Verificação formal de protocolos TDMA quanto a características de tempo real e tolerância a falhas

Alberto Rubens Beckler, Sérgio Vale Aguiar Campos

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
{rubinho, scampos}@dcc.ufmg.br

Abstract *- This paper describes a methodology for modeling TDMA protocols and possible faults that can occur in fault tolerant real time systems using automatic methods of verification. This allows guarantee several qualitative properties and quantitative upper bound time constraints. As case study, presents the modeling for TTP/C protocol, identifying the upper bound for reach agreement for group-membership property in a fault scenario.*

Resumo *- Este trabalho apresenta uma metodologia para representar protocolos TDMA e possíveis falhas que possam ocorrer em sistemas de tempo real tolerantes a falhas utilizando métodos automáticos de verificação. Isto permite garantir diversas propriedades qualitativas e resultados quantitativos como o limite superior de tempo para atingir tais propriedades. Como estudo de caso, apresenta a representação do protocolo TTP/C, determinando o limite de tempo máximo para se atingir a auto-estabilização das visões que cada nodo possui do sistema (Group-membership), possibilitando garantir o tempo máximo de recuperação do sistema, em um cenário de falhas.*

1. Introdução

Recentemente, tem havido uma crescente utilização de protocolos TDMA (*Time division Multiple Access*) em sistemas distribuídos para aplicações de tempo real, como em telefonia celular, por exemplo. Esta estratégia se baseia no escalonamento do canal de comunicação entre os nodos do sistema distribuído, o que torna possível determinar um limite mínimo de qualidade de serviço na comunicação. Cada nodo do sistema possui periodicamente um *slot* TDMA, que é um intervalo de tempo onde lhe é permitido acessar o canal de comunicação e difundir mensagens aos outros nodos do sistema.

Uma outra propriedade importante desta estratégia é permitir a detecção de falhas em algum dos nodos do sistema quando não é possível receber uma mensagem dentro do intervalo de tempo previamente definido, permitindo sua utilização em sistemas tolerante a falhas.

A partir da detecção da falha de um nodo, é possível reconfigurar o sistema distribuído de forma a suprir a falha de um dos nodos. Neste cenário de falha, o sistema permanece em um estado inválido até que a reconfiguração esteja concluída. Esta estratégia de recuperação de falhas, denominada redundância dinâmica [6], pode ser utilizada em sistemas de tempo real desde que o atraso gerado pelo processo de recuperação não viole os limites de tempo (*deadline)* de execução das tarefas do sistema.

Um exemplo é o protocolo TTP/C [5] (*time-triggered protocol - class C*), recentemente desenvolvido por um consórcio de empresas denominado TTTech para aplicação nas indústrias automobilística e de aviação. Este protocolo prove uma infra-estrutura básica para a redistribuição de tarefas entre os nodos restantes do grupo, como a propriedade de *Group Membership*, que garante a mesma visão em cada nodo de todos os nodos corretos em um *cluster*. Entretanto, em um cenário de falha, o protocolo estabelece que o tempo máximo para a

estabilização destra propriedade, ou seja, que todos os nodos detectem a falha de um nodo, é de 2n-1 slots TDMA, onde n é o número de nodos do sistema.

Estabelecer o intervalo de tempo máximo em que o protocolo permanece em um estado inválido e o atraso provocado pelo gerenciamento do sistema de tolerância a falhas é fundamental para a execução correta de um protocolo para sistemas tolerante a falhas em aplicações de tempo real. Como as falhas ocorrem de maneira aleatória e incontrolável, apenas com testes e simulações é praticamente impossível validar protocolos para utilização em situações críticas.

Existem ferramentas automáticas capazes de realizar a verificação formal de diversas propriedades em sistemas distribuídos. Estas ferramentas modelam a execução de um protocolo em fórmulas matemáticas, de maneira precisa e não ambígua. Realizam operações matemáticas sobre estes modelos a fim de determinar se uma determinada propriedade não é violada. Estas ferramentas produzem resultados mais significativos quando utilizadas ainda na fase de concepção do produto, detectando possíveis erros de projeto reduzindo os custos de correção e produção de protótipos.

