UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

EDSON LEMES DA SILVA

GERENCIAMENTO DESCENTRALIZADO DE TEMPO VIRTUAL GLOBAL EM SIMULAÇÃO DISTRIBUÍDA

CHAPECÓ
2018

EDSON LEMES DA SILVA

GERENCIAMENTO DESCENTRALIZADO DE TEMPO VIRTUAL GLOBAL EM SIMULAÇÃO DISTRIBUÍDA

Trabalho de conclusão de curso de graduação apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Dr. Braulio Adriano de Mello

Co-orientador: Bel. Ricardo Parizotto

CHAPECÓ
2018

Lemes da Silva, Edson

Gerenciamento descentralizado de tempo virtual global em simulação distribuída / por Edson Lemes da Silva. – 2018.
76 f.: il.; 30 cm.

Orientador: Braulio Adriano de Mello
Co-orientador: Ricardo Parizotto
Monografia (Graduação) - Universidade Federal da Fronteira Sul, Ciência da Computação, Curso de Ciência da Computação, SC, 2018.

1. Simulação Distribuída. 2. Tempo Virtual Global. 3. Gerenciamento de GVT. I. Adriano de Mello, Braulio. II. Parizotto, Ricardo. III. Título.

© 2018
Todos os direitos autorais reservados a Edson Lemes da Silva. A reprodução de partes ou do todo deste trabalho só poderá ser feita mediante a citação da fonte.
E-mail: edsonlemesds@gmail.com

EDSON LEMES DA SILVA

GERENCIAMENTO DESCENTRALIZADO DE TEMPO VIRTUAL GLOBAL EM SIMULAÇÃO DISTRIBUÍDA

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Braulio Adriano de Mello

Este trabalho de conclusão de curso foi defendido e aprovado pela banca em: 02\07\2018

BANCA EXAMINADORA:

Dr. Braulio Adriano de Mello - UFFS

Bel. Ricardo Parizotto - UFRGS

Dr. Claunir Pavan - UFFS

Dr. Emílio Wuerges - UFFS

RESUMO

Um modelo de simulação distribuída pode suportar a cooperação entre componentes síncronos e assíncronos. Quando os componentes são distribuídos, controles adicionais são necessários para manter a sincronização entre eles, e isto pode ser feito através do conceito de tempo virtual. Nesta ideia, cada componente possui um tempo virtual local (LVT) definido pela ocorrência dos eventos. Como forma de manter o sincronismo dos eventos entre os componentes, é necessário um tempo conhecido por todos, denominado de tempo virtual global (GVT). Esse tempo pode ser definido como: o menor *timestamp* entre as mensagens em trânsito e os LVTs dos componentes. Na simulação síncrona, o GVT define um estado de progresso, impedindo que os eventos cujo o tempo seja menor que o GVT possam ser executados, para que violações de tempo sejam evitadas.

O gerenciamento do GVT pode ser feito de forma centralizada ou descentralizada. A primeira abordagem define um componente como gerenciador de GVT, que recebe informações de controle dos participantes da simulação e calcula o tempo global baseado nelas. Essa forma de gerenciamento torna o controle dependente do processo central, o que pode impor alguns obstáculos, tais como: a existência do ponto único de falha, isto é, na dificuldade de comunicação com o processo controlador, o cálculo de GVT pode ficar comprometido. A segunda abordagem trabalha de forma descentralizada, permitindo que todos os componentes participem da estimativa do tempo global. A distribuição sobre o controle de GVT requer estruturas adicionais que permitam aos componentes manter registros sobre os participantes da simulação, a fim de conseguir estimar o tempo global de simulação baseado nas informações locais disponíveis. Este trabalho apresenta uma estratégia descentralizada para o cálculo de GVT, a partir de soluções correlatas, permitindo uma distribuição da carga de controle.

Palavras-chave: Simulação Distribuída. Tempo Virtual Global. Gerenciamento de GVT.

ABSTRACT

A distributed simulation model can support the cooperation between synchronous and asynchronous components. When components are distributed, additional controls are needed to maintain synchronization between them, and this can be done through the concept of virtual time. Each component has a local virtual time (LVT) defined by the occurrence of events. As a way to keep the synchronization of events between components, a time known by all is necessary, called global virtual time (GVT). The GVT can be defined as the lowest timestamp between messages in transit and the LVTs of the components. In the conservative simulation, the GVT defines a state of progress, preventing events whose time is less than the GVT can be executed. In this way, violations can be avoided.

The management of GVT can be done in a centralized or decentralized way. The first approach defines a component as a GVT manager, which receives control information from the participants of the simulation and calculates the global time based on them. This form of management makes control dependent on the central process, which can impose some obstacles. An example of this is the existence of the single point of failure. In the difficulty of communication with the controller process, the GVT calculation can be compromised. The second approach works in a decentralized manner, allowing all components to participate in the global time estimation. The distribution over the GVT control requires additional structures that allow the components to keep records on the simulation participants in order to be able to estimate the global simulation time based on the available local information. This work presents a decentralized strategy for the GVT calculation, based on related solutions, allowing a distribution of the control message load.

Keywords: Distributed Simulation, Global Virtual Time, GVT management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CMGVT	*Continuously Monitored Global Virtual Time*
DCB	*Distributed Co-Simulation Backbone*
DCBK	*Distributed Co-Simulation Backbone Kernel*
DCBR	*Distributed Co-Simulation Backbone Receiver*
DCBS	*Distributed Co-Simulation Backbone Sender*
GVT	*Global Virtual Time*
LCC	*Local Causality Constraint*
LVT	*Local Virtual Time*
MM	*Matriz de Mensagens*
TFV	*Tabela de Vetores Forçados*
TVT	*Target Virtual Time*
PL	*Processo Lógico*

SUMÁRIO

1 INTRODUÇÃO

A simulação computacional permite imitar o comportamento de processos e sistemas da vida real através de modelos que representam as suas características. Assim, é possível observar o comportamento da simulação através de uma sequência de eventos executados ao longo do tempo. Os resultados da simulação podem, então, ser utilizados para a compreensão do funcionamento do processo ou do sistema, além da possibilidade de realização de testes.

Uma simulação pode ser executada de forma centralizada ou distribuída. Na primeira situação, toda a execução é feita de forma sequencial em um único ambiente computacional, enquanto a segunda maneira tenta reduzir o tempo de execução da simulação através da distribuição de tarefas entre vários ambientes computacionais. A troca de informações, neste caso, acontece através de uma rede de comunicação (FUJIMOTO, 1999).

Um modelo pode ser dividido em diferentes componentes para que possam ser executados de forma distribuída (FUJIMOTO, 1998). Quando os componentes apresentam diferenças em termos de tecnologia, arquitetura ou linguagens de programação, são ditos heterogêneos. E os componentes que são semelhantes quanto as suas definições ou atributos, são ditos homogêneos (REYNOLDS JR., 1988).

Os componentes podem ser definidos como síncronos ou assíncronos. Na primeira abordagem, elementos síncronos não podem regressar no tempo e os eventos são executados de forma crescente em relação ao tempo. Os componentes assíncronos podem infringir esta restrição de tempo, porém, requerem a existência de mecanismos para tratar violações de tempo, tais como a utilização de *rollbacks* e a criação de *checkpoints*.

Quando os componentes de um modelo são distribuídos, controles adicionais são necessários para manter a sua correção, tais como: gerenciamento de tempos virtuais, tratamento de interfaces dos componentes e gerenciamento de memória. Dentre os controles necessários, o conceito de tempo virtual é essencial para a sincronização dos componentes, isto é, para manter uma ordem de execução dos eventos.

Algumas técnicas clássicas utilizam a ideia de tempos virtuais, que definem um tempo virtual local (LVT) para cada componente da simulação. Cada LVT é definido por um marcador de tempo (*timestamp*), utilizado para manter um progresso local em cada componente da simulação. Um outro conceito é chamado de tempo virtual global (GVT), este mecanismo é utilizado para manter um estado de tempo global e esta marca de tempo baseia-se no menor

LVT entre todos os componentes (JEFFERSON, 1985).

O gerenciamento de GVT em uma simulação distribuída pode ser utilizado tanto para componentes síncronos, quanto para assíncronos. O controle em elementos síncronos serve para definir um estado de progresso na simulação, evitando que os componentes possam executar eventos em um tempo menor que o GVT. O tempo global em elementos assíncronos possui uma finalidade diferente, por exemplo, pode ser utilizado como parâmetro para a eliminação de informações antigas (*fossil collection*). Assim, após calcular o GVT, um mecanismo de gerenciamento de memória pode utilizar o tempo global como referência para a liberação de memória, eliminando os registros sobre os estados (*checkpoints*), cuja marca de tempo seja menor que o valor global calculado (YOUNG; RADHAKRISHNAN; WILSEY, 1999).

O mecanismo responsável por obter o cálculo de GVT pode ser centralizado ou descentralizado (JAFER; LIU; WAINER, 2013). Na primeira abordagem, há um processo central (controlador) que coordena as operações de sincronização de relógio virtual global da simulação. A centralização do cálculo torna a simulação totalmente dependente do controlador, este fato pode impor alguns problemas, tais como: a existência do ponto único de falha, afetando o cálculo de GVT quando há dificuldades de comunicação com o controlador, o que pode comprometer o progresso da simulação. Por outro lado, o cálculo descentralizado permite que o controle seja feito de forma conjunta entre os componentes da simulação. Esta segunda abordagem elimina o ponto único de falha, já que o gerenciamento torna-se distribuído. A principal desvantagem dos algoritmos descentralizados é o aumento da quantidade de mensagens de controle transitadas entre os componentes.

Um exemplo para a computação de GVT é o algoritmo de Samadi. Esta solução utiliza a abordagem centralizada, onde cada componente informa o seu menor *timestamp* e o controlador analisa o valor do GVT e faz um *broadcast* informando o novo valor do tempo global (SAMADI, 1985). Este algoritmo necessita que todas as mensagens trocadas entre os componentes sejam confirmadas, gerando um aumento significativo no número de mensagens de controle. Outra estratégia para este propósito é baseado em estado global, onde usa-se uma "fotografia" do instante de tempo da simulação para obter o GVT, este procedimento é chamado de *Snapshot algorithm*. (MATTERN, 1993). A proposta deste algoritmo é permitir que os componentes salvem informações sobre seus estados, permitindo assim que os componentes, em conjunto, definam um estado global. Desta forma, as soluções que são apresentadas para este propósito, precisam ser analisadas em ambientes de simulação distribuída, por exemplo, no

DCB (*Distributed Co-Simulation Backbone*).

O DCB é descrito como uma arquitetura cujo o propósito geral é o suporte à execução distribuída dos modelos heterogêneos de simulação (MELLO, 2005). O gerenciamento de tempo virtual global no DCB atualmente é feito de forma centralizada, onde existe um componente responsável por receber os LVTs de todos os participantes da simulação e devolver através de *broadcast* o valor do GVT atualizado. Este método possui as desvantagens encontradas nos algoritmos centralizados, devido a total dependência do componente dedicado ao cálculo de GVT.

O objetivo deste trabalho é apresentar uma estratégia descentralizada para o gerenciamento de GVT em simulação distribuída visando obter economia em processamento e comunicação, diminuindo a dependência ocasionada por soluções centralizadas. A partir de trabalhos correlatos, buscou-se a definição de uma estratégia considerando os fatores que influenciam no cálculo de GVT, assim como a existência de recursos que permitam a descentralização. Nesta ideia, o algoritmo CMGVT (DEELMAN; SZYMANSKI, 1996), mostrou-se compatível com as necessidades impostas. O algoritmo especificado, foi implementado no DCB, utilizado como ambiente de teste. A análise de resultados levou em conta os dados estatísticos obtidos através de um estudo de caso baseado em experimentos, a partir da comparação entre a versão centralizado e descentralizada. O foco deste trabalho considera apenas o controle de tempo global em relação aos componentes síncronos.

A partir da escolha do algoritmo CMGVT como referência principal deste trabalho, verificou-se a necessidade de adaptá-la para funcionar sob uma perspectiva conservadora (simulação síncrona), visto que o seu propósito geral é direcionado ao auxílio de exclusão das informações obsoletas em simulações otimistas. Como resultado da implementação do algoritmo no DCB, foram obtidas duas novas versões descentralizadas: uma com o uso do CMGVT e uma outra simplificada. Os experimentos realizados no estudo de caso demonstraram algumas observações, tais como: o aumento no número de mensagens reais transitadas entre os componentes em relação ao algoritmo centralizada (considerando a mesma quantidade de tempo simulado), também houve uma redução do número de mensagens perdidas (aquelas que não são enviadas, pois causariam violações de tempo). Por outro lado, notou-se um aumento no tempo real de simulação e também no número de mensagens de sincronização, devido aos mecanismos empregados na estratégia escolhida.

1.1 Justificativa

A dificuldade de definir um tempo global se dá justamente pelo fato de que a atualização é feita em tempo de simulação, ou seja, o cálculo de GVT é realizado sem que ocorra a parada da execução dos eventos. Quando o tempo global é obtido desta forma, existem alguns problemas que precisam ser considerados, por exemplo: se as informações certas estão sendo consideradas na estimativa do tempo global, tais como as mensagens que estão em trânsito durante a ativação do mecanismo de atualização. Este problema pode interferir na estimativa do GVT, levando a valores menos precisos.

Em alguns ambientes de simulação distribuída, tais como o DCB, o algoritmo que calcula o GVT é centralizado. Nesta ideia, existe a possibilidade de acontecer uma sobrecarga no componente controlador, dificultando a atualização de GVT quando há uma instabilidade sob ele. Deste modo, torna-se fundamental explorar estratégias alternativas sob uma perspectiva descentralizada.

Na bibliografia há diversos trabalhos que discutem estratégias de gerenciamento de GVT (centralizadas ou não). Neste aspecto, o algoritmo CMGVT, com as devidas adaptações, se mostra compatível com o processo de descentralização do cálculo de GVT em componentes síncronos.

A utilização do DCB como ambiente experimental, permitiu a realização da implementação do algoritmo CMGVT, eliminando a gestão centralizada de GVT que estava contida na última versão. Deste modo, a troca de mensagens de controle passou a ser feita diretamente entre os componentes (de maneira conjunta), permitindo o processo de descentralização do cálculo de GVT.

Como resultado, a descentralização possibilitou aos componentes gerenciar localmente as informações de controle sobre todos os demais participantes da simulação, permitindo que cada um deles tenha uma visão distinta sobre o tempo global. Esse processo implica diretamente no aumento do tempo total necessário e no aumento no número de mensagens de sincronização. Contudo também há constatações positivas, tais como a possibilidade do aumento no número de eventos gerados sob uma mesma quantidade de tempo simulado, além da redução da perda de mensagens, ocasionadas por avanços exagerados no GVT.

2 REFERENCIAL TEÓRICO

2.1 Simulação computacional

A simulação computacional é um recurso que permite construir modelos que representam sistemas ou processos da vida real, com isso, é possível observar o seu comportamento em relação ao tempo. A execução da simulação acontece através de uma sequência de eventos. Desta forma, a utilização de simulação computacional facilita a compreensão dos aspectos de funcionamento do modelo para fins de estudo (BANKS, 1999).

A execução de eventos pode ser caracterizada de duas formas: contínua ou discreta. Na primeira forma, os estados mudam continuamente conforme a execução dos eventos, enquanto na simulação de eventos discretos, há mudança de estado apenas em instantes específicos do tempo conforme a ocorrência dos eventos (FERSCHA; TRIPATHI, 1994).

2.1.1 Simulação distribuída

Uma simulação pode ser executada em um único ambiente computacional, de forma sequencial. Nesta ideia, os eventos são escalonados de acordo com a capacidade de processamento, o que pode exigir um longo tempo de execução para simulações complexas.

Para reduzir o esforço computacional gerado pela execução de uma simulação, a mesma pode ser definida de forma distribuída. Neste aspecto, as tarefas de simulação são distribuídas entre vários ambientes computacionais. Neste caso, a comunicação entre os elementos acontecem através de uma rede (FUJIMOTO, 1999).

A execução da simulação distribuída necessita da divisão de um modelo em componentes. Assim, cada componente é descrito por um processo lógico (PL). A interação entre os PLs acontece através da troca de mensagens, que podem ser classificadas como: mensagens de evento (simulação) ou de controle.

A simulação distribuída necessita da existência de alguns mecanismos de controle para manter a execução consistente, tais como: gerenciamento de tempos virtuais (JEFFERSON, 1985). Este controle é essencial para manter a sincronização entre os componentes da simulação. Neste ideia, cada componente possui um controle de tempo local, chamado de tempo virtual local (LVT), que é definido por um marcador de tempo (*timestamp*) conforme a ocorrência dos eventos nos componentes. Para o gerenciamento de todos os LVTs da simulação, é

necessário que haja um instante de tempo conhecido por todos os PLs da simulação, isto é, um tempo virtual global (GVT).

A sincronização entre os componentes deve garantir a ocorrência consistente dos eventos conforme os seus respectivos *timestamps*. Neste ideia, os componentes podem ser definidos como síncronos ou assíncronos, podendo haver a cooperação entre eles em uma simulação.

2.2 Simulação síncrona

Em uma simulação síncrona (conservadora), os eventos são executados apenas quando são considerados seguros, ou seja, quando há uma ordem de tempo bem definida para eles, evoluindo conforme o tempo da simulação. Essa evolução deve ser assegurada para que não ocorra uma violação de tempo, denominada de LCC (*Local Constraint Causality*). Esta situação acontece quando um PL recebe uma mensagem cujo o tempo de execução dela é menor que o tempo local do componente (FUJIMOTO, 1999). Caso haja uma violação, a mensagem em questão pode ser considerada perdida, e por consequência, influenciar negativamente no resultado final.

A figura 2.1 descreve uma situação onde ocorre violação de LCC. Neste exemplo, o PL0 envia uma mensagem de evento para o PL1 com a marca de tempo igual a 40, porém o PL1 está com LVT igual a 60. Então, como essa mensagem foi recebida com um tempo menor que o seu LVT, há uma situação de violação de tempo, ocasionando a perda dela.

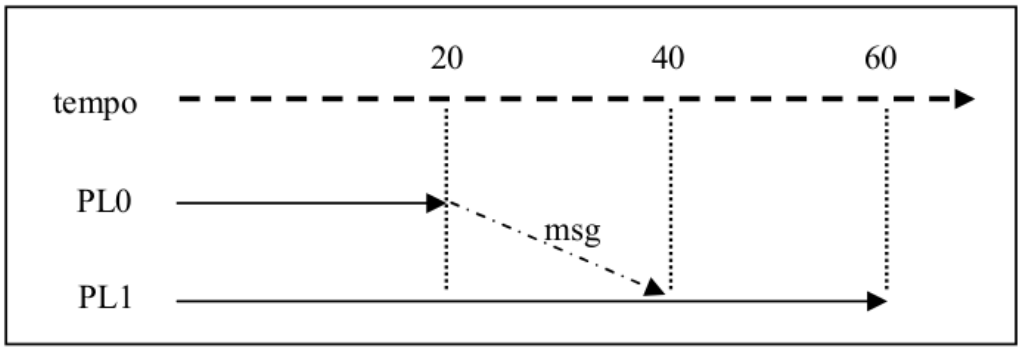

Figura 2.1 – Violação de LCC

Outro conceito utilizado em simulação síncrona é chamada de predição de comportamento (*lookahead*). A sua principal função é atribuir aos componentes a habilidade de predizer quais eventos serão gerados no futuro, permitindo definir um parâmetro de avanço no tempo, dos quais os demais componentes possam utilizar (FUJIMOTO, 1988).

