MODELAGEM E ANÁLISE FORMAL DE ALGUMAS FUNCIONALIDADES DE UM PROTOCOLO DE TRANSPORTE ATRAVÉS DAS REDES DE PETRI

MARCOS GILTON MIRANDA MARTINS

Dissertação apresentada ao Instituto Nacional de Telecomunicações – INATEL, como parte dos requisitos para obtenção do Título de Mestre em Engenharia de Telecomunicações.

Santa Rita do Sapucaí

Dezembro/2003

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em 22 / 12 / 2003,

pela comissão julgadora:

Prof. Dr. Adonias Costa da Silveira – DTE, Inatel.

Prof. Dr. Pedro Frosi Rosa – UFU, Universidade Federal de Uberlândia.

Prof. Dr. Anilton Salles Garcia – DTE, Inatel.

Coordenador do Curso de Mestrado

Ao Senhor Deus da minha salvação,

À Minha Família,

À Minha Pátria.

AGRADECIMENTOS

A Deus, por me ter dado resignação e perseverança para terminar esta dissertação e por Jesus Cristo meu salvador e santificador na verdade.

Ao meu pai e à minha mãe, pelo carinho, paciência e amor.

À minha tia, Maria da Graça Domingues Martins (PROEP/MEC), pelo apoio.

Aos irmãos da Igreja Presbiteriana do Brasil que estiveram comigo e muito me ajudaram espiritualmente.

Ao INATEL, pela oportunidade oferecida. (Parceiros para toda vida!)

Ao professor Dr. Adonias Costa da Silveira, pela coordenação do curso e por presidir a banca de avaliação desta dissertação, por força das circunstâncias.

Ao professor José Geraldo, Pró-Diretor Acadêmico, pelo auxílio nas correções gramaticais desta dissertação.

Ao professor Dr. Miguel Menasche, pela orientação prestada na elaboração desta dissertação, enquanto professor no INATEL, e pelas instruções nas disciplinas TP505 e TP600.

Ao professor Dr. Anilton Salles Garcia, pelas instruções nas disciplinas TP508, TP519, TP520, TP524 e TP525.

Ao professor Dr. Carlos Alberto Ynoguti, pelas instruções em TP501.

Às secretárias Cristina e Robélia, pela paciência.

Aos funcionários do CICT – Centro de Informações Científicas e Tecnológicas, aos funcionários do SEE – Seção de Editoração Eletrônica e demais funcionários do INATEL pela amizade e auxílio prestado.

Ao amigo Joe, pelo incentivo e pelos livros [Coll01] e [Russ00].

E, finalmente, aos colegas e amigos do mestrado pelo companheirismo.

Lista de Figuras

Lista de Tabelas

Lista de Definições e Siglas

AC	-	*Authentication Center*
AIN	-	*Application Intelligent Network*
API	-	*Application Program Interface*
ATM	-	*Asynchronous Transfer Mode*
CA	-	*Call Agent*
Call Id	-	Identificador de Chamada
CCS	-	*Common Channel Signaling*
CHUNK	-	Unidade de informação (Mensagem) do SCTP
CPN	-	*Colored Petri Net*
Cross Bar	-	Matriz de comutação temporal
CTP	-	*Common Transport Protocol*
DFD	-	Diagrama de Fluxo de Dados
EIR	-	*Equipment Identity Register*
Endpoint	-	Entidade lógica de envio e recebimento de pacotes
ESTELLE	-	Linguagem de Descrição Formal, definida na ISO 9074
GTT	-	*Global Title Translation*
HLR	-	*Home Locater Register*
IETF	-	*Internet Engineering Task Force*
IN	-	*Intelligent Network*
INA	-	*Information Network Architecture*
INAP	-	*Intelligent Network Application Part*
IP	-	*Internet Protocol*
IPv6	-	*Internet Protocol Versão 6*
IP(2)	-	*Intelligent Peripheral*
ISDN	-	*Integrated Services Digital Network*
ISO	-	*International Organization for Standardization*
ISP	-	*Internet Service Provider*
ISUP	-	*ISDN User Part*

ITU	-	*International Telecommunication Union*
ITU-T	-	ITU – *Telecommunications Standardization Sector*
IUA	-	ISDN Q.921 – *User Adaptation Layer*
K.A.P.	-	*Karl Adam Petri*
LIDB	-	*Line Information DataBase*
LNP	-	*Local Number Portability*
MAC	-	*Message Authentication Code*
M2UA	-	MTP2 – *User adaptation Layer*
M3UA	-	MTP3 – *User Adaptation Layer*
MAC	-	*Message Authentication Code*
MAP	-	*Mobile Application Part*
MD5	-	Algoritmo de verificação de integridade de mensagens
MG	-	*Media Gateway*
MGC	-	*Media Gateway Control*
MGCP	-	*Media Gateway Control Protocol*
MPLS	-	*MultiProtocol Label Switching*
MSC	-	*Mobile Switching Center*
MTP	-	*Message Transfer Part*
OG	-	*Ocorrency Graph*
OSI	-	*Open Systems Interconnection*
PDU	-	*Protocol Data Unit*
PN	-	*Petri Net*
PSTN	-	*Public Switched Telephone Networks*
P/T	-	*Place/Transition*
Q.931	-	*Call Signaling Protocol*
QoS	-	*Quality of Service*
RdP	-	Rede de Petri
RdPs	-	Redes de Petri
RFC	-	*Request For Comments*
RG	-	*Residential Gateway*
RSVP	-	*Resource Reservation Protocol*
RTO	-	*Retransmission Time-Out*