Para aumentar a utilização destas ferramentas por profissionais de diversas áreas da ciência da computação é necessário o desenvolvimento de metodologias simples, generalizando soluções para a representação de aspectos padrões a uma determinada área, abstraindo de detalhes específicos, a fim de diminuir o tempo e a complexidade do aprendizado de tais ferramentas.

Neste cenário, este trabalho descreve uma metodologia para a representação das características básicas presentes no desenvolvimento de protocolos para sistemas de tempo real tolerante a falhas na ferramenta de verificação formal VERUS [7], como a modelagem e a topologia do *cluster*, as mensagens enviadas por cada um dos nodos, além da representação do tempo e das falhas que possam ocorrer no sistema. Tornando possível garantir diversas propriedades e os limites inferiores e superiores de tempo para atingi-las.

A fim de comprovar sua aplicabilidade, utilizou-se a metodologia proposta para verificar e comprovar o limite de tempo de 2n-1 slots de transmissão TDMA para a auto-estabilização do *Group-Membership* para o protocolo TTP/C, conforme a especificação do protocolo.

2. Trabalhos relacionados

2.1 Propriedades de protocolos para sistemas de tempo real tolerante a falhas.

Para realizar a verificação formal de protocolos é necessário determinar quais propriedades devem ser atendidas a fim de garantir o funcionamento correto do protocolo.

K. Birman [8], cita as condições básicas que devem ser atendidas para que sistemas distribuídos permitam uma operação continuada mesmo na presença de falhas. Dentre estas características destaca-se a propriedade de *Group-Membership* [4], que é a necessidade de gerenciar a formação e coordenação de grupos de nodos corretos, tanto na presença de falhas, quanto durante a recuperação do sistema. Este conhecimento, além de ser essencial por motivos de enumeração e endereçamento dos seus membros, é fundamental, sobretudo para a recuperação de uma falha no sistema.

Sistemas *time-triggered* [5] [11] são sistemas distribuídos onde todo o controle de um protocolo é disparado em instantes pré-determinados do tempo. Existem também sistemas *event-triggered* [11], onde o controle ocorre na presença de eventos, como o recebimento de uma mensagem, por exemplo.

Sistemas time-triggered podem detectar uma falha não apenas pelo recebimento de uma mensagem incorreta, mas também pela ausência de uma mensagem após o intervalo de transmissão TDMA de um nodo. Assim, apesar de uma falha ser um evento do sistema, a detecção de uma falha pelos outros nodos de um cluster é controlada por protocolos *time-triggered*.

Recentemente, tem havido um crescente desenvolvimento de protocolos de redundância ativa para aplicações tolerante a falhas em sistemas de tempo real. Como os protocolos TTP/C [11] e TTCAN [12].

A principal estratégia destes protocolos é controlar o acesso de um nodo ao canal de comunicação, a fim de evitar as colisões entre as mensagens e determinar um tempo limite para a entrega de uma mensagem pelo canal de comunicação.

O protocolo TTP/C utiliza uma estratégia TDMA de acesso ao canal de comunicação. Foi desenvolvido especificamente para a área de tolerância a falhas, inclui diversas características que auxiliam o projeto de aplicações, como sincronização de relógio, mecanismos de confirmação do recebimento de mensagens, detecção de falhas e de gestão de grupos.

A verificação formal do protocolo TTCAN quanto à auto-estabilização da propriedade de *group-membership* também foi realizada e encontra-se na versão completa deste artigo.

3. Verificação formal

Ferramentas de verificação formal representam o funcionamento de um circuito ou a execução de um algoritmo em modelos matemáticos, de maneira precisa e não ambígua, realizam operações matemáticas sobre estes modelos a fim de verificar se uma determinada propriedade é satisfeita ou não.

Estas ferramentas são baseadas principalmente em técnicas de *simbolic model checking* [2]. Representando a execução de um algoritmo, seqüencial ou distribuído, em um grafo onde cada estado corresponde a uma possível configuração de valor para as variáveis do modelo, e a transição entre os nodos correspondendo a uma mudança de valor para uma determinada variável.