2.3 Simulação assíncrona

A abordagem assíncrona ou otimista, permite a execução de eventos sem a definição de uma ordem temporal, isto é, dá suporte a situações onde possam ocorrer violações de LCC. Neste caso, um mecanismo de retrocesso (*rollback*) é necessário para fazer o tratamento dessa violação. O retrocesso permite que a simulação volte a um estado consistente (antes da ocorrência ser detectada), e reexecute novamente os eventos (FUJIMOTO, 1999).

Para que a simulação possa fazer retrocessos, é necessário que haja o salvamento de informações em um instante de tempo (estados), denominado de *checkpoint*. Deste modo, a simulação pode fazer o *rollback* para um instante de tempo salvo em memória.

Essa situação pode ser exemplificada pela figura 2.1, onde o PL0 envia uma mensagem para o PL1 em um tempo passado, gerando violação de LCC. Neste caso, o PL1 deve fazer um retrocesso (*rollback*) até o tempo 40 (instante que causou a violação), e então reexecutar os eventos a partir deste ponto.

2.4 Gerenciamento de Tempo Virtual Global

Dentre os principais tópicos de interesse em estudos sobre o tempo virtual global (GVT), está a minimização do número de mensagens necessárias para computar este valor. Além disso, a precisão no cálculo ou estimativa do tempo global também apresentam desafios (SRINIVASAN; REYNOLDS JR., 1993).

O gerenciamento de GVT em simulação distribuída pode ser utilizado tanto para componentes síncronos, quanto para assíncronos. O GVT é uma marca de tempo não decrescente que define um estado de progresso da simulação, utilizado para manter a consistência na sua execução. Este tempo pode ser definido como: o menor LVT entre todos os componentes.

Na figura 2.2 está exemplificado a escolha do valor de GVT. Neste cenário, existem três processos lógicos (PL0, PL1 e PL2) com LVTs: 60, 40 e 50, respectivamente. Assim, neste caso o GVT será dado pelo tempo local do PL1, já que é o menor entre todos os componentes.

Nos componentes síncronos, para manter a ordem de execução dos eventos, a simulação pode trabalhar com o valor de GVT como parâmetro para definir uma ordenação dos eventos. Neste aspecto, a definição dos *timestamps* devem levar em consideração o GVT, isto é, o tempo dos eventos devem ser superiores a ele.

O tempo global de simulação também pode ser utilizado nos componentes assíncro-

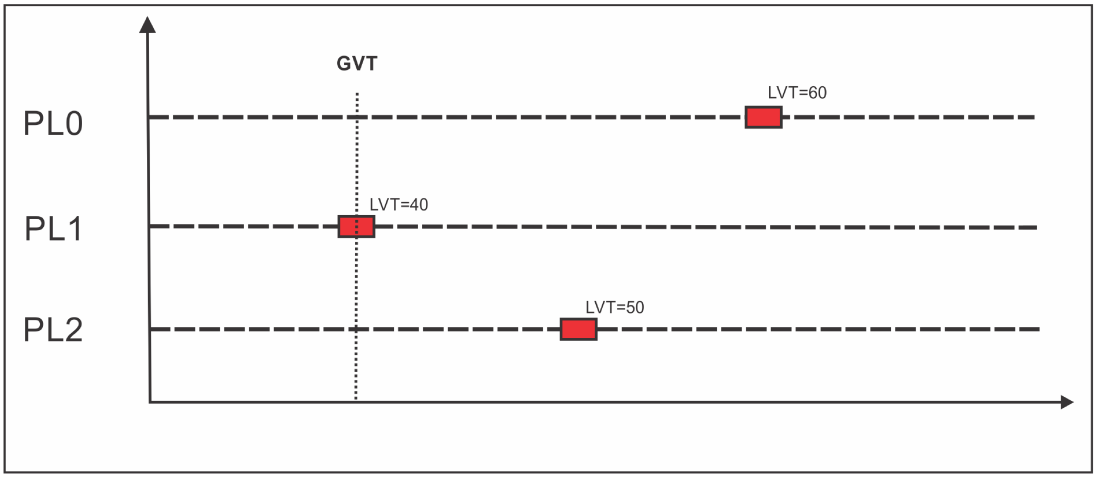

Figura 2.2 – Definição de GVT

nos, neste caso, o GVT é definido como: o menor *timestamp* de qualquer operação futura de *rollback* (FUJIMOTO; HYBINETTE, 1997). A figura 2.3 descreve o GVT em uma simulação assíncrona, onde o tempo global será dado pelo instante tempo igual a 2, já que é a menor marca de tempo de um possível *rollback*. Assim, o GVT pode ser utilizado, por exemplo, como parâmetro para liberação de memória (SOLIMAN, 1998), tais como a eliminação de *checkpoints* antigos, controlado por um mecanismo de *fossil collection*. Deste modo, todas as informações cuja marca de tempo são menores que o GVT serão descartadas (YOUNG; RADHAKRISHNAN; WILSEY, 1999).

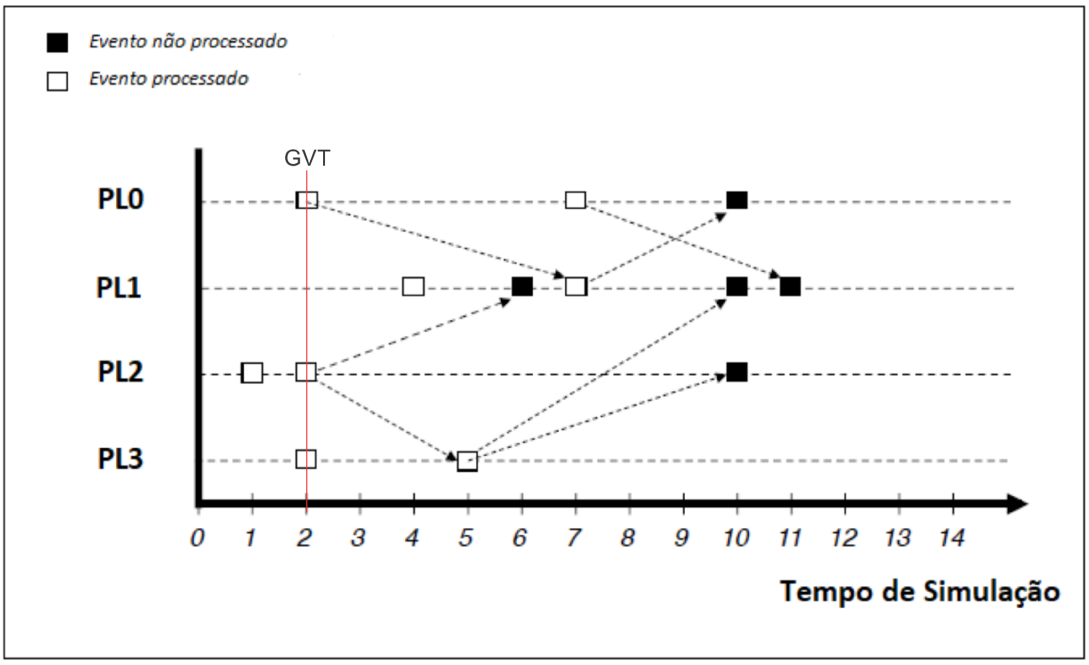

Figura 2.3 – GVT em componentes assíncronos

Para exemplificar a importância de uma boa estimativa no cálculo de GVT na simulação assíncrona, utilizando a figura 2.3 como exemplo: se o GVT for estimado como sendo igual a 7, então, as informações sobre os eventos com *timestamps* menores que ele poderiam ser removidas. Assim, se houver a necessidade de fazer uma operação de *rollback* para algum desses eventos anteriores ao GVT, a mesma não poderia ser concluída, pois já não existe uma referência a esses eventos, podendo comprometer de forma geral a simulação.

Uma possível maneira para calcular uma estimativa de GVT pode ser através da interrupção da execução de uma simulação, assim antes de atualizar o valor de GVT, a simulação para de executar novos eventos. Outra forma possível é calcular em segundo plano (*background*), ou seja, o tempo global virtual é obtido mesmo com a ocorrência dos eventos. Quando o cálculo é feito desta maneira, existem alguns problemas que precisam ser considerados, entre eles, a existência de mensagens em trânsito durante a ativação do mecanismo de GVT. Uma mensagem é dita em trânsito, quando ela foi enviada por um componente, mas ainda não foi entregue no seu destino. Esse problema pode levar a um valor impreciso e afetar o progresso da simulação.

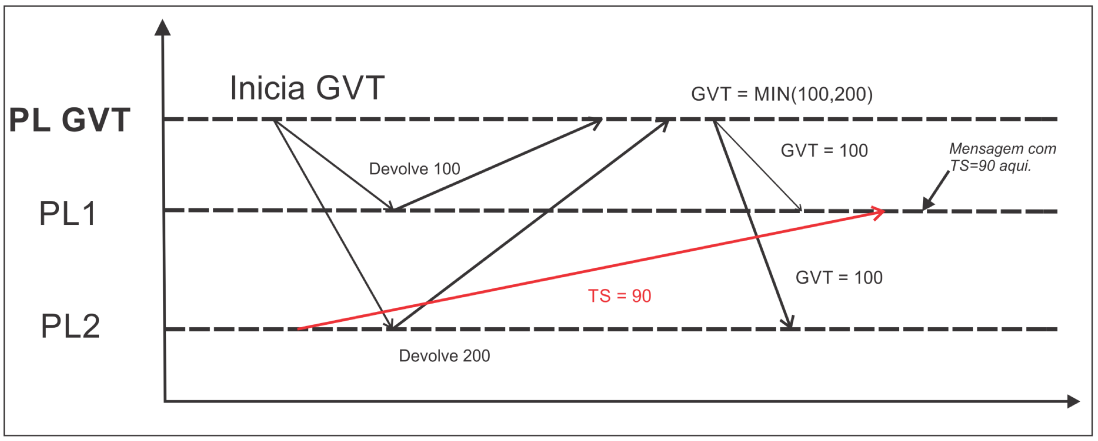

Figura 2.4 – Mensagem em trânsito (FUJIMOTO; HYBINETTE, 1997)

A figura 2.4 exemplifica o problema das mensagens em trânsito, onde o PL gerenciador de GVT inicia o cálculo enviando uma requisição para todos os outros PLs da simulação. Quando o PL1 recebe essa notificação, ele informa que seu LVT é igual a 100, enquanto isso, o PL2 envia uma mensagem de evento para o PL1 com *timestamp* igual a 90, então avança seu tempo local. Posteriormente, o PL2 recebe a notificação de GVT, e responde que seu LVT é igual a 200. Quando o processo controlador recebe todos os LVTs dos componentes, ele determina o valor mínimo global e informa todo mundo. Neste cenário, o GVT está incorreto, pois o valor estimado é igual a 100, quando na verdade deveria ser 90.

Existem alguns recursos que podem ser utilizados para contabilizar as mensagens em

trânsito. Uma possibilidade é a utilização de confirmação das mensagens, assim como ocorre no algoritmo de Samadi. Nesta ideia, é possível armazenar nos componentes o número de mensagens não confirmadas, então, ao atualizar o GVT, estas mensagens devem fazer parte do cálculo. Um outro recurso utilizado é chamado de redução global. Com este artifício, a quantidade de mensagens em trânsito podem ser mensuradas através da diferença entre as mensagens enviadas e as recebidas (CHEN; SZYMANSKI, 2007). A fim de conseguir reduzir o custo da redução global, ela pode ser realizada através de operações com auxílio de *Hardware*, onde as informações ficam disponíveis para os componentes diretamente na memória.

Um outro fator a ser destacado refere-se ao instante da realização do cálculo de GVT, isto é, quando ele será atualizado. Por exemplo, na simulação otimista, o início do cálculo de GVT pode ser ativado através da contagem de eventos, onde após a execução de n eventos, inicia-se a atualização. Um outro parâmetro utilizado, pode ser a quantidade de avanço do tempo virtual, assim, após t unidades de tempo, o GVT é recalculado. Essa segunda opção visa manter o progresso do GVT nos componentes sob uma mesma taxa de atualização (MIKIDA; KALE, 2018). Já na simulação conservadora, o instante deve ser definido pela ocorrência dos eventos, ou seja, quando não há eventos seguros para serem executados, o GVT precisa ser recalculado.

Em relação às abordagens de gerenciamento de tempo virtual global, existem duas, são elas: centralizada ou descentralizada.

2.4.1 Gerenciamento centralizado

A primeira abordagem para o gerenciamento de GVT é dita centralizada, pois o cálculo deste valor é feito por um PL dedicado. Nesta ideia, todos os outros PLs da simulação comunicam-se com ele para fazer a atualização de GVT. Na figura 2.4 está representado o comportamento de um mecanismo centralizado, onde o componente responsável pelo cálculo obtém as informações locais de todos os demais componentes da simulação, e devolve o tempo global virtual atualizado.

Os algoritmos que utilizam essa abordagem de gerenciamento sofrem com o problema de mensagens em trânsito, mencionado anteriormente. Outra dificuldade encontrada, é a existência da sobrecarga no PL destino ao GVT, devido ao excesso de mensagens de controle que este processo pode receber (FERSCHA; TRIPATHI, 1994).

2.4.2 Gerenciamento distribuído

Uma outra forma de gerenciamento de GVT é feito de forma distribuída, onde este valor é estimado em conjunto por todos os componentes da simulação, sem que exista um PL central para o gerenciamento (OVEREINDER, 2000). Entre as vantagens proporcionadas com o gerenciamento distribuído, está a possibilidade do balanceamento de carga referentes ao mecanismo de controle de sincronização (BRAGARD; VENTRESQUE; MURPHY, 2014). Essa constatação sobre distribuição de carga, pode ser estendida aos aspectos relacionados ao gerenciamento de GVT descentralizado, onde busca-se a diminuição da dependência causado pela centralidade do cálculo de GVT através da cooperação entre os componentes.

Algumas soluções propostas para esta abordagem fazem uso de estados globais, através da definição de cortes consistentes. Este método é chamado de *snapshot algorithm*. A sua principal ideia é definir um instante de tempo e tratá-lo como um corte temporal, assim pode haver a classificação dos eventos entre: passado ou futuro.

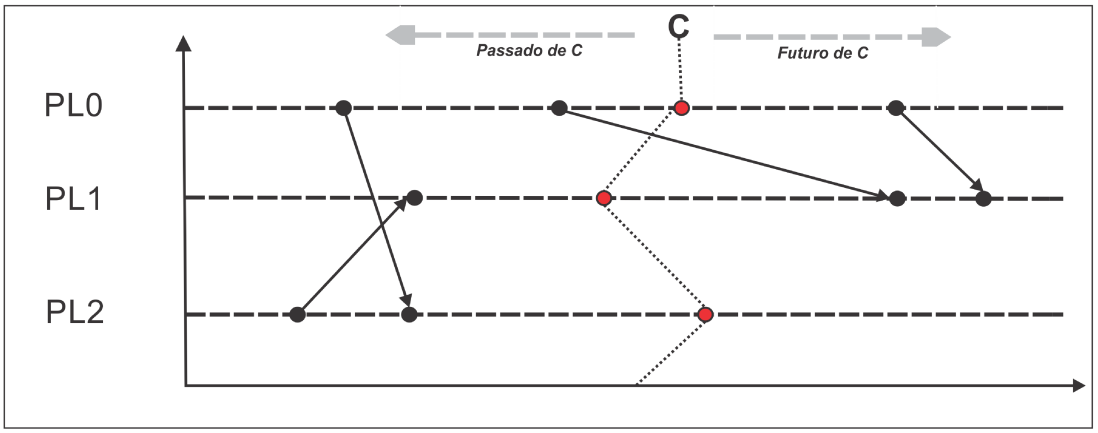

Figura 2.5 – Utilização de corte (MATTERN, 1993)

A figura 2.5 demonstra a definição de corte utilizada como parte das estratégias dos algoritmos de GVT baseado em estados globais. Os eventos que ocorrem antes do corte C, são ditos eventos passados, e após C são os eventos futuros. A consistência de um corte se dá quando não há nenhuma mensagem enviada de um tempo futuro para um tempo no passado.

A utilização deste tipo de algoritmo permite eliminar a necessidade de confirmação das mensagens transitadas, devido a participação de todos os PLs no gerenciamento. Por outro lado, aumenta a complexidade de controle e quantidade de mensagens necessárias para o gerenciamento de GVT.

2.5 Distributed Co-Simulation Backbone

O DCB é descrito como uma arquitetura cujo o propósito geral é o suporte à execução distribuída dos modelos heterogêneos de simulação (MELLO, 2005). O DCB é utilizado neste trabalho como ambiente de implementação e avaliação.

A estrutura interna do DCB é composta por quatro módulos principais: o Expedidor (DCB *Sender* - DCBS), o Receptor (DCB *Receiver* - DCBR), o Núcleo (DCB *Kernel* - DCBK) e o Gateway. A figura 2.6 descreve a organização dos módulos do DCB.

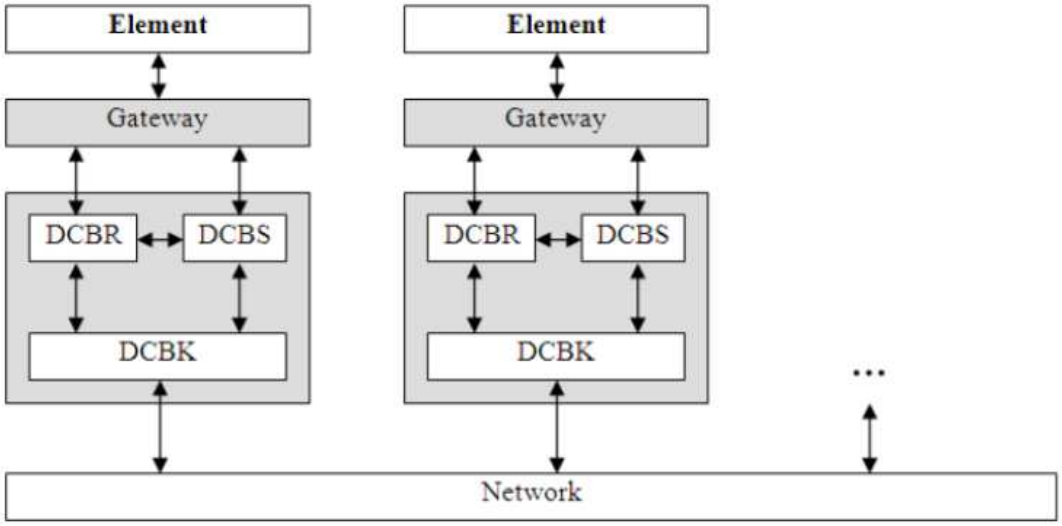

Figura 2.6 – Arquitetura do DCB

A principal função do *Gateway* é o tratamento das interfaces dos componentes, isto é, oferecer traduções de um formato de entrada para um formato de saída conforme as exigências de cada elemento pertencentes à simulação.