RTT - *Round-Trip Time*
SCN - *Switch Circuit Network*
SCP - *Service Control Point*
SCCP - *Signaling Connection Control Part*
SCTP - *Stream Control Transmission Protocol*
SDL - *System Description Language*
SG - *Signaling Gateway*
SLA - *Service Level Agreements*
SMS - *Short Message Service*
SMSC - *Short Message Service Center*
SP - *Signaling Point*
SS6 - *Signaling System Number 6*
SS7 - *Signaling System Number 7*
SSN - *Subsystem Number*
SSP - *Service Switching Point*
Stream - "Fluxo". Canal lógico unidirecional no SCTP
STP - *Signaling Transfer Point*
SUA - SCCP – *User Adaptation Layer*
TCAP - *Transaction Capabilities Application Part*
TCB - *Transmission Control Block*
TCP - *Transmission Control Protocol*
TDF - Técnica de Descrição Formal
TDM - *Time Division Multiplex*
ToS - *Type of Service*
UDP - *User Datagram Protocol*
ULP - *User Lower Protocol*
URL - *Uniform Resource Locators*
VLR - *Visitor Location Register*
VoIP - *Voice over IP*
WAP - *Wireless Application Protocol*
WIN - *Wireless Intelligent Network*

Lista de Símbolos

#	-	Número
#(t_j, Pré (p_i))	-	Número de transições $\mathbf{t_j}$ em **Pré** de $\mathbf{p_i}$
δ	-	Função estado seguinte de uma RdP
θ	-	Função associando um intervalo fechado sobre os reais não-negativos $\mathbf{[a_i, b_i]}$ à cada transição $\mathbf{t_i \in T}$.
λ	-	Seqüência vazia de T*
μ	-	Marcação de uma RdP
σ	-	Seqüência de disparo
τ	-	Duração associada à transição de uma RdP temporizada
C	-	Matriz de incidência de uma PN
Initial TSN	-	Número inicial do *chunk* de dados
Initiate_Tag	-	Inteiro aleatório de 32 bits, diferente de 0, recebido e utilizado pelo *endpoint* no **Ver_Tag** de todos seus pacotes SCTP.
MAC	-	Inteiro de 8 bits para autenticação da associação.
M(PN, μ)	–	RdP Marcada
N	-	Número natural
P	-	Conjunto finito de lugares
PN(P, T, Pré, Pós)	–	Estrutura de uma rede de Petri (RdP)
Pós	-	Função saída de transição
Pré	-	Função entrada de transição
R(PN, μ_o)	-	Conjunto de alcançabilidade de uma RdP
T	-	Conjunto finito de transições
TCB	-	Variável de estado do tipo registro
Ver_Tag	-	Inteiro de 32 bits que provê um chave para verificar se o pacote SCTP pertence a uma associação corrente.

Resumo

Martins, M. G. M. **Modelagem e Análise Formal de Algumas Funcionalidades de um Protocolo de Transporte Através das Redes de Petri**. Santa Rita do Sapucaí, 2003. Instituto Nacional de Telecomunicações – INATEL.

Esta dissertação aplica um método formal para o estudo, modelagem e análise de algumas fucionalidades do protocolo de transporte denominado SCTP. Este protocolo vem sendo desenvolvido há mais de cinco anos pelo IETF e alcançou um nível de eficiência em funcionalidade tão grande que está sendo considerado o provável candidato para substituir o TCP. É ideal para ser utilizado conjuntamente com o IPv6. Para a modelagem e análise, é empregado um formalismo matemático denominado Rede de Petri, que é uma ferramenta de modelagem e análise formal de sistemas concorrentes e paralelos, e cujo estudo e apresentação é parte integrante do trabalho. Neste trabalho, utilizam-se algumas ferramentas computacionais de auxílio à modelagem e análise de Redes de Petri, denominadas PAREDE e Design/CPN. Estas ferramentas computacionais foram escolhidas por implementarem extensões da teoria formal básica de Rede de Petri, possibilitando a modelagem de Redes de Petri com propriedades temporais e coloridas, além de implementarem métodos de análise formal das redes. Os modelos do protocolo SCTP desenvolvidos em Redes de Petri fornecem um nível de detalhes suficientes para conduzir uma análise de comportamento funcional e de certas propriedades, como ausência de bloqueios (*deadlocks*), sincronismo e seqüência correta de mensagem, além da checagem da consistência da especificação oficial do SCTP, a RFC2960.

Palavras-chave: Redes de Petri, protocolos de comunicação, modelagem e análise formal.

Abstract

Martins, M. G. M. ***Formal Modelling and Analysis of Some Functionalities of Transport Protocol through the Petri Nets***. Santa Rita do Sapucaí, 2003. Instituto Nacional de Telecomunicações – INATEL.