Por se basear na enumeração de todos os possíveis estados e transições do modelo, estas ferramentas necessitam de uma representação eficiente do grafo que representa o algoritmo. O estado da arte destas ferramentas utiliza grafos de decisão binários - BDDs (*Binary Decision Diagram)* [2] para a representação do modelo. Um BDD é uma representação canônica de fórmulas booleanas. Além disso, possuem um arcabouço de operações sobre estas estruturas a fim de possibilitar uma representação eficiente das transições entre os estados, como operações lógicas booleanas e o caminhamento entre os nodos do grafo[2].

Estas operações permitem realizar uma busca exaustiva aos nodos do grafo e garantir se uma determinada propriedade é satisfeita ou não, garantido a correção de um algoritmo ou, em caso de erro, apresentar o caminho percorrido que provocou a violação de uma propriedade.

Aplicações críticas exigem técnicas de tolerância a falhas tanto em hardware quanto em software, baseadas em redundância, o que aumenta sua complexidade exigindo a utilização de ferramentas para auxiliar na análise do processo. Além disso, gerenciar os recursos redundantes provoca um atraso na execução do sistema aumentando a complexidade da análise dos limites de tempo em sistemas críticos de tempo real.

Para realizar a verificação formal de sistemas de tempo real é necessária a utilização de ferramentas que possuam construções para a representação do tempo. Existem ferramentas desenvolvidas com este propósito como a ferramenta de verificação formal VERUS [7] utilizada neste trabalho. Esta ferramenta possui uma representação discreta do tempo. Esta simplificação é necessária para permitir a modelagem e a verificação de sistemas complexos, e não provoca prejuízos quanto a qualidade da modelagem, visto que em sistemas time-triggered, é necessário representar de forma discreta os intervalos de tempo de controle do protocolo.

4. Metodologia para verificação de protocolos para sistemas de tempo real tolerante a falhas.

Garantir que uma propriedade é satisfeita durante toda a execução de um protocolo, mesmo em caminhos pouco prováveis de execução, além de estabelecer os limites superiores de tempo para atingi-las é fundamental para um projetista durante a elaboração de um protocolo.

Em sistemas de tempo real, estes parâmetros são fundamentais para estabelecer propriedades, como a velocidade de comunicação e número máximo de nodos de um cluster. Entretanto, controlar a simulação de todos os caminhos de execução de um protocolo e todos os instantes possíveis para a ocorrência de falhas pode levar a um enorme número de possibilidades. A utilização de técnicas de *simbolic model checking* possibilita a representação eficiente da estrutura de dados utilizada para modelar todas estas possibilidades.

Este trabalho propõe a utilização da ferramenta VERUS para representar as principais características de um cluster telerante a falhas, o protocolo TDMA executado, a representação da passagem do tempo, necessária para garantir propriedades temporais e as possíveis falhas que possam ocorrer no sistema.

4.1 Modelagem do protocolo

Para representar o algoritmo, a ferramenta VERUS possui uma linguagem muito semelhante a C. Permite a execução em paralelo de diversos processos e a representação discreta do tempo. A tabela 1 representa a modelagem de um protocolo TDMA para a linguagem da ferramenta VERUS.

Tabela 1. Modelagem do cluster tolerante a falhas em Verus

CLUSTER	VERUS
Nodo	Processo
Registradores	Variáveis locais
Slot de tempo TDMA	wait (1);
Mensagens	Variáveis globais
Propriedades a serem verificadas	Lógica CTL
Falhas	Modelo de falhas

Como a verificação é realizada sobre um modelo, e não sobre a implementação do protocolo, a facilidade e a qualidade da modelagem são importantes tanto para ter certeza que o modelo implementa realmente o protocolo, quanto para a análise dos resultados da verificação do modelo, mapeando as informações obtidas da ferramenta para o protocolo em questão. Como a ferramenta VERUS utiliza uma linguagem muito semelhante a C, a tarefa de modelagem do protocolo torna-se bastante simplificada. A modelagem das construções básicas para a representação do sistema distribuído está descrita de forma a fornecer ao projetista de um protocolo uma metodologia simples de representação de seu projeto.