O módulo DCBR é responsável pela gerência das mensagens recebidas através do *Gateway*. Internamente, ele decodifica os pacotes recebidos para um formato de controle específico. Este módulo também ajuda a definir o tempo de simulação em relação às atividades.

O DCBS gerencia as mensagens enviadas, fazendo os tratamentos necessários para o envio. Além disso, juntamente com DCBR, mantém controles internos que permitem definir o valor de LVT para fins de sincronização.

O DCBK mantém os serviços de comunicação por troca de mensagens, além do controle de GVT do qual este módulo é responsável.

2.5.1 Sincronização no DCB

O DCB utiliza o conceito de tempo virtual para a sincronização das mensagens, definidas através de marcas de tempo (*timestamp*). Deste modo, cada um deles representa o tempo em que um evento deve ocorrer no elemento destino.

Outra característica do DCB é o suporte à simulação híbrida. Nesta ideia, podem haver componentes síncronos e assíncronos cooperando entre si. Além destas duas abordagens, o DCB permite empregar componentes *notime*, dos quais não utilizam uma definição explícita de tempo.

Nos componentes síncronos, apenas as mensagens cujo o *timestamp* seja maior que o GVT poderão ser enviadas, como forma de garantir a ordenação temporal das mensagens, evitando assim, violações de tempo (LCC). Por outro lado, nos componentes assíncronos não há restrição de envio das mensagens, mas para garantir a consistência da simulação, é necessário que ocorra um *rollback* sempre que acontecer uma violação de tempo. Os elementos *notime* apenas organizam suas mensagens de acordo com a ordem de envio.

A estimativa do GVT é mantido pelo DCB com base no tempo local de simulação (LVTs). Nesta ideia, o valor é dado pelo menor LVT entre todos os componentes. Já que o DCB utiliza o conceito de *lookahead*, o cálculo do GVT para os componentes síncronos é dado por: GVT + *lookahead* (MELLO, 2005). O DCB utiliza uma abordagem centralizada para o gerenciamento de tempo virtual global, onde existe um componente responsável por estimá-lo.

3 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados alguns trabalhos correlatos que discutem estratégias propostas para o gerenciamento de tempo virtual global.

Em (SAMADI, 1985), o autor apresenta um algoritmo centralizado para cálculo de GVT, onde há um componente dedicado para gerenciar o tempo global de simulação. O funcionamento do algoritmo é simples, o controlador de GVT emite uma mensagem para todos os componentes via *broadcast*, informando o início do cálculo de tempo global. Assim, cada componente reporta o seu tempo local e o controlador determinada o tempo mínimo global e repassa o valor atualizado para todos os componentes da simulação.

O algoritmo de Samadi trata do problema de mensagens em trânsito, que podem levar ao controlador estimar um valor incorreto de GVT. Para isso, a sua estratégia utiliza um mecanismo de confirmação (*acknowledgement message*) para cada mensagem recebida. Deste modo, caso haja uma mensagem em trânsito, ela será considerada no cálculo de GVT. Esse algoritmo utiliza *broadcasts* para a sincronização do tempo global, e isso exige várias trocas de mensagens. O custo das mensagens de confirmação também é um ponto negativo, já que é necessário saber se todas as mensagens foram entregues aos seus destinos.

Lin e Lazowska (LIN; LAZOWSKA, 1990) apresentam um algoritmo semelhante ao de Samadi. Entretanto, esta estratégia possui uma otimização, ela não usa mensagens de confirmação para todas as mensagens transitadas. Ao invés disso, utiliza uma sequência numérica para reduzir a quantidade de mensagens de confirmação transmitidas. Para isso, as mensagens enviadas de um componente para o outro são marcadas com números sequenciais. Deste modo, quando o algoritmo de GVT é iniciado, cada componente notifica seus vizinhos informando a última sequência que cada um conhece, assim as sequências maiores que as recebidas são assumidas como mensagens em trânsito, e devem ser consideradas para o cálculo de GVT.

Outra estratégia foi descrita por Bauer e Sporrer (BAUER; SPORRER, 1992). Este algoritmo utiliza um gerenciador central para o GVT que recebe informações de outros componentes da simulação. Nesta ideia, o algoritmo contabiliza as mensagens transmitidas; cada componente administra o número de mensagens enviadas e recebidas. Assim, periodicamente cada componente envia suas informações locais (número de mensagens contabilizadas e o LVT) para o gerenciador, que deduz a partir dos dados recebidos o menor LVT (considerando as possíveis mensagens em trânsito) para então poder estimar o GVT.

O algoritmo apresentado por Bauer e Sporrer utiliza uma abordagem passiva, ou seja, o componente responsável pelo GVT espera que os demais componentes encaminhem suas informações locais de forma independente. Essa abordagem adotada permite que o controlador estime um valor de GVT com uma quantidade reduzida de mensagens, pois o gerenciador de GVT não precisa requisitar nenhuma informação dos outros componentes.

Em (D'SOUZA; FAN; WILSEY, 1994), os autores apresentam o pGVT (*passive response GVT*). Esse algoritmo trabalha em ambientes onde falhas de transmissão de mensagens possam ocorrer. Nesta abordagem, existe um gerenciador de GVT que recebe informações dos demais componentes. Assim como na solução de Bauer e Sporrer, este algoritmo também assume que o controle é passivo. Cada componente monitora o avanço do GVT, e apenas envia informações para o gerenciador quando for necessário. Para isto, os componentes mantêm um histórico local de progresso do GVT, que permite a cada um deles determinar quando um LGVT (Local GVT) deve ser estimado e reportado ao controlador. Posteriormente, o controlador recebe o LGVT e determina o valor estimado para o GVT e distribui o resultado para os demais componentes.

Tomlinson e Garg (TOMLINSON; GARG, 1993) apresentam um algoritmo que minimiza o tempo de latência no cálculo de GVT (tempo entre a ocorrência e detecção). Nesta ideia, a entrega das mensagens nos seus destinos não precisam ser confirmadas, o que diminui o *overhead* de comunicação. Esta estratégia trabalha com a especificação de um tempo virtual alvo (TVT). Assim, algum componente inicializa um TVT, cujo o valor é igual ao seu LVT e então envia uma mensagem *target* para os outros componentes, o *timestamp* desta mensagem é igual ao TVT atribuído pelo componente que a enviou. Quando um processo recebe essa mensagem, ele reporta as suas informações locais para o componente que inicializou o TVT. Deste modo, ele pode determinar se o GVT precisa ser atualizado.

O principal objetivo do algoritmo descrito por Tomlinson e Garg é a liberação de memória, através da eliminação de *checkpoints* criados para operações de *rollbacks* em componentes assíncronos. O GVT, neste caso, serve como parâmetro utilizado para remoção destas informações, permitindo que o gerenciamento de memória retire os *checkpoints* cujo os *timestamps* sejam menores que o valor calculado.

Mattern (MATTERN, 1993) apresenta uma estratégia sem a utilização de um controle central para o gerenciamento de GVT. Este método baseia-se na ideia de estados globais e não requer a confirmação explícita das mensagens transitadas. O objetivo deste procedimento é

definir dois cortes na linha de tempo da simulação, assim o primeiro corte fica responsável por delimitar a contagem das mensagens em trânsito. Caso nenhuma mensagem tenha sido enviada de um componente ao outro através deste corte, então este é um ponto seguro para estimar o GVT; se ainda há mensagens em trânsito (aquelas que cruzaram o primeiro corte), então é necessário ativar o segundo corte, este por sua vez precisa esperar que todas as mensagens enviadas sejam entregues, isto é, cheguem ao destino antes de ultrapassarem o segundo corte. Deste modo, o GVT pode ser estimado, caso contrário é preciso fazer uma nova tentativa.

O algoritmo proposto por Mattern elimina a necessidade de confirmação das mensagens. No entanto, exige um controle mais rigoroso devido a descentralização do cálculo de GVT. O algoritmo requer que haja uma maneira de sincronização entre os componentes, com o intuito de definir os cortes ao logo da linha de simulação. Neste caso, os cortes podem ser definidos através da utilização de *broadcasts* a partir do inicializador ou alguma outra maneira de comunicação entre os componentes. A descentralização nesta estratégia, permite aos componentes salvar informações locais sobre os participantes da simulação (oriundas de um *token* de controle repassado entre os componentes) com o objetivo de identificar o número de mensagens em trânsito e realizar uma melhor estimativa possível para o GVT.

A versão apresentada por Mattern demonstra o funcionamento do algoritmo com uma topologia organizada em anel, onde o *token* de controle é repassado entre os componentes até retornar ao PL inicial. Entretanto, este mecanismo pode ser adaptado para outras topologias, por exemplo: a estrutura lógica poderia ser definida em forma de árvore, assim, o primeiro corte seria definido a partir da raiz até as folhas, enquanto o segundo corte ocorreria de forma inversa. Deste modo, o componente raiz receberia o *token* de controle após a garantia de que ele passou por todos os outros participantes da simulação (ELLEITHY; RIZVI; ELLEITHY, 2010).

Em (PERUMALLA; PARK; TIPPARAJU, 2014), os autores apresentam uma estratégia derivada de soluções que buscam definir um estado global, tais como o algoritmo de Mattern. Nesta ideia, a execução da simulação é dividida em "épocas", onde cada uma delas definem um ponto consistente, isto é, onde garante-se que nenhuma mensagem é enviada para uma época anterior (no tempo passado). Internamente, essa estratégia determina a quantidade de mensagens em trânsito através da diferença entre a quantidade enviada e recebida. Assim, através de redução global, essa contagem é utilizada para definir as mensagens em cada uma das épocas. Quando a redução foi feita em todos os PLs, ou seja, quando a diferença entre as mensagens enviadas e recebidas é igual a zero, o GVT pode ser estimado pelo menor *timestamp*

computado até esse momento.

O algoritmo baseado em épocas permite obter uma estimativa segura do GVT. Conforme sua estrutura, pode ser utilizado de forma síncrona (suspende envio de novas mensagens) ou assíncrona (mensagens continuam sendo enviadas), e também define a utilização entre simulação conservadora ou otimista. Essa estratégia, assim como a de Mattern, precisa contar as mensagens enviadas e recebidas a fim de determinar quantas delas estão em trânsito, isto é, há a existência de um mecanismo de espera empregado no cálculo de GVT.

Dellman e Szymanski (DEELMAN; SZYMANSKI, 1996), apresentam uma estratégia chamada de CMGVT (*Continuously Monitored Global Virtual Time*). A ideia deste procedimento é conseguir estimar o tempo virtual global através das informações locais nos componente, tais como: o LVT de cada PL da simulação e a quantidade de mensagens enviadas ou recebidas. Este algoritmo é baseado nos conceitos de *vector clock* e *matrix clock*, utilizado para sincronização e ordenação de eventos em sistemas distribuídos (RAYNAL; SINGHAL, 1996). No entanto, ao invés de atribuir *timestamps* para os eventos, estes mecanismos são utilizados como contadores de mensagens. Neste aspecto, o *vector clock* consiste em um vetor de tamanho igual ao número de PLs; a utilização de *matrix clock* representa o conhecimento do qual cada componente tem sobre os vetores dos participantes da simulação (número de mensagens enviadas). Essa estrutura é chamada de matriz de mensagens (MM). Para manter informações sobre os LVTs dos PLs e sobre as mensagens em trânsito, existe uma estrutura adicional, chamada de tabela de vetores forçados (TFV). Esta tabela permite gerenciar o conhecimento sobre o LVT de cada PL, além disso, permite controlar o envio das mensagens, através de um artifício chamado de *Forcing Vector* a fim de conseguir mensurar quando uma mensagem está em trânsito ou não. Um *Forcing Vector* é composto de três informações: *timestamp* do evento, PL destino e o número de mensagens de saída contabilizados pelo componente. Para estimar o GVT, cada componente obtém a menor marca de tempo entre todos os LVTs e os *Forcing Vectors* da TFV.

A estratégia apresentada por Deelman e Szymanski utiliza uma abordagem descentralizada baseada nas informações locais disponíveis em cada componente. Como forma de comunicação, o algoritmo troca mensagens de controle com seus vizinhos, com a intenção de obter dados suficientes para estimar um valor de GVT. Essa abordagem gera *overhead* de comunicação devido ao fato de que as mensagens de controle são propagadas entre os componentes para fins de sincronização. A principal utilização do CMGVT, refere-se a simulação otimista, principalmente no auxílio do gerenciamento de memória.

4 METODOLOGIA

Este trabalho tem origem em estudos correlatos sobre estratégias de gerenciamento de tempo virtual global em simulação distribuída. Através da pesquisa, foi feita a busca de um artifício para gerenciamento de GVT, levando em conta as principais diferenças entre abordagens centralizadas e descentralizadas. Posteriormente, utilizando o DCB como ambiente experimental, foi realizado a adaptação da estratégia escolhida. Os resultados foram coletados através de uma série de testes de simulação, realizados conforme o estudo de caso.

Inicialmente foi feito o levantamento bibliográfico sobre os aspectos gerais de simulação, e em seguida a pesquisa foi direcionada ao gerenciamento de tempo virtual global. Nesta etapa, buscou-se identificar as principais características, problemas e definições que são importantes para o controle de GVT (centralizado ou não).

Em seguida, a partir das informações coletadas na pesquisa bibliográfica, foi definido um algoritmo descentralizado (CMGVT). Essa estratégia escolhida sofreu algumas pequenas alterações na sua forma original, que foram necessárias no gerenciamento de GVT para componentes síncronos, tais como a consideração dos *timestamps* das mensagens e não apenas a contagem sobre elas.

Na etapa seguinte, foi feita uma análise da atual versão do DCB implementada, verificando quais seriam as modificações necessárias. Durante a adaptação do algoritmo escolhido, observou-se a possibilidade de realizar experimentos com uma simplificação da estratégia especificada, gerando assim, uma versão alternativa. A implementação no DCB exigiu que ficasse mantido a antiga definição do *lookahead* estático em cada componente, pois sem ele, o GVT não é atualizado e por consequência a simulação não evolui.

A realização dos experimentos levaram em conta três versões do DCB: a última implementada (utilizando o controle de GVT centralizado), a implementação de um novo algoritmo (com uma abordagem descentralizada) e por fim uma versão simplificada da estratégia escolhida. O levantamento dos resultados foi feito através da execução do estudo de caso para as três versões. A partir dos dados coletados, foi feito a análise dos experimentos e a verificação dos prós e contras de cada situação. Por último, em um experimento reduzido, foram comparadas as versões: original e adaptada do algoritmo escolhido, justificando as alterações feitas.

5 GERENCIAMENTO DESCENTRALIZADO DE TEMPO VIRTUAL GLOBAL

Este capítulo apresenta um estudo sobre estratégias descentralizadas de cálculo de GVT em simulação distribuída. Os métodos escolhidos fazem o uso de mensagens de controle, mantendo a compatibilidade com as características do DCB durante o processo de adaptação do algoritmo escolhido. Como base, a versão utilizada é proveniente do Trabalho de Conclusão de Curso realizado pela Acácia Terra para o cálculo de *lookahead* dinâmico e individual (TERRA, 2017). Deste modo, a partir dele, foram realizadas as adaptações e os experimentos referentes aos métodos selecionados.

A organização do capítulo está definida da seguinte forma: na seção 5.1 estão descritas as informações pertinentes ao cálculo de GVT descentralizado, recursos, vantagens e desvantagens; Já a seção 5.2 foca na explicação de uma técnica de gerenciamento descentralizado, juntamente com a especificação do algoritmo e seu funcionamento; Por último, na seção 5.3 estão relatados os aspectos relacionados a adaptação e implementação dos algoritmos escolhidos no ambiente DCB.

5.1 Estratégias descentralizadas

O principal objetivo do gerenciamento descentralizado é a diminuição da dependência gerada pelo uso de um controle centralizado, permitindo que a gestão de sincronização seja determinada em conjunto pelos componentes da simulação, assim como destacado na seção 2.4.2.

De forma geral, os algoritmos descentralizados tendem a exigir um número maior de mensagens de controle, pois a atualização do GVT depende da disponibilidade das informações de controle em cada componente, o que pode gerar uma demora maior para o avanço (cálculo) do tempo global dependendo do mecanismo empregado para a transmissão dos dados de controle. Algumas soluções utilizam rodadas de sincronização como artifício no cálculo de GVT. A utilização deste recurso permite aos componentes mensurar o GVT de forma segura, podendo ser empregado de vários formas, por exemplo, na busca de um estado global, reduzindo o erro entre o valor real e o calculado (MATTERN, 1993). A dificuldade neste tipo de algoritmo está na quantidade de tempo necessário durante o cálculo de GVT, já que a busca por um estado global exige que não haja nenhuma mensagem em trânsito entre os componentes. Outro recurso

que pode ser utilizado é a propagação de valores, onde a comunicação e o envio de mensagens acontecem diretamente entre elementos vizinhos.

5.1.1 Uso de propagação

O princípio deste artifício é a possibilidade de compartilhar informações locais com os componentes vizinhos, sem que haja a necessidade de comunicação direta com todos os participantes da simulação (BRAGARD; VENTRESQUE; MURPHY, 2014). Deste modo, a propagação permite que as informações de controle sejam disseminadas, proporcionando que elas cheguem a todos os PLs. Essa técnica pode ocasionar um atraso natural na evolução do GVT, pois o seu avanço depende do progresso global da simulação (acesso a informações atualizadas). Nesta situação, cada PL determina localmente o GVT, possibilitando que haja valores distintos em cada componente. Deste modo, ao haver o envio de uma de mensagem de simulação, há a possibilidade de que ocorra uma violação de tempo no componente destino. Para exemplificar esse cenário, na figura 5.1 está descrito três PLs (0, 1 e 2), o PL1 recebeu as informações atualizadas sobre os demais via propagação, então ele pôde evoluir o seu GVT (de 900 para 1000); neste instante o PL2 (com GVT igual a 900) enviou uma mensagem (*timestamp* igual a 980) para o PL1, porém como o *timestamp* dela leva em consideração apenas o GVT do componente origem, então, esta mensagem causará uma violação de tempo, pois a visão do tempo global é diferente para o PL1 (destino).

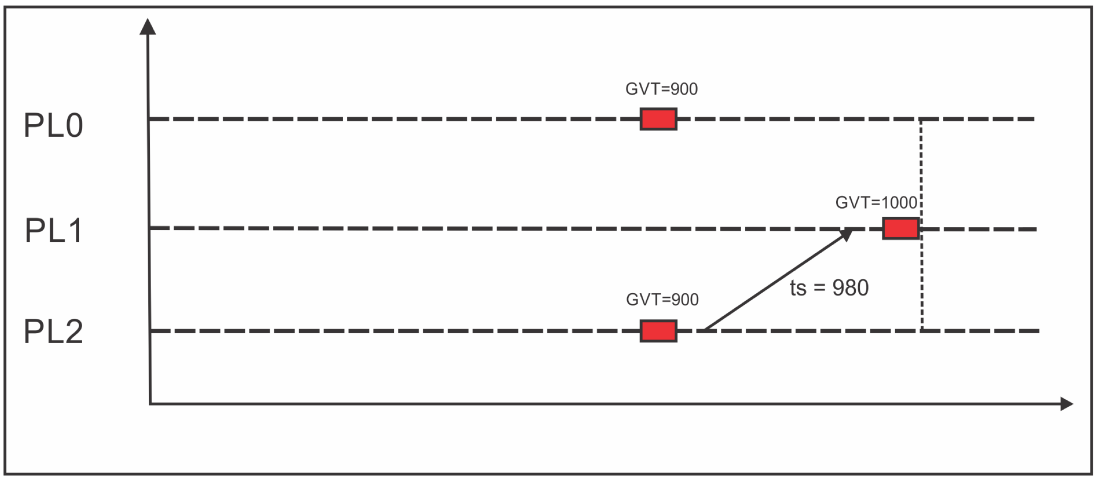

Figura 5.1 – Problema com propagação

Apesar das dificuldades apresentadas no uso de propagação, a mesma foi utilizada neste trabalho considerando as estratégias direcionadas ao gerenciamento de tempo virtual global apenas para os componentes síncronos (conservadores).