This dissertation presents an approaching for studying, modeling and makes some analysis about functionalities of association of SCTP transport protocol. This protocol has been developed in the last five years by IETF. This protocol achieved a high level of efficiency and it has been considered a probable substitute for TCP and ideal to work with IPv6. The modeling and analysis are developed using a mathematical formalism called Petri nets, which is a tool employed in parallel and concurrent systems and its study is an important part of this work. Some computer implementation of Petri nets formalism called PAREDE and Design/CPN are used. These were chosen because they implement extensions of the Petri net basic theory and make possible its modeling, considering time and color properties, besides implement methods of formal analysis. The SCTP protocol models developed using Petri nets, provide some essential details to perform a functional behavior analysis and specific properties, like absence of deadlocks, synchronization, and correct sequences of messages, besides the SCTP official specification checking of consistency, the RFC2960.

Keywords: Petri Nets, communication protocols, modeling and formal analysis.

Capítulo I - Introdução

1.1 - Contexto e Motivações

É comum os sistemas computacionais apresentarem mal funcionamento sistêmico, estrutural, originados por falha de projeto. Freqüentemente, nos deparamos com o problema em que o computador pessoal funcionando, por exemplo, com Windows ou Linux, principalmente em rede, entra num estado de bloqueio e a única saída é reiniciar a máquina (Desligar e Ligar). Muito mais grave é quando isso ocorre com uma Central Telefônica de milhares de assinantes, ou com um computador que controla naves espaciais ou equipamentos nucleares. O impacto dessa tecnologia em nossa sociedade é imenso e nossa dependência dela é cada vez maior. Porém, muitas vezes esses sistemas não são suficientemente confiáveis.

A carência de informações e métodos para abordagem desses problemas de forma clara é uma das dificuldades encontradas. Há deficiência de "ferramentas" adequadas para tratamento e solução dessas falhas sistêmicas. Segundo Claude e Valk [Gira98], têm-se argüido, freqüentemente, que métodos e padrões da engenharia tradicional devem ser adaptados para o necessário desenvolvimento de software. "Por essa razão, surgiu o campo da engenharia de desenvolvimento de software com o desafio de realizar esse desenvolvimento com qualidade". Algumas de suas propostas são norteadas pelo uso de métodos formais [1] nos processos de elaboração dos sistemas.

---

[1] Um método se diz formal quando o conjunto dos procedimentos e técnicas utilizadas são formais, isto é, têm um sentido matemático preciso, sobre o qual se pode raciocinar logicamente, obtendo-se completude, consistência, precisão, correção, concisão, legibilidade e reutilização das definições abstratas [Berg82].

No contexto da evolução corrente no mundo dos serviços de telecomunicações, oriundos principalmente da evolução computacional dos elementos de telecomunicações, observa-se que há a preocupação com a questão da correção dos sistemas e os investimentos em pesquisa para se evitar os problemas de mau funcionamento são intensos. Porém, ainda não se chegou a uma solução utilizável de forma geral e trivial em todos os tipos de aplicações.

Um sistema de telecomunicações permite a comunicação entre entidades remotas. Ele é composto de dois subsistemas, chamados de rede de transporte e sistema de processamento. A rede de transporte é o conjunto de recursos físicos de comunicação que permitem a transferência da informação entre as entidades da comunicação. O sistema de processamento é o conjunto de recursos de computação e programas que controlam e gerenciam a rede de transporte, ou seja, implementa o software de comunicação. Este software é projetado de acordo com regras sintáticas e semânticas chamadas **protocolos.**

O desenvolvimento destes protocolos constitui a ***Engenharia de Protocolos***, que pode ser definida como uma especialização da engenharia de software para software de comunicação. [Gira98]: "*Engenharia de protocolos abrange o conjunto de técnicas, métodos e ferramentas que permitem a produção e exploração de software de comunicação com um controle industrial. Este processo de produção é organizado de acordo com os estágios clássicos do ciclo de desenvolvimento, chamados de especificação, projeto, implementação, distribuição e manutenção"*.

No caso da engenharia de protocolos, essas atividades são fortemente dependentes da complexidade do sistema de telecomunicações. Exemplos dessa complexidade são o tamanho do sistema, a heterogeneidade do hardware e software, o aspecto de tempo real onde o sistema tem de operar e estar permanentemente disponível e a garantia de uma operação correta. Não obstante, a contínua evolução dos sistemas de telecomunicações também é um fator de complexidade por demandar ampliação das funcionalidades dos sistemas. Nesse contexto, os protocolos de comunicação estão, a cada dia, tornando-se mais complexos, proporcionalmente à

funcionalidade a que se propõem e, conseqüentemente, o controle de qualidade do projeto também se torna mais complexo e difícil.

Diante da evolução dos sistemas, ocorre a evolução natural das ferramentas de aferição desses sistemas. Antigas ferramentas e métodos são deixados de lado e outras ferramentas novas são desenvolvidas e usadas para garantir o funcionamento correto dos sistemas. Conforme [Gira98], embora algum progresso seja observável nas áreas de modelagem, análise, validação e implementação, há falta de um método poderoso para conexão de todas essas áreas ou fases de desenvolvimento e, embora técnicas de modelagem de base gráfica sejam de interesse crescente, poucas são fundamentadas em métodos formais.

Essa preocupação reflete-se nos documentos oficiais que especificam os protocolos de comunicação. Por serem basicamente textuais, portanto ambíguos, não fornecem a necessária precisão lógica das funcionalidades do sistema, tornando difícil (para a mente humana) a análise do comportamento do sistema com a precisão necessária. Conseqüentemente, também há necessidade de emprego de linguagens ou métodos mais adequados para a especificação e documentação desses sistemas.