Cada nodo de um sistema distribuído é representado como um processo em VERUS e seus registradores como variáveis locais, inteiras e booleanas.

Os processos trocam informações (mensagens) entre si através da atribuição de valores a variáveis globais. É importante ressaltar a necessidade de representar quando uma determinada mensagem não é mais válida cancelando o valor de uma variável após seu recebimento.

VERUS possui o comando wait(1) que representa a passagem de uma unidade de tempo. Os processos executam em modo de passo, ou seja, os trechos de código de cada processo compreendidos entre comandos wait(1) são executados simultaneamente de maneira atômica. Neste ponto de sincronização, os processos tomam conhecimento de alterações em variáveis globais do sistema. Assim, o trecho do algoritmo que deve ser executado em um intervalo TDMA do protocolo deve ser representado entre dois comandos wait(1). Desta forma, é possível determinar o número de slots e, por conseqüência, o tempo necessário para atingir um determinado estado do sistema.

A ferramenta VERUS representa toda a execução do algoritmo em fórmulas booleanas, utilizando para isto um grafo de decisão binária (BDD). Faz uma busca exaustiva a este grafo para verificar se uma determinada propriedade, descrita em lógica CTL [1], é satisfeita ou não.

É possível determinar também o intervalo de tempo máximo entre dois estados do modelo, utilizando a propriedade MAX (p, q), que determina o número máximo de transições entre o momento em que uma propriedade p se torne verdadeira até que q também seja.

Para representar todas as possíveis execuções de um algoritmo, uma atribuição a uma variável pode ser feita a partir do comando select {0, 1, 2 ...}. Desta forma, é gerado um caminho no grafo para cada possível valor atribuído.

VERUS fornece ainda o caminho que o algoritmo percorreu até atingir uma propriedade ou uma violação, o que favorece o entendimento do algoritmo e sua depuração.

Para aumentar a eficiência da ferramenta e, por conseqüência, o tamanho dos modelos que ela é capaz de verificar é necessário realizar algumas otimizações. Isto ocorre porque o tamanho da estrutura de dados utilizada para representar o modelo é dependente de fatores como a quantidade de bits necessários para representar um inteiro e da ordem em que as variáveis são analisadas.

Uma variável inteira é representada como um vetor de variáveis booleanas, portanto, para reduzir o tamanho do modelo é muito importante utilizar o menor número de bits necessários para representar o maior inteiro do modelo, reduzindo assim o número de variáveis analisadas.

O tamanho de um BDD sofre enormes alterações devido a ordem em que as variáveis são analisadas pela ferramenta. Infelizmente, determinar a ordenação ótima para um determinado modelo é um problema NP-Completo [2]. Entretanto, existem técnicas para aproximar o modelo de uma ordenação ótima.

A primeira medida a ser implementada é analisar inicialmente as variáveis que estão mais relacionadas, por exemplo, analisar as variáveis que representam uma mensagem das variáveis do processo que recebe, ou envia estas mensagens. Após esta etapa é necessário utilizar sobre esta ordenação o recurso de reordenação automática presente na ferramenta.

4.2 Modelo de falhas

Faz parte da especificação de um protocolo para sistemas tolerantes a falhas a descrição de todas as falhas que o sistema deverá suportar. A verificação formal do protocolo deve ser direcionada a garantir que as falhas propostas neste modelo são efetivamente suportadas pelos recursos redundantes, inclusive que os *deadlines* das tarefas não serão violados. É uma tarefa relativamente simples descrever o modelo de falhas de um protocolo em VERUS. A geração e o controle de todas as possíveis falhas em um sistema são realizados a partir da declaração de variáveis de controle e do comando *select* que gera todas as possibilidades necessárias para uma atribuição a estas variáveis.

```
nodofalho = select {0, 1, 2, 3, 4};
instante  = select {1, 2, 3, 4};

if ( instante == 2 && nodofalho == 1)
      Aok = false;
wait(1);
```

Fig 1. Representação de uma falha de omissão em VERUS.