5.2 Gerenciador descentralizado de GVT

A estratégia escolhida usa o artifício de propagação, a partir do envio de mensagens de controle diretamente entre os componentes vizinhos.

O algoritmo escolhido é chamado de CMGVT (*Continuously Monitored Global Virtual Time*). Esta estratégia permite a descentralização do cálculo de GVT; também permite que mantenha-se informações sobre as mensagens trocados entre os componentes. O CMGVT é baseado no uso de *Vector clock* e *Matrix clock* (utilizados para ordenação de eventos em sistemas distribuídos). Deste modo, fazendo o uso de tais estruturas, permite ao PL armazenar localmente informações sobre os demais PLs da simulação, e assim, monitorá-las e utilizá-las como parâmetro para o cálculo de GVT (DEELMAN; SZYMANSKI, 1996). É importante destacar que esta estratégia não exige uma rodada de sincronização, apenas a propagação entre os participantes da simulação. Os eventos seguem sendo gerados e enviados enquanto ocorre a atualização de GVT.

5.2.1 Estruturas de controle

Internamente o CMGVT está organizado com a utilização de três estruturas de controle, são elas: Matriz de Mensagens (MM), Vetor Forçado (F) e Tabela de Vetores Forçados (TFV).

A Matriz de Mensagens é responsável por armazenar a contagem de mensagens enviadas pelos PLs da simulação. Nesta ideia, a matriz MM_i representa a estrutura do PL_i; a linha $MM_i(i,i)$ contabiliza as mensagens enviadas pelo próprio componente; enquanto a linha $MM_i(i,j)$ mostra o que PL_i sabe sobre a quantidade que o PL_j enviou; e da mesma maneira, a linha $MM_i(j,k)$ representa o conhecimento indireto armazenado pelo PL_i, isto é, a partir do PL_j, as informações conhecidas pelo PL_k. A matriz abaixo demonstra a utilização dessa estrutura (considerando os índices a partir do zero):

$$MM_0 = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 0 & 0 \\ 0 & 3 & 2 \end{bmatrix}$$

Neste exemplo, há três PLs (PL0, PL1 e PL2). A matriz representa as informações conhecidas pelo PL0. Neste caso, sabe-se que: PL0 enviou 5 mensagens ($MM_0(0,0)$); da mesma forma, as posições $MM_0(0,1)$ e $MM_0(0,2)$ indicam que o PL0 sabe que foram enviadas 3 e 2 mensagens pelos PL1 e PL2, respectivamente. As demais linhas, representam o conhecimento sobre

os outros PLs, por exemplo, a linha 2 mostra que: o PL0 sabe indiretamente que o PL2 têm conhecimento de que o PL1 enviou 3 mensagens ($MM_0(2,1)$), e ele mesmo enviou 2 mensagens ($MM_0(2,2)$). Por outro lado, o PL0 não sabe nada sobre o conhecimento do PL1 (linha 1).

Outra estrutura utilizada é chamada de vetor forçado. O seu objetivo é armazenar informações relevantes às mensagens enviadas. Deste modo, é possível manter um traço sobre as mensagens que são trocadas entre os componentes, auxiliando posteriormente no cálculo de GVT, pois tentam minimizar o erro de estimativa com relação às mensagens em trânsito, como apontados na seção 2.4. Um vetor é formado pelas seguintes informações:

$$F(ts, n, c)$$

Cada vetor contém três parâmetros, o primeiro deles representa o *timestamp* da mensagem; o segundo indica qual o componente origem; e o último mostra a quantidade de mensagens enviadas pela origem até então. Por exemplo, o vetor $F(10, 0, 10)$ indica que uma mensagem foi enviada pelo PL0, com *timestamp* igual a 10 e a quantidade de mensagens enviadas até o momento é igual a 10 (incluindo ela mesma).

A última estrutura a ser descrita representa a tabela de vetores forçados. A sua função é armazenar os tempos locais de cada componente, assim como as mensagens (vetores forçados) enviadas entre os componentes. Nesta ideia, cada PL mantém as informações sobre as mensagens nesta tabela. Por fim, o GVT é calculado a partir desta estrutura, determinando o valor mínimo entre todas as mensagens e os LVTs. Deste modo, a tabela TFV_i representa o conhecimento que PL_i tem sobre as mensagens transitadas entre os PLs da simulação. A linha $TFV_i[i]$ mostra as mensagens enviadas ao PL_i, sendo assim, ficará vazia, pois as mensagens que chegam no componente são automaticamente confirmadas e não precisam ser inseridas nesta tabela; já a linha $TFV_i[j]$ mostra o conhecimento que o PL_i tem sobre o PL_j, seja de mensagens enviadas por ele ou pelos outros componentes. A TFV abaixo demonstra como acontece a utilização desta estrutura:

$$TFV_0 = \begin{bmatrix} 17 & \\ 0 & [F(10,0,4), F(10,2,1)] \\ 19 & [F(9,1,2)] \end{bmatrix}$$

A tabela TFV_0 representa as informações do PL0. Neste exemplo, a linha 0 mostra as mensagens enviadas pelos demais componente ao PL0 (neste caso ficará vazia, pois como

este é o processo origem, não é necessário incluir nenhuma mensagem na tabela). O vetor $F(10, 0, 4)$ na linha 1 mostra que: o PL0 enviou uma mensagem com *timestamp* igual a 10 para o componente 1; o PL0 também sabe que o componente 2 enviou uma mensagem ao PL1 ($F(10, 2, 1)$). A primeira coluna da TFV_0 mostra os LVTs que o PL0 conhece sobre todos os componentes da simulação. Neste caso, também acrescenta-se o seu próprio LVT (linha 0), já que esta estrutura é posteriormente encaminhada aos componentes vizinhos.

5.2.2 Funcionamento do algoritmo

O CMGVT possui dois procedimentos básicos: envio (*sender*) e recebimento (*receiver*). Assim, o procedimento de envio é chamado ao enviar uma mensagem de evento. Já o procedimento de recebimento é chamado quando uma mensagem chega em um componente.

Algoritmo 1: Enviar mensagem ($PL_i \rightarrow PL_j$)

```
1 while LVT < endTime do
2 |   incrementa MM_i(i,i)
3 |   adiciona F(ts,i,MM_i(i,i)) em TFV_i[j]
4 |   TFV_i[i].lvt = lvt do PL.
5 end
```

A função de envio está descrita no algoritmo 1. Antes de colocar a mensagem na fila de saída do componente, primeiramente, é chamado o procedimento *sender*. O algoritmo descreve o envio de uma mensagem do PL_i para o PL_j. Na linha 2 é feito o incremento da contagem de mensagens enviadas pelo componente local; na linha 3, acrescenta-se na TFV_i na posição *j*, o vetor que representa a mensagem em questão; e por último na linha 4 é acrescentado na tabela o LVT do componente origem. Deste modo, as duas estruturas (matriz e tabela) são anexadas na mensagem e enviadas para o destino (PL_j).

Por outro lado, o algoritmo 2 descreve o procedimento executado ao receber uma mensagem. Deste modo, antes de processar a mensagem em si, é necessário atualizar as informações de controle local baseado no que o componente origem encaminhou. Este algoritmo descreve uma mensagem enviada por PL_j e recebida pelo PL_i. A função está divida em três partes: atualização da TFV, dos LVTs e por último a Matriz de Mensagens.

Primeiramente, os vetores presentes na TFV_i (destino) são comparados com a MM_j (origem) nas linhas 2-5, assim, comparando cada vetor $F(ts, n, c)$ em TFV_i com MM_j, se a contagem de mensagens presente em $MM_j(k, n)$ for maior ou igual que o valor c e o LVT para o referido componente em relação ao tempo da mensagem, então este vetor é removido

Algoritmo 2: Receber mensagem ($PL_j \rightarrow PL_i$)

```
1  while LVT < endTime do
2      foreach k in TFV_i do
3          foreach F(ts, n, c) in TFV_i[k] do
4              if MM_j(k, n) ≥ c and TFV_j[k].lvl ≥ ts then
5                  remove F de TFV_i[k]
6              end
7          end
8      end
9      foreach k in TFV_j do
10         foreach F(ts, n, c) in TFV_j[k] do
11             if MM_i(k, n) < c and TFV_i[k].lvt < ts then
12                 adiciona F em TFV_i[k]
13             end
14         end
15     end
16     foreach k in TFV_i do
17         if TFV_j[k].lvt > TFV_i[k].lvt then
18             TFV_i[k].lvt = TFV_j[k].lvt
19         end
20     end
21     Atualiza PL_i baseado em PL_j:
22     MM_i(i, j) = MM_j(j, j)
23     Atualiza PL_i baseado no que PL_j sabe sobre os outros:
24     MM_i(i, k) = max(MM_i(i, k), MM_j(j, k))
25     Atualiza PL_i baseado no conhecimento indireto de PL_j:
26     MM_i(k, l) = max(MM_i(k, l), MM_j(k, l))
27 end
```

da TFV_i, já que isto significa que a origem (PL_j) tem um conhecimento mais recente sobre a simulação (contabilizou mais mensagens que o componente origem, referente aquelas indicadas no parâmetro *n*). Posteriormente (linhas 9-12), é feito o contrário, compara-se a MM_i (destino) com TFV_j (origem); neste aspecto, se há mensagens que o PL_i não conhece (contagem e *timestamp* maior na origem), então, adiciona-se na TFV_i o vetor referente à mensagem em questão.

A segunda parte do procedimento atualiza os LVTs conhecidos pelo componente destino (PL_i). Neste caso, compara-se diretamente o LVT sobre cada componente (linha 17), atualizando apenas se há um conhecimento mais recente (crescente em relação ao tempo).

Por último (linhas 21-26), atualiza-se a MM_i baseado em MM_j. Na linha 22 é feita a

atualização direta da quantidade de mensagens que PL_j enviou; na linha 24 é comparado o conhecimento que PL_j tem sobre todos os outros, então os valores são modificados em MM_i; e na linha 26 atualiza-se as informações que PL_j sabe sobre o conhecimento dos demais componentes (indiretamente), modificando o restante das posições da MM_i que não foram comparadas (índices diferentes de *i* ou *j*).

O CMGVT conforme apresentado por seus autores, foi desenvolvido para utilização em componentes assíncronos, auxiliando na remoção de informações obsoletas. Deste modo, o parâmetro de ativação do mecanismo de cálculo de GVT é controlado de maneira independente dos eventos, isto é, fica a critério de um gerenciador de memória definir o momento que é necessário uma atualização de GVT. Outro ponto a ser destacado é o fato de que o CMGVT assume que as mensagens são entregues na mesma ordem de envio. Por estes motivos, para tornar possível a utilização em componentes síncronos, foram necessárias algumas adaptações no método executado ao receber uma mensagem (algoritmo 2). A primeira delas é em relação as operações de atualização da TFV local. Na linha 4, a segunda condição foi imposta para considerar também os *timestamps* das mensagens em relação aos seus LVTs (sob a visão da origem), para garantir que as mensagens com *timestamp* menores que o LVT sejam removidos, afim de preservar o princípio de ordenação de eventos. Na linha 11 vale a mesma argumentação, ao adicionar um vetor na TFV local, é necessário que o *timestamp* da mensagem seja maior que o LVT em relação ao componente indicado, assim, apenas as mensagens cuja o tempo são superiores ao LVT do respectivo componente serão adicionadas (sob a visão do destino), esta medida visa garantir que apenas mensagens não conhecidas sejam adicionadas, isto é, crescentes em relação ao LVT.

Uma outra modificação realizada foi em relação as condições de atualização dos LVTs dos componentes (linha 17). Na versão original do algoritmo, o LVT é atualizado baseado na contagem de mensagens enviadas, isto é, prevalece o valor do componente que contém um número maior de mensagens enviadas em relação aos outros. Neste caso, como cada componente tem um conhecimento diferente em relação aos LVTs dos participantes da simulação, considerando apenas a contagem de mensagens enviadas, pode acontecer a substituição dos valores de LVT para um valor menor daquele que está presente (conhecimento ultrapassado). Como exemplo: o PL1 sabe que o LVT do PL0 é 500 e o PL2 sabe que o LVT do PL0 é 400. Assim, quando o PL1 recebe informações de controle do PL2, há possibilidade de ocorrer a substituição do conhecimento sobre o LVT do PL0, isto é, passando de 500 para 400. Em outras

palavras, o conhecimento mais recente é sobrescrito. Essa situação influencia diretamente no avanço da barreira do GVT (localmente), pois pode atrasar a atualização dela. Um dos efeitos causados por isso, é o eminente acréscimo no tempo real de simulação. Contudo, buscando alternativas que equilibrem este aumento, preferiu-se comparar diretamente os valores de cada LVT, considerando apenas valores crescentes em relação aos tempos locais.

Conforme essas modificações, observou-se a necessidade de incluir mensagens de sincronização (independente dos eventos de simulação), visto que a sua ausência gera uma dificuldade de atualização de GVT localmente. Deste modo, ao evoluir o LVT, envia-se uma mensagem de sincronização aos vizinhos, contendo as estruturas MM e TFV. Essa ideia permite aos componentes atualizar-se em relação as informações de controle globais necessárias. Por fim, o avanço da barreira do GVT ainda fica dependente das mensagens que chegam nos componentes, seja ela de sincronização ou não.

Passo	**PL0**	**PL1**	**PL2**	**PL3**
0	-	-	-	-
1	$0 \rightarrow 2, ts:2$	-	$2 \rightarrow 0, ts:10$	-
2	$0 \rightarrow 3, ts:4$	$1 \rightarrow 0, ts:5$	-	$3 \rightarrow 2, ts:20$
3	-	-	-	$3 \rightarrow 1, ts:30$
4	-	$1 \leftarrow 3, ts:9$	$2 \leftarrow 0, ts:25$	$3 \leftarrow 0, ts:40$
5	$0 \leftarrow 1, ts:10$	-	-	-

Tabela 5.1 – Exemplo CMGVT

Como exemplo do funcionamento deste algoritmo, considere a tabela 5.1, onde nela está representado o envio e recebimento de mensagens entre os componentes durante cinco ciclos. Para simplificação do exemplo, o LVT de cada PL é considerado igual ao tempo das mensagens (ts).

Analisando como referência o PL2 na tabela 5.1, na inicialização (passo 0), as estruturas de controle (TFV e MM) estão organizadas da seguinte forma:

$$MM_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad TFV_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Seguindo a execução, PL2 envia uma mensagem para o PL0 (passo 1) com *timestamp* igual a 10. Neste instante, ele deve atualizar as estruturas conforme a respectiva mensagem sendo encaminhada ao destino, isto é, aumentar a contagem na matriz referente a origem ($MM_2(2,2)$); e também deve acrescentar o vetor da mensagem na TFV na posição que in-

dica o destino ($TFV_2[0]$) e por fim colocar o seu novo LVT. Deste modo, a atualização ficará assim:

$$MM_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad TFV_2 = \begin{bmatrix} 0 & F(10,2,1) \\ 0 & \\ 10 & \\ 0 & \end{bmatrix}$$

A próxima ação é um recebimento de mensagem (passo 4). Neste caso, o PL2 está recebendo do PL0 ($PL2 \leftarrow PL0$). Seguindo o algoritmo 2, primeiramente é feita a comparação entre a TFV_2 (destino) com a MM_0 (origem). As estruturas em questão estão descritas abaixo:

$$MM_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad TFV_2 = \begin{bmatrix} 0 & F(10,2,1) \\ 0 & \\ 10 & \\ 0 & \end{bmatrix}$$

Neste ideia, compara-se o vetor $F(10,2,1)$ com a matriz do PL0, seguindo a orientação dos índices, ou seja, $MM_0(0,2)$ cujo o valor é igual a zero; então, posteriormente também compara-se o LVT na tabela da origem em relação ao destino ($TFV_0[0].lvt$) com o *timestamp* da mensagem. Como o LVT (4) é menor que o tempo da mensagem (10), ambas as condições não foram atendidas, significando que o PL0 (origem) não tem informações mais recentes do que o PL2 (destino). Portanto, o vetor segue na tabela.

Posteriormente é feita outra comparação, desta vez, considerando o TFV_0 (origem) e MM_2 (destino):

$$MM_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad TFV_0 = \begin{bmatrix} 4 & \\ 0 & \\ 0 & F(2,0,1) \\ 0 & F(4,0,2) \end{bmatrix}$$

Neste caso, o vetor $F(2,0,1)$ é automaticamente confirmado, pois representa a própria mensagem. Assim, resta comparar o vetor $F(4,0,2)$ com a matriz MM_2 no índice: $MM_2(3,0)$, cujo o resultado é igual a zero; e também comparar o LVT do destino em relação a origem ($TFV_2[3].lvt$) com o *timestamp* da mensagem. Como o LVT (0) é menor que o tempo da mensagem (4), ambas as condições foram satisfeitas, isto significa que a origem possui um conhecimento mais recente sobre as mensagens enviados pelos componentes em questão, e o vetor deve ser acrescentado na TFV_2. Na segunda parte do algoritmo, é feita a atualização dos LVTs no PL2 em relação ao que PL0 sabe, levando em consideração o valor superior. E por

último, é atualizado a matriz de mensagens enviadas. Deste modo, após toda a execução do algoritmo, as estruturas do PL2 ficaram assim:

$$MM_2 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad TFV_2 = \begin{bmatrix} 4 & F(10,2,1) \\ 0 & \\ 25 & \\ 0 & F(4,0,2) \end{bmatrix}$$

Portanto, baseado nas informações coletadas, o GVT é dado pelo: menor *timestamp* entre os LVTs e os vetores na TFV. Neste exemplo, o GVT para o PL2 é igual a 0, já que neste instante, o PL2 não tem informações suficientes sobre os componentes 1 e 3.

5.3 Gerenciamento de GVT no DCB

O ambiente de teste utilizado neste trabalho foi o DCB (implementado na linguagem Java). A escolha do algoritmo descentralizado levou em conta a compatibilidade com a atual arquitetura do DCB, conforme descrito anteriormente na seção 2.5.