Métodos formais têm sido usados desde que a atividade de desenvolvimento de protocolos surgiu. Na verdade, conforme [Gold00], a primeira substituição efetiva da noção intuitiva de procedimento por uma idéia formal, ou matemática, foi realizada por *Alan Turing*[2], cujo resultado foi a construção de uma conceituação matemática da noção de *algoritmo*[3], uma noção que ele modelou baseando-se nos passos que um ser humano dá quando executa um determinado cálculo ou cômputo. Turing formalizou definitivamente o conceito de algoritmo formal de um sistema.

---

[2] O trabalho de Alan Mathison Turing ficou documentado no artigo *On Computable Numbers with an aplication to the Entscheidungs problem*, publicado em 1936.

[3] A palavra algoritmo na matemática designa um procedimento geral de cálculo, que se desenvolve, por assim dizer, automaticamente, poupando-nos esforço mental durante o seu curso.

A descrição formal é necessária para expressar a funcionalidade do sistema sem ambigüidade, para analisar e validar a sua descrição a fim de detectar erros de projeto, e para desenvolver ferramentas computacionais resultantes do formalismo.

Genericamente, há dois tipos de técnicas formais, identificadas em [Gira98]:

- Os modelos representam os principais mecanismos, mas não representam totalmente uma especificação. Desse modo, uma análise exaustiva dos mecanismos representados (verificação) é praticável. As Redes de Petri (RdP) são um exemplo de tais modelos.

- As linguagens representam todos os mecanismos, mas não permitem uma exaustiva análise da especificação. Ao invés disso, simplesmente fazem a simulação ou teste de seqüências. Exemplos de linguagens são ESTELLE e LOTOS desenvolvidas na ISO[4] pelo grupo de trabalho TDF - Técnica de Descrição Formal, e a SDL - Linguagem de Descrição de Sistemas, desenvolvida no ITU-T[5]. Deve ser observado que, no contexto das telecomunicações, o termo Técnica de Descrição Formal (TDF) é comumente usado para referir-se a essas três linguagens.

Esses dois tipos não são opostos, mas cada um deles aborda um escopo específico. A modelagem permite a verificação dos mecanismos básicos da comunicação, especialmente a sincronização. Para esse propósito, a RdP tem sido amplamente usada na área de engenharia de protocolos, em função de seu poder de expressividade muito bom na especificação de paralelismo, sincronização e causalidade. Além do mais, ela permite a verificação de propriedades relacionadas ao controle e sincronização.

Do outro lado, o uso das TDFs permite a representação e teste de mecanismos orientados à implementação para os quais a modelagem não seria

---

[4] ISO (*International Standards Organizatio*n) - Organização Internacional de Padronização.
[5] ITU-T – International Telecommunications Union – Telecommunication Standardization Sector.

apropriada. Essas TDFs também são amplamente usadas no campo da engenharia de protocolos. Porém, as TDFs não possuem recursos, ou ferramentas de análises, para avaliar e validar os aspectos dinâmicos do sistema no estágio de especificação. Eles não provêem recursos de validação e verificação para avaliar a correção e submissão do modelo do sistema.

Nesse contexto de verificar e validar especificações e atestar a correção dos protocolos de comunicação, é que se dá a motivação para o desenvolvimento deste trabalho, cujo objetivo é colaborar com o desenvolvimento de pesquisa nesta linha, empregando especificamente as RdPs e mostrando como ela satisfaz muitas das necessidades desejáveis de modelagem de sistema, sua verificação e até implementação.

De [Mena01], [Gira98], [Reis98], [Jens97], e [MCPN93] infere-se que as vantagens conhecidas das redes de Petri para modelagem e análise de sistemas são:

- Elas provêem um formalismo de modelagem fundamentado gráfica e matematicamente. Isto é um contraste com outras técnicas similares, onde somente uma destas propriedades é bem desenvolvida e a outra é sistematicamente menos utilizada. Esses dois lados da moeda são de alta importância para o processo de desenvolvimento de sistemas que precisam de gráficos tanto quanto das ferramentas algorítmicas.

- Existe uma enorme quantidade de algoritmos para projeto e análises das RdPs e poderosas ferramentas computacionais têm sido desenvolvidas para auxiliar nesse processo. Para citar um exemplo, menciona-se a análise de alcançabilidade como um subcampo da checagem do modelo, o qual é utilizado nesta dissertação.

- A abstração e o projeto hierárquico são cruciais para o efetivo projeto de sistemas complexos e de grande escala. As RdPs provêem mecanismos para abstração e refinamento que são bem integrados dentro do modelo básico.

- Existe um grande número de ferramentas universitárias e comerciais para o projeto, simulação e análise de sistemas baseadas em RdPs.

- As RdPs têm sido usadas em muitas áreas de aplicação diferentes e, como resultado desse uso, existe um grande número de experiências no campo da modelagem.

- Diferentes variações dos modelos de RdPs têm sido desenvolvidas e são todas derivadas do formalismo básico desenvolvido a partir das concepções originais de K. A. Petri, idealizador da ferramenta de Rede de Petri, em 1962. Isso permite atender as necessidades dos diferentes domínios de aplicação, gerando facilidade para o intercâmbio e a transferência dos métodos e ferramentas de um campo para outro. Correntemente, fora o modelo básico, existem extensões tais como redes de Petri temporizadas, com temporização, estocásticas e orientadas a objetos, atendendo as necessidades especificas de várias áreas de aplicação.