Como a falha em um dispositivo pode ocorrer a qualquer instante do tempo, é necessário representar todos os possíveis instantes para a presença de uma falha. O controle de uma falha de omissão [6], por exemplo, é gerada atribuindo false à variável local que representa a visão de um processo sobre seu próprio funcionamento. Cada passo do algoritmo é executado apenas se seu estado for correto. A figura 1 apresenta um trecho de código em VERUS que representa a geração e o controle de uma falha de omissão.

A variável *nodofalho* controla qual dos nodos de um *cluster* irá falhar. A variável *instante* determina em qual intervalo de transmissão TDMA ocorrerá a falha, e a variável *Aok* representa a visão de um nodo sobre seu funcionamento. Desta forma, atribuições às variáveis: *instante* e *nodofalho*, através do comando select, permite a representação de um possível caminho de execução no modelo. O exemplo da figura 1 representa uma falha de omissão para o nodo A (nodofalho = 1) no intervalo de transmissão do nodo B (instante = 2). A ferramenta gera

todas as possíveis combinações para as variáveis e verifica se as propriedades são satisfeitas em todas as suas execuções.

Para efeito de modelagem de sisemas *time-triggered*, falhas temporais são interpretadas da mesma forma que falhas de omissão. Entretanto, sistemas TDMA devem possuir mecanismos que evitem o envio de mensagens fora do intervalo de transmissão do nodo (*bus guardians*).

O controle sobre a duração de uma falha pode ser realizado através de atribuições sucessivas a uma variável, permitindo a modelagem de falhas transientes.

5. Estudo de caso

5.1 Verificação da propriedade de *Group Membership* para o protocolo TTP/C

5.1.1 Descrição do protocolo

O TTP/C [5] (time-triggered protocol class C) é um protocolo usado para interconexão de micro-controladores através de uma estratégia TDMA de acesso ao canal de comunicação, o C indica que o protocolo atende às características da indústria automobilística. Encontra-se em fase de especificação por um consórcio de empresas chamado TTTech.

Este protocolo conecta diversas centralinas eletrônicas no interior de um veículo, estas centralinas controlam diversas tarefas em um automóvel, como: suspensão eletrônica, direção eletrônica, freios ABS, Air Bag, computador de bordo, injeção eletrônica, vidros elétricos etc.

Como as tarefas executadas são críticas, uma determinada falha de um controlador pode levar a conseqüências terríveis. O principal objetivo do protocolo é fornecer um meio de utilizar os recursos de outras centralinas para realizar as tarefas de um controlador com defeito, tornando o sistema tolerante a falhas. Para isto, deve haver um determinado limite superior de tempo para que todos os controladores corretos detectem a falha, distribua as tarefas e atue sobre o dispositivo em questão, antes que seja impossível evitar um acidente.

Cada controlador correto é denominado um nodo em um cluster. As mensagens são enviadas em broadcast aos outros nodos do cluster. O protocolo tolera apenas uma única falha a cada vez no sistema. Garante um tempo máximo para a detecção de uma falha e que todos os nodos corretos formarão um novo cluster.

5.1.2 Implementação do Group Membership

Para manter uma visão estável dos membros corretos de um cluster o TTP/C utiliza um mecanismo de confirmação implícita de mensagens. Cada nodo de um cluster (digamos A) envia em seu slot de transmissão sua visão do grupo. Seu sucessor (digamos B) é responsável pela confirmação. Se o nodo A estiver na visão de B, então sua mensagem foi enviada corretamente. Senão, algum erro ocorreu; ou A não enviou corretamente seu broadcast, ou B está com problemas de recepção. Para solucionar este impasse, o próximo nodo (digamos C) ao enviar sua visão em broadcast informa qual problema foi detectado. Se A estiver na visão de C, então o problema estava em B, senão, o problema estava em A.