Atualmente o DCB trabalha utilizando uma abordagem centralizada para o cálculo de GVT. A figura 5.2 demonstra a organização do mecanismo no DCB. Neste ideia, há um componente chamado de *fedgvt* que comunica-se com todos os outros componentes da simulação; deste modo, quando um componente precisa saber o valor atualizado de GVT, ele repassa o seu LVT ao *fedgvt*, este por sua vez determina o valor mínimo e manda uma mensagem com o valor atualizado para cada componente (via *broadcast*). Essa implementação enfrenta os problemas encontrados em estratégias centralizadas. Outra dificuldade observada é em relação a operação de *broadcast*, já que as mensagens de GVT são encaminhadas através de uma fila, um PL pode receber o valor de tempo global antes que um outro PL, fazendo com que momentaneamente, o primeiro componente assuma que a barreira de tempo esteja um pouco a frente dos demais, e as mensagens enviadas nesse período podem causar violações de tempo em alguns casos (semelhante ao problema descrito na figura 5.1).

Para a descentralização do cálculo de GVT, buscou-se estratégias que não modificassem a atual arquitetura do DCB. Dentro desta ideia, o algoritmo CMGVT mostrou-se compatível com a utilização de dois procedimentos básicos: *sender* e *receiver*. Deste modo, as principais modificações foram feitas nos módulos EDCB (ou DCB *Sender*) e EF (ou DCB *Receiver*).

A primeira modificação a ser descrita refere-se na criação de uma nova classe, chamada de *StoreGVT*, nela, estão presentes os métodos e as estruturas que compõem a definição do

tempo global em cada componente. A figura 5.3 descreve os novos atributos que representam a MM, LVTs e TFV.

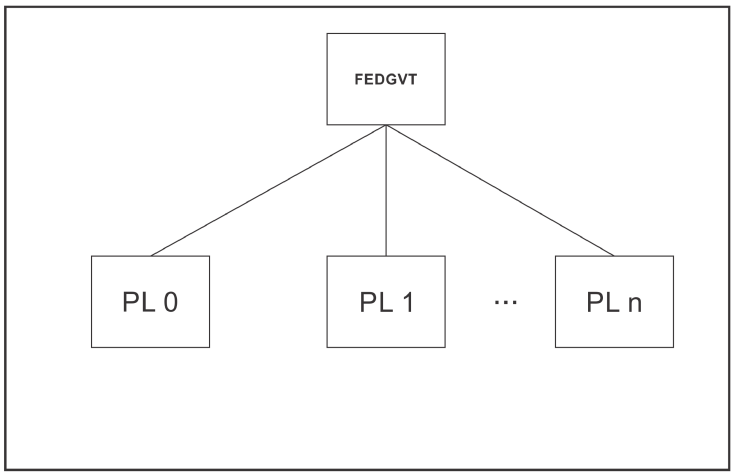

Figura 5.2 – Centralizador do DCB

```
1 public class StoreGVT{
2
3
4         public int[][] MessageMatrix = null;
5         public int[] localLVTs = null;
6
7         ArrayList<String>[] TFV ;
8
9 }
```

Figura 5.3 – Classe GVT

Atualmente no DCB, cada mensagem possui algumas informações de controle referentes aos componentes origem e destino. Para a descentralização do cálculo de GVT, foi necessário acrescentar as mesmas estruturas destacadas na figura 5.3. Deste modo, para cada mensagem estas estruturas são incluídas, caso haja necessidade. Portanto, há um aumento no tamanho geral da mensagem, já que as informações de controle do GVT são requisitadas ao enviar mensagens para um determinado componente.

No DCB, há um parâmetro que identifica cada tipo de mensagem, implementado durante o trabalho do Ricardo Zanuzzo (ZANUZZO, 2017). Esse parâmetro é definido como: real (M), *carrier* (C) e nula (N). Durante a adaptação do algoritmo, observou-se a necessidade de criar mais um tipo, chamado de mensagem de sincronização (S). A sua utilização é requisita no envio das estruturas de controle para os componentes vizinhos. Originalmente, a ideia era enviar mensagens do tipo S apenas quando recebesse uma mensagem de simulação, porém isto afetou drasticamente a simulação em termos de tempo real e consistência, pois algumas execuções

simplesmente não evoluíam.

No expedidor EDCB (*sender*), que é responsável pelo tratamento de envio das mensagens, foram feitas modificações do método *Code*, responsável pela codificação e encaminhamento das mensagens aos destinos. Nela, foram acrescentadas algumas condições de envio, permitindo que as estruturas de controle (MM e TFV) sejam anexadas às mensagens apenas em tipos específicos (M ou S).

Algoritmo 3: Codificação de mensagem

```
1 Function Code():
2     ...
3     if Msg.Type = 'M' ou Msg.Type='S' then
4         Chama expedidor do CMGVT
5         Msg.append(TFV,MM)
6     end
7     ...
8     Send(Msg)
9 End Function
```

O algoritmo 3 descreve o método chamado ao enviar uma mensagem. Primeiramente, verifica-se o tipo da mensagem (linha 3), se ela for mensagem real ou de sincronização, chama-se o algoritmo 1 para atualizar as estruturas de controle, então, posteriormente é feito a inclusão delas na mensagem (linha 5), e somente aí, ocorre o envio de fato.

O outro procedimento está incluído no módulo EF (receptor do DCB), responsável por receber as mensagens de outros componentes e realizar a devida decodificação. Estas ações acontecem mais especificamente no método *Decode*. Para permitir a adaptação do CMGVT neste método, foi necessário acrescentar algumas condições, isto é, ao identificar uma mensagem sendo do tipo M ou S, inicializa-se a atualização das estruturas MM e TFV locais a partir dos dados oriundos da origem.

Algoritmo 4: Decodificação de mensagem

```
1 Function Decode(Msg):
2     ...
3     if Msg.Type = 'M' ou Msg.Type='S' then
4         Chama receptor do CMGVT
5         DCB.updateGVT()
6         ...
7     end
8     ...
9 End Function
```

O algoritmo 4 mostra as instruções seguidas ao receber uma mensagem. Primeiramente,

caso ela for do tipo M ou S, chama-se a atualização das estruturas de controle (linha 4), isto é, através do procedimento descrito no algoritmo 2. Em seguida, ocorre a atualização de GVT (linha 5), para isto, é chamado o método dentro da classe *StoreGVT* através do núcleo do DCB. Esse método busca o menor valor entre os LVTs e os vetores da TFV no componente (estruturas destacadas na figura 5.3). Durante a adaptação do algoritmo no DCB, observou-se a necessidade de manter um valor de *lookahead* estático na atualização de GVT, assim, o método *DCB.updateGVT* retorna o GVT+*lookahead*. Esse valor ficou definido da mesma forma como era anteriormente com o uso do *fedgvt*, igual a 100.

O envio das mensagens de sincronização ocorre quando há uma tentativa de evolução do tempo local em cada componente. Esta tentativa é controlada dentro do módulo receptor (EF). O método responsável é chamado de *updateLVT*; a primeira ação dele é verificar se o LVT pode ser evoluído para o tempo requisito, isto é, se este tempo é inferior ao GVT; caso ele for maior, o LVT deverá ficar na barreira de tempo definida pelo tempo global vigente. O algoritmo 5 mostra como a atualização acontece:

Algoritmo 5: Atualização de LVT

```
1  Function updateLVT(newLVT):
2  |  ...
3  |  if LVT <= GVT then
4  |  |  LVT = newLVT
5  |  else
6  |  |  LVT = GVT
7  |  end
8  |  Atualiza na TFV o novo LVT
9  |  Envia para os vizinhos uma mensagem tipo S
10 |  ...
11 End Function
```

A execução do algoritmo 5 é inicializada quando há necessidade de evolução do LVT no componente. A primeira ação é comparar o novo valor de LVT desejado (linha 3), ou seja, se ele for menor que o valor da barreira de GVT, permite-se o avanço, caso contrário, permanece dependente da evolução de GVT. Em seguida, na linha 8, atualiza o novo valor do LVT na estrutura da tabela (para posteriormente ser enviada). E por fim, na linha 9, é feita a requisição de envio de uma mensagem de sincronização para cada vizinho lógico (ignora-se o sentido das arestas no grafo). As requisições são feitas através de chamadas ao método destacado no algoritmo 3 (*Code*).

Uma derivação do algoritmo CMGVT também foi implementada. Essa segunda versão

foi simplificada: são considerados apenas o envio dos LVTs entre os componentes, desconsiderando as estruturas MM e TFV. Deste modo, a organização de cada mensagem inclui apenas um vetor de tamanho *n* (número de componentes) a mais em relação à estrutura original (versão centralizada). A utilização das mensagens do tipo S ficou mantida.

Primeiramente, a classe *StoreGVT* continua existindo nesta segunda versão, porém, internamente há apenas o vetor de LVTs e os métodos relacionados a atualização de tempo global.

Os métodos *Code* e *Decode* também sofreram algumas alterações. No primeiro deles, ao enviar uma mensagem, se ela for do tipo S, então acrescenta-se o conhecimento que cada componente tem sobre os LVTs dos demais (incluindo o seu próprio tempo local), e não mais para o tipo M. Já no método *Decode*, houve pouca alteração, quando há uma mensagem do tipo S sendo recebida, o componente atualiza o que sabe sobre os LVTs dos outros componentes através da mensagem, assim o procedimento descrito no algoritmo 2 é ignorado. Nesta ideia, se a origem conhece algum LVT maior que o do destino, localmente essa informação é alterada. Após a atualização dos LVTs, calcula-se o valor de GVT baseado neste vetor (dentro do método *Decode*).

A comunicação entre os componentes, isto é, o envio de mensagens de sincronização para os vizinhos ainda acontece quando há evolução de LVT. Assim, é gerado uma mensagem do tipo S para cada vizinho quando há tentativa de evolução do tempo local.

As duas implementações realizadas no DCB eliminam o uso do *fedgvt*. O principal artifício utilizado em ambos é a propagação de valores, o CMGVT buscando enviar o máximo de dados possíveis para cálculos de GVT com melhor precisão. Por outro lado, a segunda versão implementada, busca a simplificação do algoritmo através da redução de conteúdo transitado entre os componentes, já que não anexa a tabela (TFV) e a matriz (MM) em cada nova mensagem enviada.

6 ESTUDO DE CASO

Neste capítulo é apresentado o estudo de caso utilizado para fins de análise em relação aos métodos implementados no DCB para o cálculo de GVT. E ainda, também são descritos os resultados obtidos durante a execução dos testes.

Os experimentos foram realizados em um ambiente computacional com as seguintes configurações: Intel (R) Core (TM) i5-6200U CPU @ 2.30GHZ; 8GB de memória RAM e sistema operacional Ubuntu 16.04 LTS.

Este estudo de caso utiliza um modelo que representa um sistema de troca de mensagens (*chat*) contendo cinco componentes (enumerados de 5 a 9), onde a cada 1 segundo, é realizado uma troca de mensagem entre eles, conforme as suas respectivas ligações lógicas. No modelo, também está definido que a tentativa de evolução de LVT nos componentes, acontecem a cada 200 milissegundos. Os valores de evolução de tempo local para os componentes: 5, 6, 7, 8 e 9, são respectivamente, 100, 200, 150, 300 e 500 unidades de tempo.

A partir desse modelo, foram definidos três grafos que representam topologias de trocas de mensagens. Nesta ideia, os vértices identificam os componentes e as arestas indicam a direção do fluxo das mensagens de simulação. Esse estudo de caso utiliza as três configurações descritas e aplicadas no trabalho da Acácia Terra.

Os componentes estão configurados como síncronos para todas as topologias, visto que o estudo sobre cálculo de GVT foi direcionado para este tipo de simulação. Os grafos a seguir representam os cenários utilizados como forma de comparação.

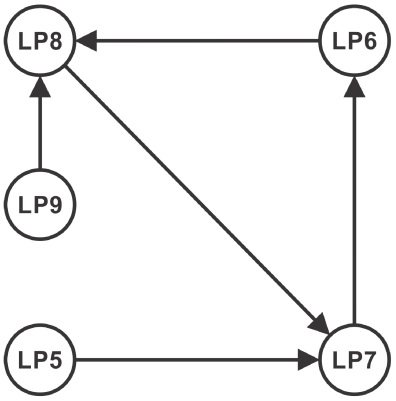

Figura 6.1 – Modelo A

A figura 6.1 mostra o modelo A, onde representa uma situação que há espera circular, utilizado no trabalho de Ricardo Zanuzzo para identificação e prevenção de impasse. Assim,

como as novas versões do DCB que foram implementadas, incorporam estas funcionalidades, o mesmo modelo de teste foi mantido neste estudo de caso.

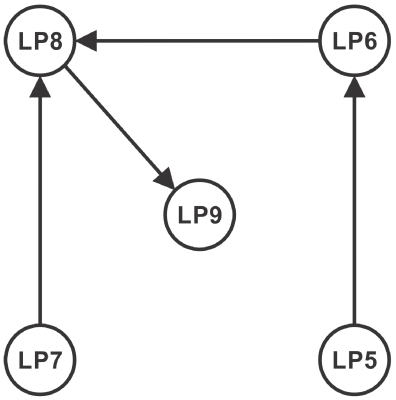

Figura 6.2 – Modelo B

O modelo B representado na figura 6.2 mostra uma situação em que não há espera circular. Este exemplo mantém a mesma distribuição do modelo A, porém as conexões lógicas entre os componentes estão dispostas de maneira diferente.

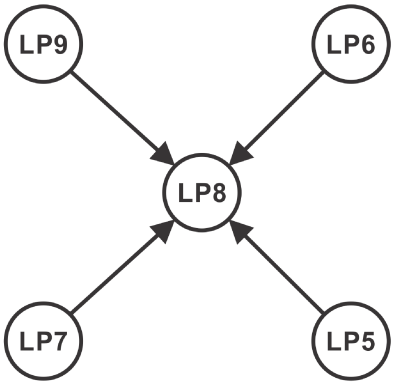

Figura 6.3 – Modelo C

Por outro lado, o modelo C (figura 6.3), exemplifica uma situação onde há um componente centralizado (LP8) que recebe mensagens de evento de todos os demais.

Para análise dos resultados, foram realizados experimentos no DCB considerando 1 milhão de unidades de tempo. Para cada modelo, foram computadas cinco execuções. Além disso, foram comparadas três implementações do DCB, são elas:

1. Última versão do DCB, modificada pela Acácia Terra, que utiliza uma abordagem centralizada para o cálculo de GVT (fedgvt);

2. Uma nova versão implementada no DCB, que utiliza o algoritmo descentralizado CMGVT adaptado;

3. Uma versão alternativa, que simplifica o cálculo de GVT (considera apenas os LVTs) em relação ao algoritmo CMGVT.

6.1 Análise de Resultados

A partir das três versões do DCB citadas acima, juntamente com as três topologias, foram realizadas cinco execuções das simulações sobre cada uma delas, totalizando 45 experimentos de simulação. Deste modo, esta seção apresenta os dados estatísticos obtidos durante os testes. Os resultados estão descritos (através da média simples) nas tabelas 6.1, 6.2 e 6.3 para cada uma das três topologias, seguido de alguns gráficos contendo dados extraídos das tabelas. As informações completas das 45 execuções encontram-se no anexo A deste trabalho. A organização das tabelas estão dispostas da seguinte forma:

- A primeira coluna indica qual algoritmo foi utilizado;
- A segunda coluna identifica o processo lógico;
- A terceira coluna mostra o número de tentativas de envio das mensagens reais pelo componente indicado;
- A quarta coluna representa o número de mensagens reais recebidas pelo respectivo componente;
- A quinta coluna apresenta o número de mensagens perdidas, isto é, aquelas que não foram enviadas, pois causariam violações de tempo (LCC);
- A sexta coluna mostra a quantidade de mensagens de controle (sincronização de GVT) transitadas durante a simulação;
- A última coluna indica o tempo real necessário para execução da simulação.

A tabela 6.1 refere-se aos dados obtidos sobre o modelo A pelos três algoritmos analisados (CMGVT, Alternativo e Centralizado). Nesta tabela, é possível observar algumas diferenças entre as versões testadas, tais como o número de mensagens reais enviadas por cada componente, onde as duas versões descentralizadas tiveram um resultado parecido. Contudo, observando o gráfico A da figura 6.4, houve um aumento de aproximadamente 43% da quantidade de mensagens enviadas em relação à versão centralizada. Esse crescimento mostra que

Execuções	*PL*	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
CMGVT	**5**	2863	0	0.8	17165.2	47M45.058S
	6	2862	2861.6	0	36483.2	47M44.904S
	7	2861.8	5723.8	0.2	58226.6	47M44.722S
	8	2861.6	5724.2	0	55091.4	47M44.860S
	9	2862.2	0	0	17221	47M44.941S
Alternativo	**5**	2869.8	0	1.2	17235.4	47M51.932S
	6	2869.2	2868.4	0.4	38516	47M51.835S
	7	2868.6	5737.6	0.2	55871.4	47M51.626S
	8	2868.8	5737.4	0	53296	47M51.660S
	9	2869.2	0	0.4	17211.2	47M51.921S
Centralizado	**5**	2002.8	0	0.8	20010.6	33M25.110S
	6	2002.2	2001.6	0.4	23947.2	33M24.935S
	7	2002.8	4003.8	1.2	23967.2	33M24.876S
	8	2002	4003	0.2	25393.6	33M24.899S
	9	2002.8	0	1.6	22544.6	33M24.879S

Tabela 6.1 – Modela A (Experimento 1)

aconteceu uma redução da ociosidade, onde com a mesma quantidade de unidades de tempo de simulação, houve um melhor aproveitamento, tornando a simulação mais realista. A justificativa referente a este aumento, deve-se ao fato de que a descentralização do cálculo de GVT causa um atraso natural na atualização deste valor, pois o avanço desta barreira de tempo depende do acesso às informações de controle atualizadas, isto é, quanto mais rápido cada componente conseguir acessá-las, mais rápida será a evolução do GVT, diminuindo assim, a quantidade de mensagens reais enviadas por cada componente em relação ao tempo simulado. Essa observação pode ser constatada na versão centralizada, onde o controlador consegue obter informações de controle atualizadas com mais frequência, o que acarreta um avanço acelerado do GVT, levando rapidamente ao término do tempo simulado e por consequência uma quantidade reduzida de mensagens reais de simulação transitadas.

Ainda sobre o modelo A, na tabela 6.1, a sexta coluna destaca o número de mensagens de sincronização transitadas para fins de atualização de GVT. Neste caso, as duas estratégias implementadas se mostraram menos econômicas em termos de quantidade, principalmente nos componente 7 e 8, que são aqueles que possuem mais vizinhos. O gráfico da figura 6.5 demonstra a diferença de mensagens de controle entre as versões; as duas versões descentralizadas tiveram um número médio parecido de mensagens de controle. Por outro lado, houve um acréscimo significativo em relação ao algoritmo centralizado, cerca de 55%. Esse grande aumento está diretamente ligado com a forma como as estratégias trabalham; as abordagens descentralizadas implementadas, enviam mensagens de controle a partir de cada componente para seus

vizinhos quando ocorre a tentativa de evolução de LVT. Portanto, com essa medida, somado ao uso de propagação, exige uma troca maior de mensagens de sincronização durante a simulação. Por outro lado, o algoritmo centralizado envia apenas uma mensagem de sincronização (diretamente ao controlador) ao evoluir o LVT, diminuindo drasticamente a quantidade transitada.