1.2- Objetivos

O objetivo geral deste trabalho é contribuir com as pesquisas e desenvolvimento de protocolos de comunicação e com métodos e ferramentas formais para estudo e verificação desses protocolos de comunicação. O objetivo específico é o emprego das redes de Petri para o estudo formal e verificação da consistência de algumas funcionalidades do protocolo de transporte denominado SCTP (*Stream Control Transport Protocol*), através de sua modelagem, verificação e validação, realizadas com o emprego das técnicas de análise de redes de Petri disponíveis em algumas ferramentas de software utilizadas.

1.3 - Metodologia

Aplicando a afirmação de [Gira98], "*A engenharia de sistemas freqüentemente usa modelos para investigar propriedades de seus sistemas",* nesta dissertação os modelos serão baseados em Redes de Petri e serão verificadas interessantes propriedades para investigação do protocolo SCTP. Porém, os modelos têm que ser primeiramente construídos. Na verdade, a construção de modelos formais, através das redes de Petri, é de grande valia em si mesmo por exigir o completo entendimento dos aspectos do sistema em estudo.

Para efetuar a experimentação da ferramenta rede de Petri em um contexto prático, foram desenvolvidos alguns experimentos de modelagem e análise das RdPs com ferramentas computacionais disponíveis. Os experimentos foram desenvolvidos de forma a conciliar o estudo do protocolo, as funcionalidades e mecanismos que o compõe com a verificação de suas propriedades.

Destacam-se três escolas de modelagem em redes de Petri que estão em evolução. A concepção mais direta [Jens92] entende as redes de Petri como uma técnica de modelagem gráfica de uso amigável e defende que os modelos são fáceis de entender e ajustáveis ao domínio do problema. Outra visão, encontrada em [Reis98], entende as RdPs como um poderoso formalismo, mas de baixo-nível de abstração, onde os modelos são construídos em um outro formalismo e trasladados para as RdPs para análise. Ambas as concepções são válidas. Mas, em [Gira98] fala-se também que modelagem é uma arte e não existe nenhuma fórmula padrão para realizá-la. De fato, modeladores com diferentes perícias construirão diferentes modelos do mesmo problema. Cada modelo terá seus méritos e é difícil definir qual o melhor. Sob determinadas condições, alguns modelos podem até ser inúteis para certos propósitos.

Algumas sugestões encontradas em [Gira98] e [Jens92] são úteis para a modelagem. A primeira sugestão diz respeito a um equívoco comum de se fazer um modelo muito detalhado que se torna muito complexo para analisar. Nesse caso, o

modelador deve seriamente considerar se a utilização de, por exemplo, extensão de cores (recurso mais complexo de abstração) é necessária ou se um modelo de RdP convencional não seria suficiente.

Outra sugestão é que, na modelagem, primeiro e principalmente, deve-se escolher quais aspectos modelar e quais deixar de fora. Conforme [Gira98], *"um novato tem a tendência, compreensivelmente, de evitar tais escolhas e isto pode ser um sério equívoco. Deve-se aprender e ousar a escolher, e mesmo que a escolha seja errada é muito menos custoso do que não fazer escolha alguma"*. Os aspectos que beneficiam a maioria dos modelos de RdP estão relacionados com a decomposição de um sistema em vários subsistemas independentes.

1.4 - Relevância do Trabalho

Este trabalho sugere que os procedimentos de especificações, recomendações e RFC´s[6], empregados atualmente, não são suficientes para permitir a verificação de consistência dos protocolos de comunicação. Esses métodos, por exemplo, apresentam dificuldades para revelar problemas que envolvem concorrência e não garantem formalmente a consistência das especificações. Por isso é comum se encontrar falhas e erros na fase de implementação desses protocolos.

O trabalho consolida muitos conceitos e idéias relacionadas às funcinalidades dos protocolos de transporte utilizados no desenvolvimento da **rede de próxima geração**. Trata, especificamente, propriedades do protocolo de transporte SCTP desenvolvido inicialmente para a convergência das redes de sinalização telefônica e IP, cuja abordagem formal empregada no seu estudo revela características importantes desse protocolo que, de outro modo, não seriam percebidas.

---

[6] ***R**equest **F**or **C**omments* - são os documentos públicos de especificação técnica e funcional das diversas tecnologias relacionadas à Internet. É através destes documentos que se divulgam novos protocolos, permitindo uma avaliação e melhoria das idéias.

A partir deste trabalho tem-se uma nova visão, mais profunda, com relação à importância da utilização de métodos formais no estudo e desenvolvimento de protocolos de comunicação e de outros sistemas concorrentes, bem como sobre a utilização de ferramentas adequadas para o estudo, modelagem, análise e especificação de sistemas.

Durante o desenvolvimento do trabalho são apresentadas algumas limitações das ferramentas de software utilizadas e identificados alguns recursos importantes não existentes, que seriam muito úteis para o desenvolvimento de trabalhos futuros. As sugestões de métodos de modelagem aplicadas no desenvolvimento dos modelos do protocolo SCTP podem ajudar tanto a outros pesquisadores e projetistas na elaboração de seus modelos quanto a todos que desejam iniciar-se no assunto, fornecendo exemplos de emprego de metodologias sistemáticas utilizadas na área de engenharia de sistema, fundamentais para a elaboração de projetos de qualidade.