5.1.3 Aplicação da metodologia

A metodologia apresentada foi utilizada para modelar e representar o mecanismo de confirmação implícita do protocolo TTP/C com 4 nodos. A aplicação da metodologia garante apenas as propriedades para esta instância de implementação do protocolo. Entretanto, a partir do pior caso para a ocorrência de uma falha, é possível generalizar os resultados para um número arbitrário de nodos. Na literatura, existem outras modelagens que utilizam esta estratégia como a verificação do protocolo PCI [7]. O código completo da modelagem pode ser obtido sob demanda.

5.1.4 Análise dos resultados

Para facilitar a análise dos resultados foi desenvolvida uma ferramenta responsável capaz de extrair os resultados dos contra-exemplos fornecidos pela ferramenta em formato texto para uma planilha eletrônica, melhorando sua visualização e entendimento. A partir desta visualização é possível, por exemplo, fazer uma análise do pior caso de execução de uma instancia do problema e, se possível, extrair resultados genéricos para um número qualquer de nodos que executam o algoritmo.

A figura 2 apresenta um exemplo de saída da ferramenta para o pior caso de falha para o nodo B, descrito pela propriedade MAX (!pB.Bok, !pA.Bok && !pC.Bok && !pD.Bok). Responsável por determinar o intervalo máximo de tempo desde que haja uma falha no nodo B até que todos os outros nodos de um cluster atualizem sua visão do grupo detectando a falha.

Cada estado é representado pelos valores de todas as variáveis em cada slot TDMA. As variáveis apresentadas no exemplo descrevem as visões de cada nodo sobre o grupo. É importante ressaltar que estas visões permanecem diferentes desde a ocorrência de uma falha até que todos os nodos cheguem a um consenso.

A ferramenta é capaz de verificar não somente propriedades qualitativas, como a diferença entre as visões de cada nodo sobre o grupo, mas também propriedades quantitativas como o tempo necessário para se atingir a auto-estabilização da propriedade de *group-membership*.

		VISAO DO NODO A				VISAO DO NODO B				VISAO DO NODO C				VISAO DO NODO D			
SLOT	WC	pA.Aok	pA.Bok	pA.Cok	pA.Dok	pB.Aok	pB.Bok	pB.Cok	pB.Dok	pC.Aok	pC.Bok	pC.Cok	pD.Dok	pD.Aok	pD.Bok	pD.Cok	pD.Dok
	1	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
A	2	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
B	3	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
C	4	True	True	True	True	True	False	True	True	True	True	True	True	True	True	True	True
D	5	True	True	True	True	True	False	True	True	True	True	True	True	True	True	True	True
A	2	True	True	True	True	True	False	True	True	True	True	True	True	True	True	True	True
B	3	True	True	True	True	True	False	True	True	True	True	True	True	True	True	True	True
C	4	True	True	True	True	True	False	True	True	True	False	True	True	True	True	True	True
D	5	True	True	True	True	True	False	True	True	True	False	True	True	True	False	True	True
A	2	True	False	True	True	True	False	True	True	True	False	True	True	True	False	True	True

Fig 2. Contra-exemplo para a execução do protocolo no pior cenário de ocorrência de uma falha.

Foram geradas falhas em cada um dos processos em todos os slot de tempo, conforme o exemplo da figura 1. Os resultados mostram que o tempo máximo para a detecção de uma falha por todos os outros nodos é de 7 slots de tempo.

A generalização dos resultados obtidos para um número arbitrário de nodos pode ser obtida a partir da análise da saída da ferramenta. No contra-exemplo anterior, nota-se que o pior cenário ocorre quando o nodo B falha logo no slot de transmissão do nodo C. Assim, os outros nodos só podem perceber que existe uma falha em B no próximo ciclo, quando B deixar de enviar seu sinal de vida ao grupo, gastando n slots de tempo. Segundo o protocolo TTP/C cada nodo checa sua caixa de mensagens e atualiza sua visão do grupo imediatamente antes de iniciar seu slot de transmissão, assim, são necessários n-1 slots até que todos os nodos atualizem em sua visão a falha no nodo B, iniciando o protocolo de recuperação de falha e redistribuição de tarefas dependente de aplicação cujo atraso provocado pela troca de mensagens também pode ser verificado pela metodologia proposta neste trabalho.