Execuções	***PL***	***M. enviadas***	***M. recebidas***	***M. perdidas***	***M. GVT***	***Tempo***
CMGVT	**5**	2943.4	0	0.6	17640.6	49M5.992S
	6	2942.6	2942.8	0.2	36082.4	49M5.788S
	7	2943	0	0.2	17647.6	49M5.892S
	8	2942.4	5885.2	0.2	58960.8	49M5.646S
	9	0	2942.2	0	17193.4	49M5.822S
Alternativo	**5**	2894.4	0	1.2	18127.8	48M23.883S
	6	2893.8	2893.6	0.8	37834,8	48M23.841S
	7	2893.6	0	0.6	17526.2	48M23.769S
	8	2894	5786	0	49651.2	48M23.722S
	9	0	2894	0	16074.8	48M23.793S
Centralizado	**5**	1823	0	1.2	22015.8	30M24.363S
	6	1822	1821.8	0.8	23681.6	30M24.169S
	7	1822	0	0.6	21700	30M24.204S
	8	1822	3642.6	0.4	24526	30M24.149S
	9	0	1821.6	0	21317.6	30M24.017S

Tabela 6.2 – Modelo B (Experimento 1)

Os dados apresentados na tabela 6.2 mostram os resultados obtidos pelas execuções no modelo B. O número de mensagens enviadas neste modelo permaneceu proporcional aos valores obtidos no modelo A. Contudo, neste caso, houve um acréscimo significativo em relação aos resultados da versão centralizada; os valores obtidos para os algoritmos CMGVT e o Alternativo tiveram um aumenta aproximado de 60% da quantidade de mensagens, conforme o gráfico B da figura 6.4. Esta constatação reafirma a diminuição da ociosidade dos componentes comparados ao tempo simulado. A simplificação do modelo B para este estudo de caso, também justifica as diferenças do modelo A e B em relação aos testes do algoritmo centralizado, já que a sua principal característica é a geração e envio de mensagem a cada 1 segundo. Deste modo, com o atraso natural da evolução de GVT nas novas versões, somado ao envio programado das mensagens, há o aumento do número total enviado.

Outro dado observado nos resultados para o modelo B, referem-se ao número de mensagens de controle, que assim como no modelo A, mostrou-se maior nos algoritmos descentralizados. De acordo com o gráfico na figura 6.5, o aumento das mensagens de controle foi de aproximadamente 22% (CMGVT) e 28% (Alternativo) em relação aos testes com o algoritmo centralizado, demostrando a característica da abordagem descentralizada com uso do artifício

de propagação.

Execuções	*PL*	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
CMGVT	**5**	2148.8	0	0.4	12886	35M50.936S
	6	2148.6	0	0	12890	35M50.788S
	7	2148.6	0	0	12898.2	35M50.729S
	8	0	8593.4	0	41987.6	35M50.789S
	9	2148.6	0	0	12905.4	35M50.781S
Alternativo	**5**	2145.8	0	1	12854.8	35M46.417S
	6	2145.2	0	0.4	12860.2	35M46.251S
	7	2145	0	0.2	12857	35M46.287S
	8	0	8579.2	0	41410.8	35M46.268S
	9	2145	0	0.2	12872.4	35M46.072S
Centralizado	**5**	1878	0	0.6	21574.8	31M19.855S
	6	1877.6	0	0.6	22420.6	31M19.523S
	7	1877.6	0	0.6	22038	31M19.583S
	8	0	7508.2	0	25040.4	31M19.728S
	9	1877.8	0	1	21857.4	31M19.687S

Tabela 6.3 – Modelo C (Experimento 1)

Os resultados obtidos dos testes do modelo C estão descritos na tabela 6.3. Esse modelo dispõe de um componente central que apenas recebe mensagens de simulação dos demais. Conforme o gráfico C da figura 6.4, houve um aumento de aproximadamente 14% de mensagens enviadas pelas versões descentralizadas em relação à centralizada, mantendo a proporcionalidade observada nos modelos A e B, apesar de demonstrar um crescimento menor. A aproximação dos valores constatados nestes testes, confirmam que uma menor distância entre os componentes (saltos), representam uma evolução mais rápida do tempo simulado nas estratégias descentralizadas apresentadas, provocando essa diminuição do número total de mensagens enviadas.

Em relação as mensagens de controle para os testes do modelo C, houve uma constatação um pouco diferente dos modelo A e B. Conforme o gráfico da figura 6.5, verificou-se que as duas versões descentralizadas tiveram um número médio de mensagens de controle menor em relação aos valores obtidos na versão centralizada. Essa diferença é de aproximadamente 20%. Contudo, os resultados do CMGVT em comparação ao Alternativo são praticamente os mesmos, já que ambos utilizam o mesmo artifício de propagação de mensagens. A redução do número de mensagens de controle para as versões descentralizadas neste modelo é justificada pela proximidade dos componentes, isto é, o número de saltos para um mensagem ser propagada é exatamente 1, assim, os componentes obtêm as informações de controle de forma mais rápida, diminuindo a espera imposta pela barreira do GVT, fazendo com que haja um avanço maior do

tempo simulado.

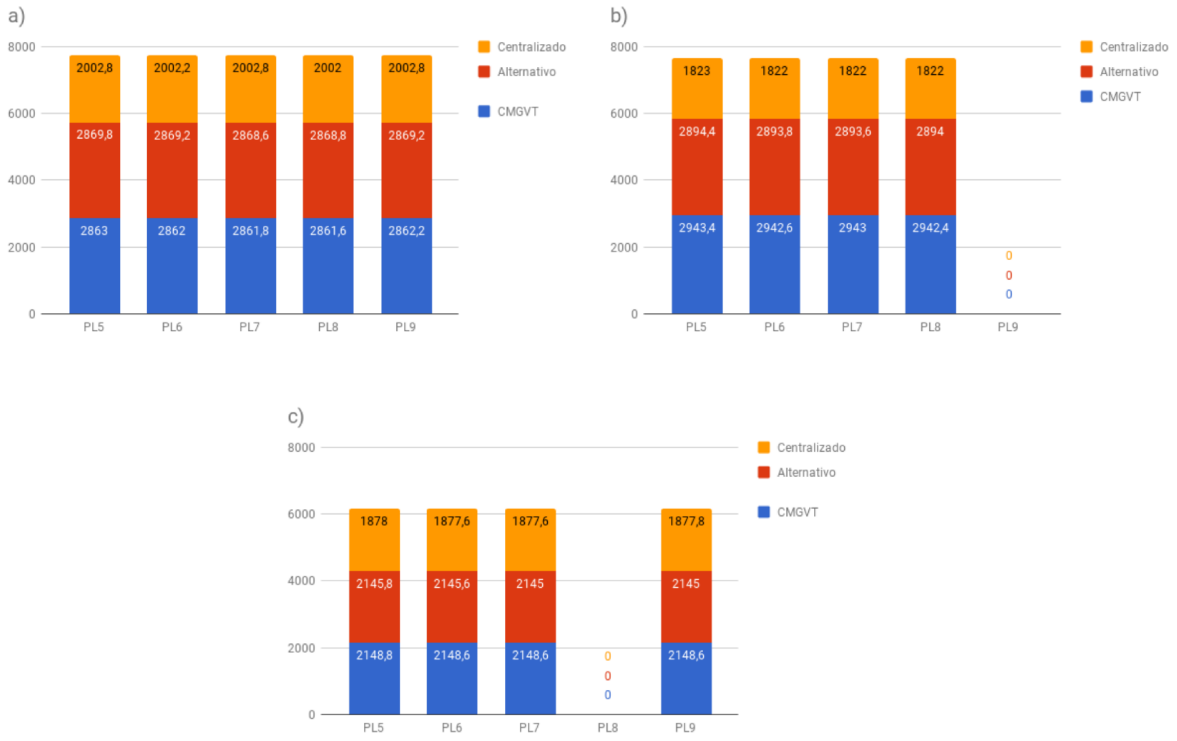

Figura 6.4 – Mensagens enviadas em cada modelo

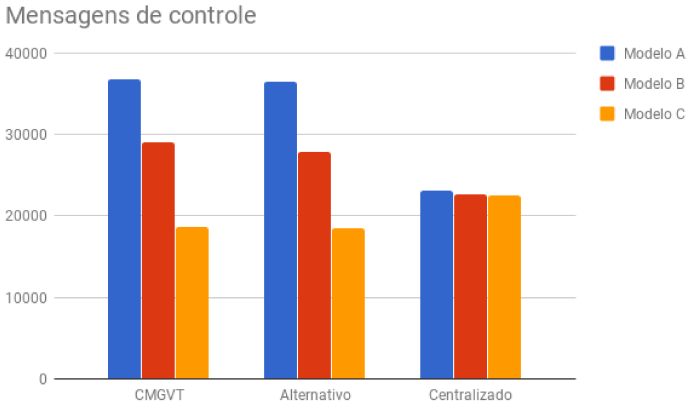

Figura 6.5 – Mensagens de controle

Um dado importante destacado nas três tabelas acima, refere-se a diferença entre o número de mensagens enviadas e recebidas (conforme a topologia de cada modelo). Uma vez que essa diferença não é necessariamente nula, essa quantidade representa o número de mensagens perdidas (quinta coluna nas tabelas). A contagem dessas mensagens, para as versões descentralizadas, demonstram que houve uma redução no número de perdas em comparação com às versões centralizadas. Conforme o gráfico da figura 6.6, o algoritmo utilizando a abordagem

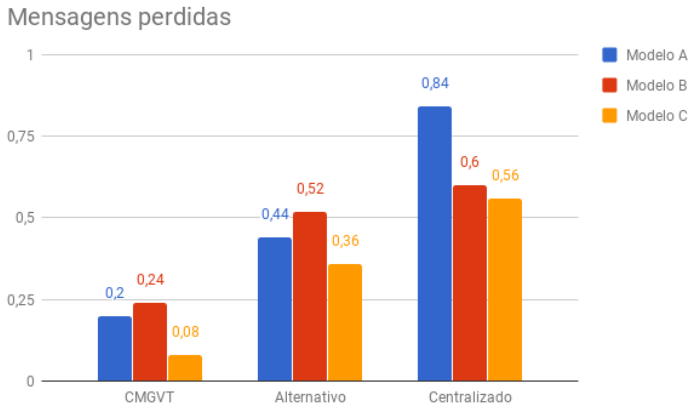

Figura 6.6 – Mensagens perdidas

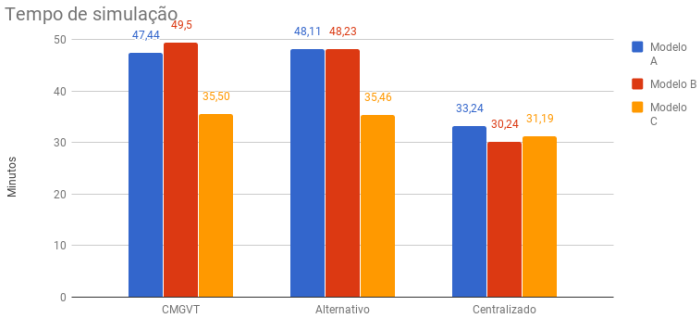

Figura 6.7 – Tempo de simulação

centralizada, apresentou uma média superior de mensagens perdidas em relação as outras versões. As perdas de mensagens estão vinculadas ao avanço frequente do GVT, pois quando uma mensagem é gerada por um componente (do modelo), o *timestamp* dela leva em conta o GVT vigente, então, quando ela é direcionada ao DCB, o GVT já poderia ter sofrido uma atualização e por consequência a mesma não pode ser enviada, pois causaria uma violação de tempo. Portanto, quanto menor a frequência de atualização de GVT (acesso às informações de controle atualizadas), menor a chance de perder a mensagem por esse motivo. Também é possível observar uma relação entre o aumento no número de mensagens de sincronização com a redução da quantidade de mensagens reais perdidas. Essa constatação pode ser vista nos modelos A, B e C, onde as versões descentralizadas demonstraram um aumento no número de mensagens de controle e uma diminuição no número de mensagens reais perdidas, enquanto isso, na versão centralizada constatou-se o inverso (comparando os gráficos 6.5 e 6.6). Uma outra observação a ser feita, é a comparação entre as duas versões descentralizadas, onde a redução das mensagens perdidas no CMGVT foi dobrada em relação ao algoritmo Alternativo. Uma possível explicação para este dado, talvez seja a forma como as duas estratégias se comportam, isto é,

enquanto a versão Alternativa utiliza apenas os LVTs como parâmetro de cálculo, o CMGVT usa também o *timestamp* das mensagens correntes (não confirmadas). Essa diferença sugere que há uma melhor precisão no cálculo considerando as mensagens não confirmadas (em trânsito) na estimativa do GVT, conforme as definições de tempo virtual global encontradas nas bibliografias.

A última coluna das tabelas 6.1, 6.2 e 6.3, descrevem os resultados obtidos em relação ao tempo real de simulação para cada caso. Nestes dados, as versões descentralizadas demostraram um acréscimo significativo do tempo total. No modelo A, o tempo médio aumentou cerca de 42% em relação aos testes feitos com a versão centralizada. E também, no modelo B, este acréscimo foi de aproximadamente 60%. Ainda, no modelo C, este aumento representou cerca de 12%. A explicação para estes valores elevados, está diretamente relacionada aos pontos destacados sobre o envio de mensagens, isto é, pela utilização da propagação que impõe um atraso natural na evolução do tempo simulado em cada componente até o recebimento de informações de controle atualizadas. Além disso, também possui relação com a quantidade de vizinhos e a distância entre os componentes (saltos), que fazem com que o alcance de uma atualização de controle seja limitada conforme a propagação de uma mensagem, ou seja, os componentes que possuem uma distância maior entre si em número de saltos, terão um atraso no recebimento atualizado de informações de controle, aumentando assim, o tempo real de simulação.

Durante o processo de adaptação das estratégias descentralizadas escolhidas, observou-se a existência de uma relação entre o tempo real de simulação e a frequência das mensagens de sincronização. Se limitada a quantidade de mensagens de sincronização enviadas, a atualização local das informações de controle tende a demorar mais, o que impacta diretamente no tempo total de simulação. Por outro lado, se a frequência do envio de mensagens de controle é alta, as informações atualizadas chegam de forma mais rápida nos componentes (via propagação), diminuindo assim, o tempo real de simulação.

Na análise feita dos três modelos em relação aos experimentos, alguns resultados obtidos foram contrários aos objetivos deste trabalho, principalmente na busca da redução das mensagens de controle, onde para a maioria dos casos testados, se mostraram superiores aos resultados apresentados pela última versão que usa uma abordagem centralizada para o cálculo de GVT. Outro ônus apresentado foi em relação ao tempo real de simulação, onde novamente, observou-se um aumento significativo. Por outro lado, houve uma diminuição na ociosidade correspondente ao tempo simulado, permitindo uma geração maior da quantidade de mensa-

gens reais de simulação transitadas entre os componentes, além de permitir uma menor perda nestas mensagens decorridas por avanços exagerados do tempo virtual global.

A utilização da propagação como recurso de atualização de GVT impõe algumas dificuldades em relação à consistência da barreira de GVT. Por exemplo, em um mesmo instante de tempo, cada componente pode possuir um conhecimento diferente (e também desatualizado) dos demais em relação aos dados de controle. Deste modo, o cálculo de GVT baseado em informações locais podem fazer com o que a noção do GVT seja diferente em cada componente. Uma consequência direta disso é a possibilidade da ocorrência de violações de tempo, quando há o envio de mensagens de simulação entre componentes com percepções distintas sobre o GVT. Contudo, a chance de ocorrer tais violações depende das características de cada modelo em relação a própria evolução no tempo. Assim, o perigo de haver perda de mensagens por violações de tempo ainda existe com a utilização da propagação, da mesma maneira como existe na versão centralizada ao enviar uma atualização de GVT aos componentes, pois as mensagens de sincronização são enviadas em ordem de acordo com uma lista, permitindo que a percepção do GVT em alguns componentes estejam momentaneamente a frente dos demais. Porém, comparada com as versões implementadas neste trabalho, a diferença da noção do GVT em cada componente não é necessariamente a mesma da atual versão centralizada.

Uma possível situação de falha (em que ocorre violação de tempo) foi representada anteriormente na figura 5.1. Neste caso, a mensagem é considerada perdida. O impacto dessa perda depende da representatividade da mensagem em relação ao modelo. Nesta ideia, é importante considerar uma análise estatística das perdas em relação ao resultado final da simulação, que visa identificar a melhor abordagem a ser adotada (centralizada ou não). Neste estudo de caso, situações em que ocorrem perdas de mensagens por violações de tempo não foram observadas, pois, apesar de haver uma noção distinta sobre o GVT entre os componentes, essa diferença foi pequena, ao ponto de não provocar tais violações.

6.2 Comparação do algoritmo CMGVT

Neste estudo de caso também foram feitas comparações entre o algoritmo CMGVT original (conforme apresentado por seus autores) e a versão adaptada para utilização em componente síncronos. As diferenças analisadas referem-se nas modificações realizadas no procedimento que recebe as mensagens (algoritmo 2). A primeira modificação remete aos parâmetros utilizados para inclusão e exclusão de novos vetores na tabela TFV. Assim, para o CMGVT

original, considera-se apenas a contagem das mensagens enviadas, e não mais os *timestamps* das mensagens. A segunda modificação está nas condições de atualização dos LVTs em cada componente, isto é, com o algoritmo original compara-se diretamente o número de mensagens enviadas (em relação as estruturas: MM destino e MM origem). Então, atualiza-se os valores de cada LVT apenas se há um conhecimento mais recente (com maior valor) comparado ao número de mensagens enviadas. O CMGVT adaptado ficou inalterado neste segundo experimento.

O algoritmo CMGVT original não utiliza explicitamente mensagens de sincronização (S). Para tornar possível essa comparação, observou-se a necessidade mantê-las com a utilização do DCB. Contudo o seu emprego é reduzido, isto é, as mensagens de sincronização são enviadas apenas para componentes origens (sentido inverso nas arestas), diferente da versão adaptada neste trabalho, que envia as mensagens para qualquer vizinho, independente dos sentidos das arestas nos grafos.

Para este segundo teste, o tempo total simulado foi reduzido para 100 mil unidades de tempo. Deste modo, foram feitas cinco execuções para cada algoritmo (CMGVT original e o adaptado) sobre cada um dos três modelos (A, B e C) apresentados anteriormente, totalizando 30 execuções. Os dados completos sobre as execuções estão descritas no Anexo B deste trabalho.

Primeiramente, é importante destacar que os experimentos realizados no DCB com o uso do CMGVT original, quando aplicados em componentes síncronos, não foram totalmente satisfatórios. A principal questão refere-se na atualização de LVTs em cada componente, onde há possibilidade de substituição do LVT conhecido para um valor menor quando recebe informações de controle de algum componente vizinho. Essa situação é possível, pois o CMGVT original usa a contagem de mensagens como parâmetro de atualização dos LVTs conhecidos, assim, pode haver a substituição do valor (para um conhecimento ultrapassado). A consequência direta disso, é o aumento no tempo real de simulação, que somado ao uso de propagação, influenciam negativamente no propósito geral da simulação. Por este motivo, o tempo simulado foi limitado neste segundo experimento, pois é necessário que haja um equilíbrio entre o tempo real de simulação e tempo simulado, ou seja, quanto menores forem ambos os tempos, melhor para a simulação.

