* can roughly be viewed as the differece (in cells) of the send time of next cell, T_s, at the source and the receive time of last cell, T_r, at the sink. Suppose that current cell spacing at

the source is ΔT, then the sink should start sending cell on behalf of the source when the following relation is true:

$$T_s - T_r >= \Delta T * D_{max}$$

Since a LP can not schedule an event with time stamp less than the receive time of current event and there is logically no guarantee that T_s will always be greater than T_r, the sink should try to adjust the send distance by sending batch of cells when the following condition becomes true:

$$T_s - T_r <= \Delta T * D_{min}$$

Under certain conditions, higher value of D_{max} will delay the time when flow control mechanism starts functioning and network will be crowded with more cells which can help reducing statistical rollback. It is also possible that the flow control will never start functioning should the D_{max} be set too high as the source may have gone far enough in the future that the global control mechanism starts to restrict its execution.

8.3 experiment results

We compare the results of tests 6 with and without flow control. It is obvious that significant performance gains can be abtained with an efficient flow control mechanism.

```
======================= TEST 6-4-1 ========================
 Partitioning --  PE0: Src1, End1-Out, Snk1. - 103.28
                  PE1: Src2, End2-Out, Snk2. - 103.30
                  PE2: SW-In0, SW-In1, End3-In. - 136
  PE3: SW-Out. - 140

 without flow control:
 =====================
 per committed event time: 25.91 uS (abs. 103.65 uS)
 ---------------------------------------------------
         forward overhead: 10.31 uS
            send overhead:  0.28 uS
                   idling:  1.08 uS
          user exec. time:  7.39 uS
        rollback overhead:  0.31 uS
  global control overhead:  6.54 uS

 efficiency: 98.24%
  abortions: 10148

 global control: per PE GVT computations - 42919
                 (PE0: 94684, PE1: 76859, PE2: 79, PE3: 55)
                 no-fossils - 69751
                 spins - 17865

 with flow control:
 ==================
 per committed event time: 22.25 uS (abs. 89.00 uS)
 ---------------------------------------------------
         forward overhead:  7.72 uS
            send overhead:  2.27 uS
                   idling:  3.43 uS
          user exec. time:  7.36 uS
        rollback overhead:  0.05 uS
  global control overhead:  1.42 uS

 efficiency: 99.73%
```

```
 abortions: 0

global control: per PE GVT computations - 55
                (PE0: 55, PE1: 55, PE2: 55, PE3: 55)
                no-fossils - 0
                spins - 0
```

In test 6-4-1, the effect of flow control is so apparent. Global control works in normal condition and its cost has dropped to less than 25% of the cost without flow control. No event abortions are recorded. The forward execution overhead also decreases significantly which has to do with the buffer access cost and is worth further studying. The performance will be much better if the loads on PEs are better balanced. Note that low send time in the test without flow control is false as accuracy for send overhead decreases with the number of abortions. Also, the lower idling time without flow control compensates partially for the high global cost which helps improve overall performance in such a extreme condition. This once again demonstrates the merits of the asynchronous global control scheme adopted by Warpkit. The performance with flow control mechanism installed would be even better if the load were well balanced (the idling time would be reduced as a result).

```
======================== TEST 6-4-2 =========================
 Partitioning --  PE0: Src1, End1-Out, SW-In0, Snk1. - 137
                  PE1: Src2, End2-Out, SW-In1, Snk2. - 137
                  PE2: SW-Out. - 140
  PE3: End3-In. - 68

 without flow control:
 =====================
 per committed event time: 22.27 uS (abs. 89.08 uS)
 --------------------------------------------------
         forward overhead:  7.57 uS
            send overhead:  2.32 uS
                   idling:  1.91 uS
          user exec. time:  7.32 uS
        rollback overhead:  0.00 uS
  global control overhead:  3.14 uS

 efficiency: 100%
  abortions: 0

 global control: per PE GVT computations - 25242
                 (PE0: 59148, PE1: 41740, PE2: 55, PE3: 27)
                 no-fossils - 41243
                 spins - 0

 with flow control:
 ==================
 per committed event time: 20.00 uS (abs. 80.00 uS)
 --------------------------------------------------
         forward overhead:  6.77 uS
            send overhead:  2.09 uS
                   idling:  2.28 uS
          user exec. time:  7.44 uS
        rollback overhead:  0.08 uS
  global control overhead:  1.34 uS

 efficiency: 99.85%
```

```
  abortions: 0

global control: per PE GVT computations - 52
                (PE0: 60, PE1: 60, PE2: 61, PE3: 30)
                no-fossils - 0
                spins - 0
```