6. Conclusões

Este trabalho apresenta uma metodologia para representar protocolos TDMA utilizando a ferramenta de verificação formal VERUS, tornando possível verificar diversas propriedades, como características de tempo real e tolerância a falhas. Uma determinada instância do algoritmo é representada através de uma linguagem simples e muito semelhante a sua implementação real, ao invés de utilizar outras estratégias de verificação que criam modelos abstratos para a representação das propriedades exigindo uma grande experiência do usuário em

verificação formal e cujo modelo se torna, apesar de correto, muito diferente de sua implementação.

A metodologia apresentada foi utilizada para verificar diversos protocolos para sitemas de tempo real tolerantes a falhas, como protocolo TTCAN e um novo protocolo para interconexão de centralinas eletrônicas em automóveis, TTP/C, descrito neste trabalho, garantindo que o tempo máximo para a detecção de uma falha de omissão pelo por todos os nodos que executam o protocolo é de 2n - 1 slots de tempo, validando sua especificação.

A passagem do tempo é representada de forma discreta pela ferramenta VERUS. Esta simplificação tem como objetivo a utilização da ferramenta na verificação de sistemas complexos. Entretanto, a representação discreta do tempo não prejudica a qualidade da modelagem sendo possível determinar a quantidade de slots TDMA necessárias para se atingir uma determinada propriedade.

É importante ressaltar que, além de garantir se o sistema funciona ou não, é possível fornecer um resultado quantitativo sobre a execução do protocolo. Garantindo os limites máximos de tempo gastos para o protocolo atingir diversas propriedades. Desta forma, um projetista pode validar ou descartar uma determinada estratégia de tolerância a falhas ainda durante a fase de projeto. Para o caso do TTP/C por exemplo, pode ser necessário diminuir o intervalo de tempo TDMA para que o protocolo atenda as necessidades de uma determinada aplicação, entretanto, será necessário uma rede de comunicação mais rápida e uma maior performance dos nodos para que sejam capazes de realizar suas tarefas dentro de um intervalo menor de tempo.

7. Referências

[1] CLARKE, E. EMERSON, E. Design and synthesis of syncronization skeletons using branching time temporal logic. **Workshop on logics of programs**, v.131, of Lectures Notes in Computer Science, p.52-71. 1981.

[2] BRYANT, R. E. Graph-based algorithms for boolean function manipulation. **IEEE Transactions on Computers**, v.C-35, n.8, 1986.

[3] JOHNSON, B. **Design and Analysis of Fault-Tolerant Digital Systems**. 1.ed. Addison-Wesley series in electrical and computer engineering, 1989. 584p.

[4] CRISTIAN, F. Reaching Agreement on Processor Group Membership in Synchronous Distributed System. **Distributed Computing**, v.4, p.175-187, 1991.

[5] KOPETZ, H. GRUNSTEIDL, G., TTP - A protocol for fault-tolerant real time systems, **IEEE Computer**, v.27#1, p.14-23. Jan 1994.

[6] JALOTE, P., **Fault tolerance in distributed Systems**. Prentice-Hall, 1994. 432p.

[7] CAMPOS, S. et al. Verifying the performance of the PCI local bus using symbolic techniques. **International Conference on Computer Design**, 1995.

[8] BIRMAN, K. Buildin secure and reliable network applications. **Manning Publications Co**, 1996.

[9] CAMPOS, S. CLARK, E. MINEA, M. The Verus tool: a quantitative approach to the formal verification of real-time systems. **Conference on Computer Aided Verification**, 1997.

[10] PASCOE, J. LOADER, R. SUNDERAM, V. Working towards the agreement problem protocol verification environment. **Communications Process Architectures**, 2001.

[11] TTP/C Specification 1.0, Disponível em:<www.tttech.com>, Acesso em 26 mar. 2003.

[12] ISO.Road Vehicles Controller Area Network (CAN) Part 4: TimeTriggered Comunication. Standard ISO/CD 11898-4, International Organization for Standardization, 2001