Observando as tabelas 6.4, 6.5 e 6.6 em relação a quantidade de mensagens enviadas, é possível verificar que o CMGVT original consegue gerar um número maior em relação ao CMGVT adaptado. Levando em conta que o tempo real no CMGVT original é muito superior

ao adaptado, é natural que sejam geradas mais mensagens durante o tempo simulado.

A contagem de mensagens de sincronização (sexta coluna nas tabelas) para o algoritmo CMGVT original também foram superiores ao adaptado, devido ao avanço lento do GVT, e por consequência o aumento do tempo de simulação. Outro fator que contribuiu para a grande diferença nesta quantidade de mensagens, refere-se ao fato de que toda tentativa de evolução de LVT força um envio de mensagem de sincronização aos vizinhos, então, quanto mais o GVT permanece sem alteração, mais mensagens serão enviadas, influenciando na contagem final. Esse problema se agrava considerando os aspectos relatados anteriormente sobre o algoritmo original (retrocesso do conhecimento sobre os LVTs).

Por outro lado, com o CMGVT original, não foi constatada perda de mensagens, todas aquelas encaminhadas ao DCB, foram enviadas com sucesso. A retenção do GVT, como apontado anteriormente, causa esse efeito.

Em relação ao tempo real de simulação, o algoritmo original apresentou um crescimento elevado, pelos mesmos fatores apontados anteriormente. Este aumento no tempo real também motivou a redução da quantidade de tempo neste experimento (de 1 milhão para 100 mil), para garantir a devida realização dos testes, visto que não foi possível observar a garantia de término da simulação utilizando o CMGVT original para simulações longas. Deste modo, para os modelos A e C as execuções levaram em média 33 minutos para finalização, enquanto para o modelo B levou cerca de 43 minutos, valores bem superiores aos resultados obtidos na versão adaptada.

O objetivo deste segundo experimento foi permitir que houvesse uma comparação entre o CMGVT original e a versão adaptada, a fim de justificar as mudanças realizadas, principalmente no procedimento executado ao receber uma mensagem (conforme o algoritmo 2) que precisou ser alterado para ser utilizado no DCB juntamente com o controle de GVT nos componentes síncronos. Neste experimento também não foram detectadas violações de tempo, apesar disso, a estratégia original não é totalmente adequada para componentes síncronos, visto que o aumento no tempo real de simulação é expressivo comparado com a adaptação feita.

Execuções	*PL*	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
CMGVT Original	**5**	998.8	0	0	22746.4	33M17.860S
	6	998.6	998.6	0	21907.6	33M17.883S
	7	998.6	1997.4	0	22053.8	33M17.936S
	8	998.6	1997.4	0	32903.8	33M17.850S
	9	998.6	0	0	10986.4	33M17.882S
CMGVT Adaptado	**5**	282.2	0	0.4	1680.4	4M39.940S
	6	278.4	278.2	0	3783.2	4M39.949S
	7	278.4	560.2	0.2	5780.4	4M39.964S
	8	278.4	557	0	5463.6	4M39.995S
	9	278.6	0	0	1678	4M40.024S

Tabela 6.4 – Modelo A (Experimento 2)

Execuções	*PL*	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
CMGVT Original	**5**	1021.2	0	0	15454.2	43M10.908S
	6	1021.6	1021.2	0	14133.2	43M8.843S
	7	1021.2	0	0	14153.4	43M8.912S
	8	1021.6	2042.6	0	43453.2	43M8.948S
	9	0	1021.6	0	28058.4	43M9.146S
CMGVT Adaptado	**5**	287	0	0.8	1734.4	4M48.834S
	6	286.8	286.2	0.2	3535.2	4M48.578S
	7	287	0	0	1729.4	4M48.674S
	8	287	573.6	0	5908.2	4M48.347S
	9	0	287	0	1731.8	4M48.445S

Tabela 6.5 – Modelo B (Experimento 2)

Execuções	*PL*	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
CMGVT Original	**5**	997.8	0	0	10977.8	33M19.132S
	6	997.4	0	0	10978.6	33M19.053S
	7	997.6	0	0	10979.4	33M18.901S
	8	0	3990.6	0	39965.2	33M18.838S
	9	997.8	0	0	10978	33M18.899S
CMGVT Adaptado	**5**	210.6	0	0.4	1275.6	3M32.185S
	6	209.6	0	0.2	1270.2	3M32.056S
	7	210.2	0	0	1271.6	3M32.064S
	8	0	839.8	0	4292.8	3M31.812S
	9	210	0	0	1272.8	3M31.943S

Tabela 6.6 – Modelo C (Experimento 2)

7 CONCLUSÕES

Neste trabalho foi apresentada uma estratégia que visa o cálculo de tempo virtual global (GVT) em componentes síncronos de forma descentralizada, isto é, sem a dependência de um único controlador. O algoritmo base escolhido, chama-se CMGVT (técnica escolhida através dos trabalhos correlatos). O ambiente experimental que foi utilizado para a implementação e a adaptação da estratégia, foi o DCB. Assim, ao final, também foram realizados testes e análises para validação dos resultados.

A distribuição do processo de cálculo de GVT possibilita uma maior independência para os componentes, ajudando a reduzir a centralização da carga de controle, observadas em soluções que utilizam um único gerenciador. Um outro aspecto propiciado é a redução da ociosidade imposta sobre o tempo simulado, acarretando um aumento no número de mensagens de simulação com uma mesma fatia de tempo.

A escolha do algoritmo CMGVT levou em consideração a existência de dois procedimentos gerais : *sender* e *receiver*, com o propósito de preservar a arquitetura e as características incorporadas pelo DCB. Essa estratégia precisou ser adaptada para funcionar no gerenciamento de GVT para os componentes síncronos, já que a sua forma original é destinada ao uso em simulação assíncrona. Durante a adaptação da estratégia selecionada, observou-se a oportunidade de implementar também uma simplificação, assim, foram geradas duas novas versões utilizando uma abordagem descentralizada.

A partir dos resultados obtidos, observou-se que ambos os algoritmos implementados contribuíram para a redução da ociosidade do tempo simulado e também na diminuição da quantidade de mensagens perdidas (aquelas que não são enviadas, pois causariam uma violação de tempo). Em contrapartida, verificou-se um aumento no número de mensagens de controle e no período real de simulação. Assim, apesar do acréscimo do custo gerado, as implementações apresentadas permitiram um melhor aproveitamento do tempo simulado, tornando a simulação mais realista neste quesito.

7.1 Trabalhos Futuros

O estudo e a implementação de um novo modelo de simulação pode ser feito, pois atualmente o estudo de caso presente para o DCB é limitado e não representa um processo ou sistema real. Neste ponto, um modelo mais complexo e representativo poderia avaliar melhor

os mecanismos implementados, não apenas em relação ao cálculo de GVT, mas também nas demais funcionalidades presentes atualmente no escopo do DCB.

A utilização da propagação como recurso no gerenciamento descentralizado, permite que cada componente possua uma noção diferente do GVT em um mesmo instante de tempo, então, o uso em simulação síncrona possibilita que ocorram violações de tempo quando há o envio de mensagens entre componentes que possuem um conhecimento distinto em relação ao GVT. Nesta ideia, poderia ser feito um estudo referente ao impacto causado, considerando a quantidade de mensagens que causam violações de tempo, pode-se fazer uma análise estatística e mensurar qual a influência no resultado da simulação com a perda dessas mensagens.

O uso do algoritmo CMGVT traz algumas dificuldades, tais como: o aumento no tamanho das mensagens, visto que as estruturas MM e TFV são encaminhadas com ela, afetando o desempenho em termos de comunicação. Teoricamente, a TFV não possui um tamanho limitado, já que não há um delimitador no número de mensagens. Deste modo, formas de restringir o tamanho da TFV são necessárias para garantir um bom desempenho quando esta estratégia é utilizada em simulação distribuída.

O propósito do algoritmo CMGVT é no auxílio da remoção de informações obsoletas na simulação otimista. Neste trabalho, ela foi adaptada para ser utilizada no gerenciamento de GVT para componentes síncronos. Tais modificações não foram comparadas em termos de desempenho nos componentes assíncronos. Nesta ideia, uma implementação poderia ser readaptada e testada sob uma perspectiva assíncrona, a partir do mesmo ambiente experimental utilizado neste trabalho (DCB). Assim, é possível fazer uma análise comparativa em relação ao seu propósito original.

REFERÊNCIAS

BANKS, J. Introduction to simulation. In: SIMULATION CONFERENCE PROCEEDINGS, 1999 WINTER. **Anais...** [S.l.: s.n.], 1999. v.1, p.7–13 vol.1.

BAUER, H.; SPORRER, C. Distributed logic simulation and an approach to asynchronous GVT-calculation. In: WORKSHOP PARALLEL AND DISTRIBUTED SIMULATION, 6. **Proceedings...** [S.l.: s.n.], 1992. p.205–208.

BRAGARD, Q.; VENTRESQUE, A.; MURPHY, L. Synchronisation for Dynamic Load Balancing of Decentralised Conservative Distributed Simulation. In: ND ACM SIGSIM CONFERENCE ON PRINCIPLES OF ADVANCED DISCRETE SIMULATION, 2., New York, NY, USA. **Proceedings...** ACM, 2014. p.117–126. (SIGSIM PADS '14).

CHEN, G.; SZYMANSKI, B. Time Quantum GVT: a scalable computation of the global virtual time in parallel discrete event simulations. , [S.l.], v.8, 01 2007.

DEELMAN, E.; SZYMANSKI, B. K. **Continuously Monitored Global Virtual Time**. [S.l.]: in Intern. Conf. Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, 1996.

D'SOUZA, L. M.; FAN, X.; WILSEY, P. A. pGVT: an algorithm for accurate gvt estimation. In: EIGHTH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION, New York, NY, USA. **Proceedings...** ACM, 1994. p.102–109. (PADS '94).

ELLEITHY, A.; RIZVI, S. S.; ELLEITHY, K. M. Investigating the Effects of Trees and Butterfly Barriers on the Performance of Optimistic GVT Algorithm. In: ADVANCED TECHNIQUES IN COMPUTING SCIENCES AND SOFTWARE ENGINEERING, Dordrecht. **Anais...** Springer Netherlands, 2010. p.449–453.

FERSCHA, A.; TRIPATHI, S. K. **Parallel and Distributed Simulation of Discrete Event Systems**. College Park, MD, USA: [s.n.], 1994.

FUJIMOTO, R. M. Lookahead in parallel discrete event simulation. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 1988. **Proceedings...** [S.l.: s.n.], 1988. p.34—-41.

FUJIMOTO, R. M. Time Management in The High Level Architecture. **SIMULATION**, [S.l.], v.71, n.6, p.388–400, 1998.

FUJIMOTO, R. M. Parallel and Distributed Simulation. In: CONFERENCE ON WINTER SIMULATION: SIMULATION—A BRIDGE TO THE FUTURE - VOLUME 1, 31., New York, NY, USA. **Proceedings...** ACM, 1999. p.122–131. (WSC '99).

FUJIMOTO, R. M.; HYBINETTE, M. Computing Global Virtual Time in Shared-memory Multiprocessors. **ACM Trans. Model. Comput. Simul.**, New York, NY, USA, v.7, n.4, p.425–446, Oct. 1997.

JAFER, S.; LIU, Q.; WAINER, G. Synchronization Methods in Parallel and Distributed Discrete Event Simulation. , [S.l.], v.30, p.3–10, 01 2013.

JEFFERSON, D. R. Virtual Time. **ACM Trans. Program. Lang. Syst.**, New York, NY, USA, v.7, n.3, p.404–425, July 1985.

LIN, Y. B.; LAZOWSKA, E. D. Determining the Global Virtual Time in a Distributed Simulation. In: INTERNACIONAL CONFERENCE ON PARALLEL PROCESSING, 1990. **Proceedings...** [S.l.: s.n.], 1990. p.201–209.

MATTERN, F. Efficient algorithms for distributed snapshots and global virtual time approximation. **Journal of Parallel and Distributed Computing**, [S.l.], v.18, n.4, p.423–434, 1993.

MELLO, B. A. de. **Co-simulação distribuída de sistemas heterogêneos**. 2005. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

MIKIDA, E.; KALE, L. Adaptive Methods for Irregular Parallel Discrete Event Simulation Workloads. In: ACM SIGSIM CONFERENCE ON PRINCIPLES OF ADVANCED DISCRETE SIMULATION, 2018., New York, NY, USA. **Proceedings...** ACM, 2018. p.189–200. (SIGSIM-PADS '18).

OVEREINDER, B. **Distributed Event-driven Simulation- Scheduling Strategies and Resource Management**. 2000. Tese (Doutorado em Ciência da Computação) — .

PERUMALLA, K. S.; PARK, A. J.; TIPPARAJU, V. Discrete Event Execution with One-Sided and Two-Sided GVT Algorithms on 216,000 Processor Cores. **ACM Trans. Model. Comput. Simul.**, New York, NY, USA, v.24, n.3, p.16:1–16:25, June 2014.

RAYNAL, M.; SINGHAL, M. Logical time: capturing causality in distributed systems. **Computer**, [S.l.], v.29, n.2, p.49–56, Feb 1996.

REYNOLDS JR., P. F. Heterogenous Distributed Simulation. In: CONFERENCE ON WINTER SIMULATION, 20., New York, NY, USA. **Proceedings...** ACM, 1988. p.206–209. (WSC '88).

SAMADI, B. **Distributed Simulation, Algorithms and Performance Analysis (Load Balancing, Distributed Processing)**. 1985. Tese (Doutorado em Ciência da Computação) — . AAI8513157.

SOLIMAN, H. M. Parallel and Distributed Simulation: methodologies and techniques. **Journal of King Saud University - Computer and Information Sciences**, [S.l.], v.10, n.Supplement C, p.27 – 51, 1998.

SRINIVASAN, S.; REYNOLDS JR., P. F. Non-interfering GVT Computation via Asynchronous Global Reductions. In: CONFERENCE ON WINTER SIMULATION, 25., New York, NY, USA. **Proceedings...** ACM, 1993. p.740–749. (WSC '93).

TERRA, A. dos Campos da. **Cálculo de Lookahead dinâmico e individual no DCB**. 2017. 58 f. Monografia (Bacharel em Ciência da Computação) - Universidade Federal da Fronteira Sul, Chapecó, 2017.

TOMLINSON, A. I.; GARG, V. K. An Algorithm for Minimally Latent Global Virtual Time. In: SEVENTH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION, New York, NY, USA. **Proceedings...** ACM, 1993. p.35–42. (PADS '93).

YOUNG, C. H.; RADHAKRISHNAN, R.; WILSEY, P. A. Optimism: not just for event execution anymore. In: THIRTEENTH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 1999. p.136–143. (PADS '99).

ZANUZZO, R. **Identificação e prevenção de situações de impasse em modelos heterogênios de simulação distribuída**. 2017. 42 f. Monografia (Bacharel em Ciência da Computação) - Universidade Federal da Fronteira Sul, Chapecó, 2017.

ANEXOS

ANEXO A – Resultados das Execuções (Experimento 1)

A.1 Resultados para o algoritmo CMGVT adaptado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2858	0	1	17134	47M40.395S
6	2857	2857	0	39242	47M40.023S
7	2857	5714	0	58506	47M40.159S
8	2857	5714	0	54732	47M40.064S
9	2857	0	0	17410	47M40.238S
5	2893	0	1	17343	48M15.251S
6	2892	2892	0	28788	48M15.241S
7	2892	5784	0	57807	48M15.103S
8	2892	5784	0	54090	48M15.119S
9	2892	0	0	17353	48M15.186S
5	2860	0	1	17151	47M42.393S
6	2859	2858	0	39066	47M42.276S
7	2858	5717	0	58584	47M42.184S
8	2858	5718	0	54663	47M42.235S
9	2859	0	0	17140	47M42.286S
5	2861	0	0	17154	47M42.847S
6	2860	2860	0	37326	47M42.704S
7	2860	5720	0	58341	47M42.007S
8	2859	5720	0	56037	47M42.775S
9	2860	0	0	17157	47M42.873S
5	2843	0	1	17044	47M24.405S
6	2842	2841	0	37994	47M24.277S
7	2842	5684	1	57895	47M24.093S
8	2842	5685	0	55935	47M24.106S
9	2843	0	0	17045	47M24.123S
Média:					
5	2863	0	0.8	17165.2	47M45.058S
6	2862	2861.6	0	36483.2	47M44.904S
7	2861.8	5723.8	0.2	58226.6	47M44.722S
8	2861.6	5724.2	0	55091.4	47M44.860S
9	2862.2	0	0	17221	47M44.941S

Tabela A.1 – Tabelas Modelo A - CMGVT adaptado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2961	0	0	17744	49M24.989S
6	2962	2961	1	36100	49M24.714S
7	2962	0	1	17765	49M24.961S
8	2961	5922	0	59349	49M24.855S
9	0	2961	0	17503	49M24.981S
5	2916	0	1	17485	48M38.505S
6	2915	2915	0	35994	48M38.326S
7	2915	0	0	17498	48M38.328S
8	2915	5830	0	58302	48M38.285S
9	0	2915	0	16877	48M38.296S
5	2943	0	1	17637	49M5.151S
6	2942	2942	0	36192	49M5.021S
7	2943	0	0	17634	49M5.336S
8	2942	5885	0	59460	49M5.144S
9	0	2942	0	17185	49M5.227S
5	2960	0	1	17737	49M22.196S
6	2959	2959	0	36124	49M21.978S
7	2959	0	0	17738	49M22.051S
8	2958	5918	0	59043	49M21.858S
9	0	2958	0	17295	49M21.933S
5	2937	0	0	17600	48M59.118S
6	2935	2937	0	36002	48M58.902S
7	2936	0	0	17603	48M58.785S
8	2936	5871	1	58650	48M58.087S
9	0	2935	0	17107	48M58.672S
Média:					
5	2943.4	0	0.6	17640.6	49M5.992S
6	2942.6	2942.8	0.2	36082.4	49M5.788S
7	2943	0	0.2	17647.6	49M5.892S
8	2942.4	5885.2	0.2	58960.8	49M5.646S
9	0	2942.2	0	17193.4	49M5.822S

Tabela A.2 – Tabelas Modelo B - CMGVT adaptado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2152	0	0	12907	35M54.323S
6	2152	0	0	12907	35M54.178S
7	2152	0	0	12904	35M54.207S
8	0	8606	0	41968	35M54.144S
9	2153	0	0	12922	35M54.189S
5	2147	0	1	12895	35M49.183S
6	2147	0	0	12889	35M49.045S
7	2147	0	0	12878	35M49.029S
8	0	8587	0	42172	35M49.174S
9	2147	0	0	12907	35M49.035S
5	2154	0	0	12922	35M57.044S
6	2155	0	0	12915	35M56.874S
7	2155	0	0	12955	35M56.897S
8	0	8618	0	42162	35M56.801S
9	2154	0	0	12943	35M56.869S
5	2142	0	1	12830	35M43.492S
6	2141	0	0	12845	35M43.365S
7	2141	0	0	12860	35M43.034S
8	0	8564	0	41876	35M43.403S
9	2141	0	0	12859	35M43.353S
5	2149	0	0	12876	35M50.633S
6	2148	0	0	12894	35M50.479S
7	2148	0	0	12894	35M50.476S
8	0	8592	0	41760	35M50.422S
9	2148	0	0	12896	35M50.459S
Média:					
5	2148.8	0	0.4	12886	35M50.936S
6	2148.6	0	0	12890	35M50.788S
7	2148.6	0	0	12898.2	35M50.729S
8	0	8593.4	0	41987.6	35M50.789S
9	2148.6	0	0	12905.4	35M50.781S