1.5- Estrutura da Dissertação

Esta dissertação é composta por mais 5 capítulos:

Capítulo II: Origens do SCTP. Todas as descrições e análises dos protocolos, realizadas no capítulo II, têm o objetivo de contextualizar os protocolos utilizados na integração das redes e explicar os motivos que levaram ao desenvolvimento do protocolo de transporte SCTP pelo grupo SIGTRAN[8], demonstrando sua necessidade diante dos requerimentos para o transporte da sinalização SS7 sobre redes IP.

Capítulo III: Especificações do Protocolo SCTP - Discorre-se sobre as especificações funcionais do SCTP que o diferenciam do TCP. Entre elas, o formato

[8] O SIGTRAN é um grupo de trabalho dentro do IETF que é dedicado ao desenvolvimento dos protocolos que viabilizam o transporte da sinalização PSTN dentro das redes IP.

do pacote SCTP e de suas mensagens de controle, o diagrama de estado da associação SCTP, a lógica de estabelecimento e terminação normal da associação, as especificações do protocolo para tratamento de eventos inesperados na associação, o gerenciamento de falhas e as especificações de terminação da associação. Essas funcionalidades serão modeladas e analisadas formalmente através das RdPs no capítulo V.

Capítulo IV: **Redes de Petri -** Esse capítulo contém os fundamentos da ferramenta RdP para o estudo formal do protocolo de comunicação. Nele são apresentados o embasamento teórico sobre as redes de Petri, englobando os fundamentos algébrico e gráfico acompanhados de sua exemplificação, as propriedades características das RdPs e alguns comentários sobre métodos formais de análise das RdPs.

Em seguida, são apresentados alguns conceitos formais para algumas extensões das RdPs que abordam parâmetros temporais. São apresentados, ainda, alguns conceitos de redes de Petri coloridas e hierárquicas (CPN)[9], de forma a esclarecer que o emprego das CPNs pode contribuir para uma bem estruturada e confiável construção de modelos de sistemas.

Por fim, tem-se uma rápida descrição das ferramentas de programação utilizadas neste trabalho: o programa PAREDE[10], que é um programa de analise de redes de Petri com temporizações; e o programa Design/CPN[11], de modelagem e análise de redes de Petri coloridas e hierárquicas.

---

[9] ***C**olored **P**etri **N**et* – Abreviação para rede de Petri colorida

[10] Programa para análise de redes de Petri com temporização. A versão do PAREDE para PC foi desenvolvida no DEE da PUC/RJ a partir da versão para "mainframe" desenvolvida pelo Prof. Dr. Miguel Menasche [Mena85]. A versão utilizada nesta dissertação é de 1995.

[11] Ferramenta software de redes de Petri colorida para projeto, especificação, simulação e verificação de sistemas, desenvolvida pela universidade de Aarhus, Dinamarca [Site3], e licenciada para uso do INATEL, conforme [Site1].

Capítulo V: Modelagem e verificação do protocolo SCTP - Basicamente, os modelos são desenvolvidos e analisados em duas etapas. Na primeira etapa são desenvolvidos modelos das fases de associação, transmissão de dados e término da associação, baseados em RdPs lugar/transição com temporização. Na segunda etapa, é elaborado um modelo baseado em RdP colorida e hierárquica (CPN) a partir do diagrama de estados alcançáveis do SCTP, presente na RFC 2960, que descreve sem muitos detalhes a associação SCTP.

A partir do método disponibilizado pelas ferramentas software utilizadas (simulação e geração do grafo de alcançabilidade), é estabelecida a relação entre as propriedades encontradas nos modelos de RdP com o sistema real. Todos os modelos desenvolvidos são analisados formalmente pelo método de geração do **Grafo de Alcançabilidade** implementado pelas ferramentas PAREDE e Design/CPN.

São apresentados os grafos de classes de estados, também conhecidos como grafos de alcançabilidade, com quantidades de estados não muito grandes (aproximadamente 250 estados). Através de análise exaustiva desses grafos verificam-se algumas propriedades do protocolo e podem ser encontradas possíveis situações de falhas no modelo ou nas especificações da RFC2960, que devem ser observadas pelos desenvolvedores do protocolo e pelos seus implementadores.

Capítulo VI: Conclusões e sugestões para trabalhos futuros -Finalmente, são apresentadas as conclusões sobre o uso de modelos baseados em RdPs para o estudo formal de protocolos de comunicação, em um contexto prático, bem como sobre as dificuldades e facilidades enfrentadas durante o desenvolvimento deste trabalho. São apresentadas também sugestões para trabalhos futuros originados a partir desta dissertação.

O Apêndice – O Apêndice está no formato digital: um CD-ROM contendo documentos e programas utilizados no desenvolvimento deste trabalho.

Capítulo II – Origens do SCTP

2.1 Introdução

Os sistemas econômicos e sociais estão intensamente atrelados ao sistema de comunicação telefônico. Apesar de existir desde 1874, quando foi inventado por Alexander Grahan Bell, sem dúvida um dos principais motivadores para a popularidade do telefone está no simplificado protocolo que especifica o modo como o usuário ainda o utiliza. De forma prática, basta alguém pegar um dos tipos de telefones existentes e discar, através de um disco giratório, teclado ou simplesmente falando o nome da pessoa com quem deseja falar e pronto: seu desejo é atendido.