The results of test 6-4-2 show the same effect. Global control cost becomes the least part among all the system overheads. We have more than 10% overall performance improvement. However, the rollback rate is relatively much higher and it could be worse in some situations. If the models tend to have higher rate of rollback, with the flow control mechanism suggested here things will get worse which might override the benefit of having flow control. That is the gains in reduced global cost might be lost in increased rollback cost. We will address this problem in next subsection. Figure 23 illustrates the results discussed above.

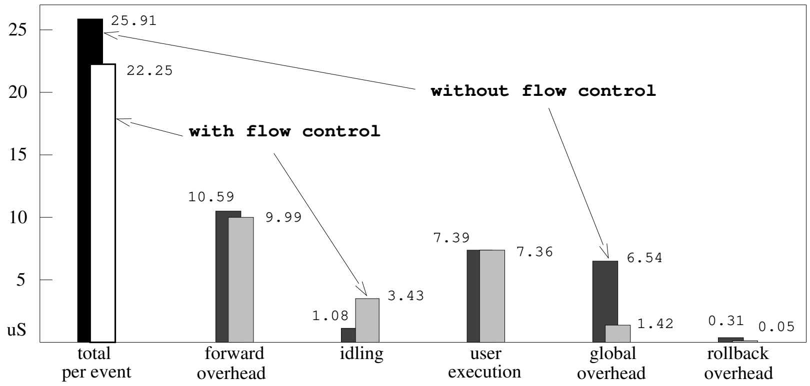
Tests with Wnet are under way. Preliminary results indicate about 10% faster with flow control, which agrees with the results presented here.

8.4 Improve the Mechanism

In test 6-4-2 of previous subsection, rollback rate has increased with flow control in operation. The statistics report of the numbers of rollback for each LP reveals the adverse effect of rollback propagation along the closed cell event threads. Rollback originated at the output port of switch cause the cancellation of events down the pipeline. All the LP on the pipeline are reported to have rollback and the distance of rollback gets longer the farther the LP is from the originator of the rollback. The longest rollback distance reaches 244 events. Without flow control, just a couple of rollback are recorded.

The problem here is caused by the event thread loop. But this is exactly what we want to have with flow control. If a straggler cell arrives at the sink, it will roll the sink LP back, all the cells sent by the sink just on behalf of the source will have be canceled as the underlying kernel, Warpkit, implements

<



TEST6-4-1

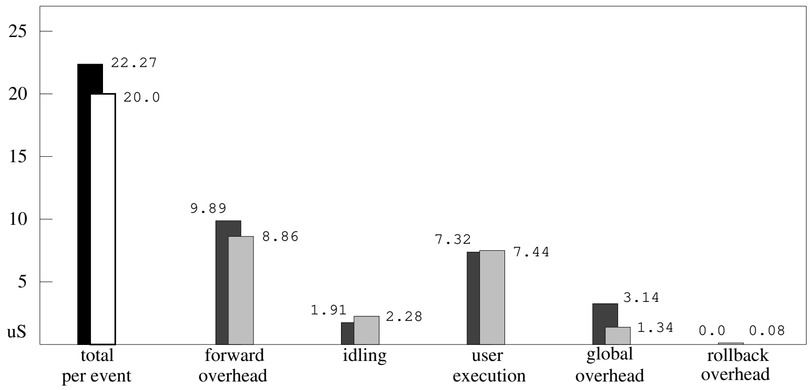
Load:

PE0 103
PE1 103
PE2 136
PE3 140

global control statistics

without flow control:
GVT computation/PE 42919
(PE0 94684, PE1 76859, PE2 79, PE3 55)
no-fossils 69751
spins 17865
abortions 10148

with flow control:
GVT computation/PE 55
no-fossils 0
spins 0
abortions 0



TEST6-4-2

Load:

PE0 137
PE1 137
PE2 140
PE3 68

global control statistics

without flow control:
GVT computation/PE 25242
(PE0 59148, PE1 41740, PE2 55. PE3 27)
no-fossils 41243
spins 0
abortions 0

with flow control:
GVT computation/PE 52
no-fossils 0
spins 0
abortions 0

The Effect of Event Flow Control

Figure 23: **The Effect of Event Flow Control**

aggressive cancellation policy. As we have mentioned before, there is no logical relation between the send time of cells at the source and the receive time of cells at the sink. The cells will be sent at times determined by some predefined random distribution whether or not the sink will get rolled back. If we could break the loop of rollback propagation, we will have the benefits of both stable system and without the adverse effect of rollback propagation along the closed event threads.

An simple yet very effective method is to introduce a function call at Simkit interface. The function marks the event to be sent as “no cancellation”. The kernel will not enqueue these marked events into the output queue of the LPs that send those events and they will not be canceled in case of rollback.

9 Conclusions

1. The cell events generated by traffic sources form Event Threads that will end at their respective traffic sinks. It is possible that some traffic sources may flood the network with infinitely large number of events (or ETs). The ATM-TN model is an open system. With such a system, the event flow control become an issue and the system is generally unstable. If the ETs in atm-tn model form a loop instead of open-ended, then the model will become a closed system and the event flow control will no longer be an issue. One possible way to do this is to merge the traffic source Lp with its sink LP and to generate a cell event only when receives a cell or is informed of a cell being dropped. However, the adverse effect of rollback propagation must be solved.

2. The determined future events (warmup events, report events, and other possible external events) are not fitted to parallel execution. We need a better mechanism to handler external events including outputs.

3. An somewhat accurate partitioning method is crucial to parallelizing ATM-TN models. The partitioning method outlined in this report gives satisfactory performance results. We are close to theoretically maximum possible speedup with current implementation of ATM-TN and Simkit.

4. The system overhead of the underlying simulator must be small enough in order to get reasonable speedup. Simkit_par needs a major optimization to get higher speedup.

5. For Simkit_par to perform well, the number of rollbacks must be brought under control. The statistical rollbacks might becomes another major obstacle in preventing speedup. In the ATM-TN cell level simulation the trouble spots are the output ports where all the incoming cells are merged onto same outgoing links. Statistical fluctuation in the receive time of incoming events may cause those output port LPs to roll back at fine grain (experiments with Wnet indicated average rollback distance of less than 3 events). Buffering incoming events in front of output port LPs might overcome the problem.

6. More understanding about the cache effect of the parallel platform is necessary to improve the scalability.

7. It seems that Time Warp works well in ATM cell level simulation. Three major costs of Time warp system can be minimized with ATM simulation. Rollback rate could be extremely low so the cost associated with rollback is neglectable; With efficient global control and flow control, global control cost will be bearable; The amount of State Saving is not much in ATM model, so is the cost of Incremental State Saving (Note, the ISS cost is included in the user execution time in all the experiment results presented in this report). We still have a lot of potential in further reducing the system overhead of Warpkit. Also there are great potential in ATM-TN model to increase the granularity which could increase speedup significantly.

8. Is Time Warp the only choice ? There might be some other approaches which will be able to incur less system overhead therefore deliver better performance.