Tabela A.3 – Tabelas Modelo C - CMGVT adaptado

A.2 Resultados para o algoritmo Alternativo

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2893	0	1	17353	48M14.901S
6	2893	2891	1	38732	48M14.789S
7	2891	5784	0	55728	48M14.782S
8	2892	5784	0	53475	48M14.091S
9	2893	0	1	17351	48M15.033S
5	2859	0	2	17142	47M40.982S
6	2858	2858	0	38254	47M40.973S
7	2858	5715	0	54789	47M40.853S
8	2858	5716	0	53646	47M40.992S
9	2858	0	0	17151	47M41.069S
5	2863	0	1	17219	47M45.037S
6	2862	2861	0	38726	47M45.111S
7	2861	5724	0	56316	47M45.013S
8	2862	5724	0	53292	47M44.966S
9	2862	0	0	17160	47M45.016S
5	2860	0	1	17215	47M41.688S
6	2859	2858	1	37852	47M41.385S
7	2859	5717	1	56496	47M41.402S
8	2858	5716	0	53535	47M41.297S
9	2859	0	1	17146	47M41.353S
5	2874	0	1	17248	47M57.052S
6	2874	2874	0	39016	47M56.915S
7	2874	5748	0	56028	47M56.079S
8	2874	5747	0	52532	47M56.955S
9	2874	0	0	17248	47M57.135S
Média:					
5	2869.8	0	1.2	17235.4	47M51.932S
6	2869.2	2868.4	0.4	38516	47M51.835S
7	2868.6	5737.6	0.2	55871.4	47M51.626S
8	2868.8	5737.4	0	53296	47M51.660S
9	2869.2	0	0.4	17211.2	47M51.921S

Tabela A.4 – Tabelas Modelo A - Alternativo

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2919	0	1	18491	48M38.217S
6	2919	2920	0	39014	48M38.112S
7	2918	0	1	17833	48M38.118S
8	2920	5836	0	47940	48M37.887S
9	0	2920	0	15154	48M38.071S
5	2919	0	1	18340	48M41.112S
6	2918	2918	1	38656	48M41.008S
7	2918	0	1	17808	48M41.007S
8	2918	5834	0	48671	48M40.989S
9	0	2918	0	16422	48M41.119S
5	2830	0	1	17995	47M32.239S
6	2830	2829	1	37990	47M32.292S
7	2830	0	1	17003	47M32.282S
8	2830	5658	0	47409	47M32.248S
9	0	2830	0	16007	47M32.296S
5	2839	0	1	18040	47M40.586S
6	2838	2838	1	38086	47M40.634S
7	2838	0	0	17102	47M40.448S
8	2838	5675	0	47708	47M40.568S
9	0	2838	0	16106	47M40.495S
5	2965	0	2	17773	49M27.263S
6	2964	2963	1	35428	49M27.088S
7	2964	0	0	17885	49M26.929S
8	2964	5927	0	56528	49M26.917S
9	0	2964	0	16685	49M26.986S

Média:

5	2894.4	0	1.2	18127.8	48M23.883S
6	2893.8	2893.6	0.8	37834.8	48M23.841S
7	2893.6	0	0.6	17526.2	48M23.769S
8	2894	5786	0	49651.2	48M23.722S
9	0	2894	0	16074.8	48M23.793S

Tabela A.5 – Tabelas Modelo B - Alternativo

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2146	0	1	12832	35M42.535S
6	2146	0	0	12838	35M42.378S
7	2146	0	0	12834	35M42.376S
8	0	8582	0	41364	35M42.238S
9	2146	0	1	12860	35M42.388S
5	2158	0	1	12940	35M59.475S
6	2157	0	1	12940	35M59.032S
7	2157	0	1	12959	35M59.357S
8	0	8626	0	41756	35M59.411S
9	2157	0	0	12957	35M59.333S
5	2151	0	1	12896	35M52.779S
6	2150	0	1	12902	35M52.746S
7	2150	0	0	12891	35M52.664S
8	0	8599	0	41456	35M52.671S
9	2150	0	0	12917	35M52.649S
5	2148	0	1	12867	35M49.314S
6	2147	0	0	12872	35M49.252S
7	2147	0	0	12887	35M49.235S
8	0	8588	0	41360	35M49.202S
9	2147	0	0	12880	35M49.192S
5	2126	0	1	12739	35M27.981S
6	2126	0	0	12749	35M27.847S
7	2125	0	0	12714	35M27.803S
8	0	8501	0	41118	35M27.817S
9	2125	0	0	12748	35M27.797S
Média:					
5	2145.8	0	1	12854.8	35M46.417S
6	2145.2	0	0.4	12860.2	35M46.251S
7	2145	0	0.2	12857	35M46.287S
8	0	8579.2	0	41410.8	35M46.268S
9	2145	0	0.2	12872.4	35M46.072S

Tabela A.6 – Tabelas Modelo C - Alternativo

A.3 Resultados para o algoritmo Centralizado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	2003	0	1	20014	33M47.219S
6	2002	2002	1	23272	33M47.001S
7	2003	4004	1	24976	33M47.067S
8	2002	4002	0	25244	33M47.075S
9	2003	0	2	22395	33M47.074S
5	2003	0	1	20008	33M25.229S
6	2003	2001	1	23605	33M24.949S
7	2003	4004	2	22461	33M25.055S
8	2002	4002	0	25335	33M25.077S
9	2003	0	3	22461	33M24.974S
5	2003	0	1	20011	33M25.645S
6	2002	2002	0	24333	33M25.423S
7	2003	4004	1	24125	33M25.358S
8	2002	4004	0	25553	33M25.406S
9	2003	0	1	22671	33M25.403S
5	2003	0	1	20012	33M25.620S
6	2002	2002	0	24401	33M25.523S
7	2003	4004	1	24201	33M25.367S
8	2002	4004	0	25560	33M25.400S
9	2003	0	1	22679	33M25.430S
5	2002	0	0	20008	33M24.836S
6	2002	2001	0	24125	33M24.778S
7	2002	4003	1	24073	33M24.531S
8	2002	4003	1	25276	33M24.538S
9	2002	0	1	22567	33M24.514S

Média:

5	2002.8	0	0.8	20010.6	33M25.110S
6	2002.2	2001.6	0.4	23947.2	33M24.935S
7	2002.8	4003.8	1.2	23967.2	33M24.876S
8	2002	4003	0.2	25393.6	33M24.899S
9	2002.8	0	1.6	22554.6	33M24.879S

Tabela A.7 – Tabelas Modelo A - Centralizado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	1818	0	1	21918	30M19.306S
6	1817	1817	0	23331	30M19.132S
7	1817	0	0	21733	30M18.945S
8	1817	3634	0	24210	30M19.001S
9	0	1817	0	21318	30M18.984S
5	1815	0	2	22027	30M15.094S
6	1814	1813	1	23749	30M15.863S
7	1814	0	1	21599	30M15.808S
8	1814	3626	1	24584	30M15.844S
9	0	1813	0	21243	30M15.882S
5	1834	0	1	22066	30M35.543S
6	1833	1833	2	23863	30M35.039S
7	1833	0	0	21538	30M35.447S
8	1833	3664	1	24293	30M35.341S
9	0	1832	0	21276	30M35.045S
5	1837	0	1	22164	30M38.422S
6	1836	1836	0	23995	30M38.348S
7	1836	0	1	21751	30M38.283S
8	1836	3671	0	24909	30M38.229S
9	0	1836	0	21394	30M38.142S
5	1811	0	1	21904	30M12.606S
6	1810	1810	1	23470	30M12.562S
7	1810	0	1	21879	30M12.537S
8	1810	3618	0	24634	30M12.431S
9	0	1810	0	21357	30M12.003S

Média:

5	1823	0	1.2	22015.8	30M24.363S
6	1822	1821.8	0.8	23681.6	30M24.169S
7	1822	0	0.6	21700	30M24.204S
8	1822	3642.6	0.4	24526	30M24.149S
9	0	1821.6	0	21317.6	30M24.017S

Tabela A.8 – Tabelas Modelo B - Centralizado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	1838	0	0	22024	30M39.891S
6	1838	0	0	22016	30M39.074S
7	1838	0	1	21692	30M39.654S
8	0	7351	0	24661	30M39.809S
9	1838	0	0	21540	30M39.703S
5	1841	0	0	21917	30M42.657S
6	1840	0	0	22035	30M42.517S
7	1840	0	0	21676	30M42.399S
8	0	7360	0	24623	30M42.747S
9	1840	0	1	21688	30M42.454S
5	1856	0	1	21816	30M57.886S
6	1855	0	1	23043	30M57.736S
7	1856	0	0	22130	30M57.939S
8	0	7420	0	25404	30M57.737S
9	1856	0	1	21949	30M57.866S
5	1853	0	1	22109	30M54.136S
6	1852	0	1	22247	30M53.851S
7	1852	0	1	21893	30M53.875S
8	0	7404	0	25112	30M53.864S
9	1852	0	2	21519	30M53.912S
5	2002	0	1	20008	33M24.704S
6	2003	0	1	22762	33M24.436S
7	2002	0	1	22799	33M24.048S
8	0	8006	0	25402	33M24.481S
9	2003	0	1	22591	33M24.501S

Média:

5	1878	0	0.6	21574.8	31M19.855S
6	1877.6	0	0.6	22420.6	31M19.523S
7	1877.6	0	0.6	22038	31M19.583S
8	0	7508.2	0	25040.4	31M19.728S
9	1877.8	0	1	21857.4	31M19.687S

Tabela A.9 – Tabelas Modelo C - Centralizado

ANEXO B – Resultados das Execuções (Experimento 2)

B.1 Resultados para o algoritmo CMGVT original

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	999	0	0	22141	33M16.061S
6	998	998	0	21895	33M15.933S
7	998	1997	0	22062	33M15.838S
8	998	1997	0	33306	33M15.883S
9	998	0	0	11048	33M16.003S
5	999	0	0	22931	33M19.011S
6	999	999	0	21926	33M19.093S
7	999	1998	0	22085	33M18.986S
8	999	1998	0	32819	33M19.029S
9	999	0	0	10974	33M19.097S
5	999	0	0	22934	33M19.028S
6	999	999	0	21920	33M19.004S
7	999	1998	0	22100	33M18.904S
8	999	1998	0	32819	33M18.947S
9	999	0	0	10973	33M19.044S
5	998	0	0	22897	33M16.164S
6	998	998	0	21881	33M16.293S
7	998	1996	0	22126	33M16.088S
8	998	1996	0	32767	33M16.414S
9	998	0	0	10963	33M16.257S
5	999	0	0	22829	33M19.035S
6	999	999	0	21916	33M19.092S
7	999	1998	0	21896	33M18.864S
8	999	1998	0	32808	33M18.975S
9	999	0	0	10974	33M19.008S
Média:					
5	998.8	0	0	22746.4	33M17.860S
6	998.6	998.6	0	21907.6	33M17.883S
7	998.6	1997.4	0	22053.8	33M17.936S
8	998.6	1997.4	0	32903.8	33M17.850S
9	998.6	0	0	10986.4	33M17.882S

Tabela B.1 – Tabelas Modelo A - CMGVT original

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	999	0	0	15541	41M43.091S
6	999	999	0	13500	41M43.893S
7	999	0	0	13512	41M43.974S
8	999	1998	0	42101	41M43.852S
9	0	999	0	26982	41M43.955S
5	1035	0	0	15680	44M26.407S
6	1034	1035	0	14296	44M26.309S
7	1034	0	0	14311	44M26.479S
8	1035	2068	0	44312	44M26.356S
9	0	1035	0	28593	44M26.491S
5	1112	0	0	15417	51M41.421S
6	1113	1112	0	16472	51M41.038S
7	1112	0	0	16475	51M41.334S
8	1113	2225	0	51005	51M41.268S
9	0	1113	0	32926	51M41.325S
5	988	0	0	15203	40M18.467S
6	989	988	0	14102	40M19.033S
7	989	0	0	14136	40M18.678S
8	988	1977	0	42356	40M18.456S
9	0	988	0	27203	40M18.944S
5	972	0	0	15430	37M45.153S
6	973	972	0	12296	37M44.942S
7	972	0	0	12333	37M44.097S
8	973	1945	0	37492	37M44.808S
9	0	973	0	24588	37M45.014S

Média:

5	1021.2	0	0	15454.2	43M10.908S
6	1021.6	1021.2	0	14133.2	43M8.843S
7	1021.2	0	0	14153.4	43M8.912S
8	1021.6	2042.6	0	43453.2	43M8.948S
9	0	1021.6	0	28058.4	43M9.146S

Tabela B.2 – Tabelas Modelo B - CMGVT original

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	999	0	0	10973	33M19.158S
6	999	0	0	10971	33M18.943S
7	999	0	0	10975	33M19.105S
8	0	3996	0	39997	33M19.081S
9	999	0	0	10974	33M19.079S
5	996	0	0	10976	33M19.256S
6	996	0	0	10979	33M19.193S
7	996	0	0	10985	33M19.021S
8	0	3984	0	39926	33M19.023S
9	996	0	0	10980	33M19.149S
5	997	0	0	10982	33M19.204S
6	996	0	0	10979	33M19.213S
7	998	0	0	10981	33M19.198S
8	0	3989	0	40001	33M19.093S
9	998	0	0	10979	33M19.125S
5	998	0	0	10979	33M19.059S
6	998	0	0	10978	33M19.083S
7	997	0	0	10971	33M19.098S
8	0	3991	0	39917	33M18.089S
9	998	0	0	10967	33M19.045S
5	999	0	0	10979	33M18.981S
6	998	0	0	10986	33M18.835S
7	998	0	0	10985	33M18.082S
8	0	3993	0	39985	33M18.904S
9	998	0	0	10990	33M18.096S

Média:

5	997.8	0	0	10977.8	33M19.132S
6	997.4	0	0	10978.6	33M19.053S
7	997.6	0	0	10979.4	33M18.901S
8	0	3990.6	0	39965.2	33M18.838S
9	997,8	0	0	10978	33M18.899S

Tabela B.3 – Tabelas Modelo C - CMGVT original

B.2 Resultados para o algoritmo CMGVT adaptado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	276	0	0	1640	4M33.056S
6	272	272	0	3656	4M33.266S
7	272	548	0	5871	4M33.383S
8	272	544	0	5229	4M33.276S
9	272	0	0	1637	4M33.331S
5	283	0	1	1686	4M41.159S
6	280	278	0	3780	4M41.116S
7	279	561	1	5850	4M41.051S
8	279	560	0	5466	4M41.174S
9	280	0	0	1682	4M41.175S
5	284	0	1	1692	4M42.243S
6	280	281	0	3940	4M42.014S
7	281	564	0	5886	4M42.129S
8	281	561	0	5502	4M42.121S
9	281	0	0	1689	4M42.198S
5	290	0	1	1725	4M47.064S
6	286	286	0	3894	4M47.351S
7	286	576	0	5829	4M47.347S
8	286	572	0	5760	4M47.458S
9	286	0	0	1725	4M47.435S
5	278	0	0	1659	4M36.177S
6	274	274	0	3646	4M35.996S
7	274	552	0	5466	4M35.908S
8	274	548	0	5361	4M35.947S
9	274	0	0	1657	4M35.983S
Média:					
5	282.2	0	0.4	1680.4	4M39.940S
6	278.4	278.2	0	3783.2	4M39.949S
7	278.4	560.2	0.2	5780.4	4M39.964S
8	278.4	557	0	5463.6	4M39.995S
9	278.6	0	0	1678	4M40.024S

Tabela B.4 – Tabelas Modelo A - CMGVT adaptado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	286	0	1	1732	4M47.994S
6	285	285	1	3510	4M47.673S
7	286	0	0	1722	4M47.803S
8	286	570	0	5832	4M47.068S
9	0	286	0	1727	4M47.802S
5	288	0	1	1739	4M49.877S
6	288	287	0	3540	4M49.565S
7	288	0	0	1736	4M49.872S
8	288	576	0	5985	4M49.653S
9	0	288	0	1763	4M49.649S
5	291	0	1	1755	4M52.513S
6	291	290	0	3600	4M52.383S
7	291	0	0	1752	4M52.511S
8	291	582	0	6000	4M52.039S
9	0	291	0	1751	4M52.055S
5	288	0	1	1741	4M49.829S
6	288	287	0	3546	4M49.595S
7	288	0	0	1736	4M49.634S
8	288	576	0	5964	4M49.551S
9	0	288	0	1711	4M49.677S
5	282	0	0	1705	4M43.956S
6	282	282	0	3480	4M43.673S
7	282	0	0	1701	4M43.548S
8	282	564	0	5760	4M43.423S
9	0	282	0	1707	4M43.041S

Média:

5	287	0	0.8	1734.4	4M48.834S
6	286.8	286.2	0.2	3535.2	4M48.578S
7	287	0	0	1729.4	4M48.674S
8	287	573.6	0	5908.2	4M48.347S
9	0	287	0	1731.8	4M48.445S

Tabela B.5 – Tabelas Modelo B - CMGVT adaptado

PL	*M. enviadas*	*M. recebidas*	*M. perdidas*	*M. GVT*	*Tempo*
5	210	0	1	1271	3M31.065S
6	209	0	0	1265	3M31.012S
7	209	0	0	1263	3M30.877S
8	0	836	0	4292	3M30.088S
9	209	0	0	1270	3M30.819S
5	214	0	1	1295	3M35.106S
6	213	0	0	1289	3M34.948S
7	214	0	0	1288	3M34.953S
8	0	854	0	4388	3M34.797S
9	214	0	0	1293	3M34.925S
5	211	0	0	1275	3M32.135S
6	210	0	1	1270	3M31.984S
7	211	0	0	1271	3M31.963S
8	0	841	0	4288	3M31.877S
9	210	0	0	1270	3M31.981S
5	209	0	0	1264	3M30.515S
6	208	0	0	1257	3M30.395S
7	208	0	0	1262	3M30.578S
8	0	833	0	4216	3M30.421S
9	208	0	0	1264	3M30.056S
5	209	0	0	1273	3M32.105S
6	208	0	0	1270	3M31.942S
7	209	0	0	1274	3M31.947S
8	0	835	0	4280	3M31.884S
9	209	0	0	1267	3M31.933S

Média:

5	210.6	0	0.4	1275.6	3M32.185S
6	209.6	0	0.2	1270.2	3M32.056S
7	210.2	0	0	1271.6	3M32.064S
8	0	839.8	0	4292.8	3M31.812S
9	210	0	0	1272.8	3M31.943S

Tabela B.6 – Tabelas Modelo C - CMGVT adaptado