Graças às novas tecnologias de microeletrônica e computação, além dos serviços básicos de comunicação, vários outros serviços foram gradativamente agregados ao sistema telefônico, tais como: a notificação de uma segunda chamada e seu atendimento; conferência, transferência ou re-encaminhamento de chamadas; correio de mensagens de voz; os serviços 800, 900; o *Local Number Portability* (*LNP*); o *call ID*; o *automatic callback*; o *Roaming; o SMS; o WAP,* etc.

Não obstante, apesar desses serviços serem de fácil utilização pelos usuários, dependem de um conjunto de tecnologias com mais de um século de evolução e que continuam sendo desenvolvidas.

Dentre as tecnologias que viabilizam o desenvolvimento dos sistemas de telecomunicações, a que é mais diretamente responsável pela implementação dos novos serviços telefônicos é a **sinalização telefônica**. Esse termo se refere às

informações transferidas dentro da rede telefônica que são usadas para estabelecer, gerenciar e terminar uma chamada telefônica. Num sentido geral, **sinalização** aplica-se a qualquer fluxo de dados relacionados com o gerenciamento de elementos internos ou bases de dados da rede telefônica. Intuitivamente, pode-se perceber que essa sinalização deve operar sob um coeso conjunto de regras bem definidas, ou seja, regida por **protocolos de comunicação**.

De modo geral, é o protocolo de sinalização número 7 - SS7 (*Signalling System # 7*) que define toda a sinalização dos serviços telefônicos atuais. Ele é o sistema de sinalização telefônico mais importante e poderoso do mundo. Criado pelo *Comité Consultatif International Télégraphique et Téléphonique* (CCITT)[12] com o propósito de prover uma rede de sinalização padrão internacional. Conforme [Coll01], o escopo do SS7 é muito amplo, uma vez que ele deve cobrir todos os aspectos da sinalização de controle para redes digitais complexas. "*O fato das especificações do SS7 totalizarem 53 recomendações do ITU-T, nas séries Q.7XX, dá uma idéia de quão complexo é este padrão*".

O sistema de sinalização SS7 tornou-se tão importante quanto a própria rede de transporte de voz, pois a operação completa da rede telefônica baseia-se nele e todos os novos serviços de transporte e processamento de dados são implementados através dele.

Paralelamente à evolução dessa poderosa rede de sinalização, também conhecida como **rede inteligente**, baseada no padrão SS7, há a evolução das redes de computadores, especialmente a Internet, baseada nos protocolos TCP/IP ([Post81]). Apesar de sua latência característica, já existem formas de superar tal inconveniente e há vantagens na utilização da Internet como meio de transporte de voz e multimídia.

---

[12] O CCITT foi renomeado em 1993 e hoje ele é parte do ITU. O organismo responsável pelos sistemas de telefonia e comunicação de dados é o ITU-T.

Apesar de alguma popularização dos sistemas de **Voz sobre IP** (VoIP) [Coll01] atuais, ainda são necessários investimentos para implementar QoS na Internet para que ela tenha desempenho ao menos igual ao da PSTN (rede pública de comutação telefônica) e seja amplamente inter-operável com esta.

Tanto o ITU-T, responsável pelo desenvolvimento do padrão SS7, quanto o IETF (Força Tarefa de Engenharia da Internet), responsável pelos padrões relacionados à Internet, estão atuando no desenvolvimento de padrões necessários para assegurar a interoperabilidade das redes e garantir qualidade de serviço, tanto no transporte de voz quanto da sinalização. O protocolo SCTP é um componente importante nesse esforço.

2.2 Arquitetura de Convergência SS7/IP

Uma arquitetura de convergência SS7/IP é constituída pela interligação dos elementos das redes, através de um elemento com características especiais, denominado de Portal (*Gateway*). A Figura 2.1 abaixo exemplifica uma arquitetura convergente de redes.

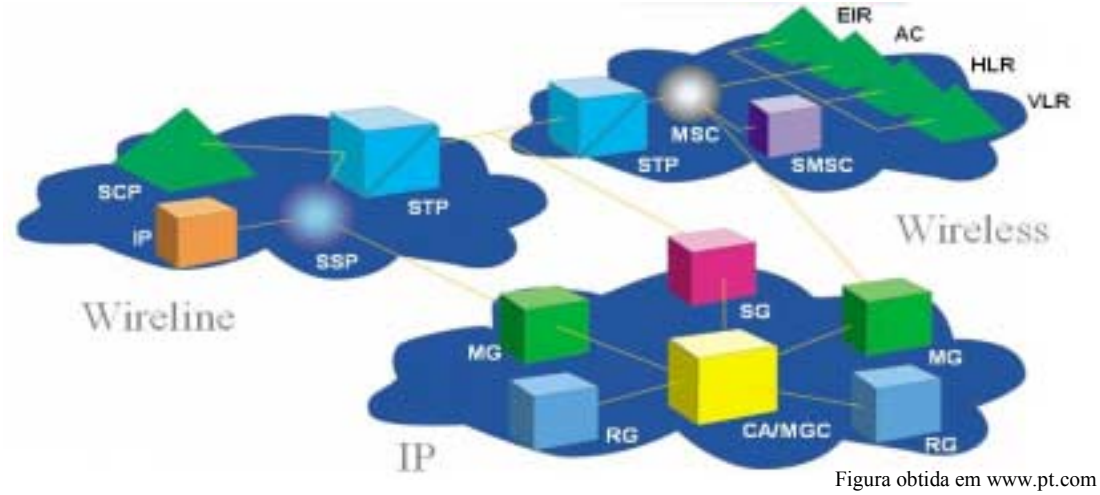


Figura obtida em www.pt.com

Figura. 2.1 – *Arquitetura de convergência das Redes SS7 e IP*

- **AC: Authentication Center**
- **EIR: Equipment Identity Register**
- **HLR: Home Location Register**
- **MSC: Mobile Switching Center**
- **SMSC: Short Message Service Center**
- **VLR: Visitor Location Register**
- **CA: Call Agent**
- **MG: Media Gateway**
- **MGC: /Media Gateway Controller**
- **RG: Residential Gateway**
- **SG: Signaling Gateway**
- **IP: Intelligent Peripheral**
- **SCP: Service Control Point**
- **SSP: Service Switching Point**
- **STP: Signal Transfer Point**

Na figura 2.1, podem ser identificados os elementos que fazem parte da arquitetura interna de cada rede particular e da interconexão das redes. Observa-se a existência de dois tipos de interconexão, a de sinalização e a de mídias (por exemplo, voz). Os STP nas redes *wireline* e *wireless* são os elementos responsáveis pela interconexão da sinalização SS7 entre elas. Na rede IP, o elemento SG faz a interface da sinalização com as redes SS7. Quanto à interconexão de mídias, a rede IP tem um elemento denominado de MG que faz as adaptações necessárias entre as interfaces SS7 e IP. No lado das redes *wireline* e *wireless,* a mídia é diretamente originária dos comutadores de circuitos chamados SSP e MSC, respectivamente.

Cada elemento (STP, SCP, SG, etc.), mostrado na rede da Figura 2.1 anterior, opera sobre um conjunto de protocolos, dependendo dos serviços e do tipo de elemento. Assim como o modelo *Open Systems Interconnection* (*OSI*)[13], os padrões SS7 e TCP/IP têm arquiteturas em camadas bem definidas, conforme a Figura 2.2.

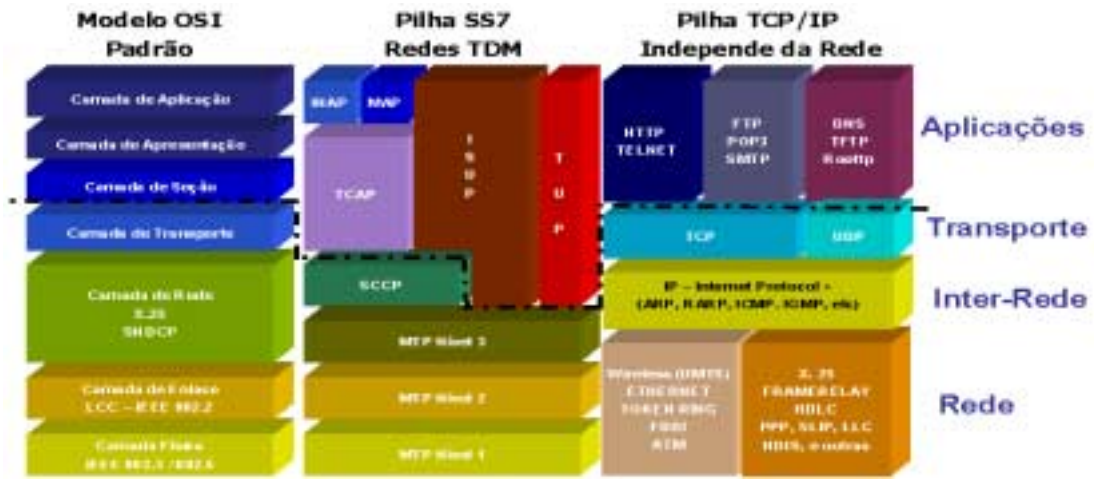


Figura. 2.2 – *Arquiteturas dos Protocolos OSI, SS7 e TCP/IP*

Na Figura 2.2, a linha pontilhada separa os protocolos de aplicação dos protocolos de rede responsáveis pelo transporte adequado das aplicações.

---

[13] O modelo OSI foi desenvolvido e publicado em 1982 pelo ***International Standards Organization*** (***ISO***). O nome vem do seu objetivo de ser um sistema de conexão aberto, isto é, sistemas que são abertos para comunicação com outros sistemas. Ele tem sete camadas, cada uma delas realiza uma tarefa bem definida e ele é freqüentemente usado para descrever os tipos de funções que um protocolo provê. Para uma boa introdução sobre o modelo OSI sugere-se a leitura da seção 1.4.1 de [Tane96].

Em termos de protocolos, de forma simplificada, pode-se considerar que a convergência se dá quando as aplicações originárias de uma rede são transportadas por outra estrutura de protocolos de transporte e processadas pelos nós desta outra rede de forma efetiva. Esse raciocínio lógico de convergência é exemplificado na Figura 2.3, na qual aplicações SS7 são postas sobre a estrutura de rede TCP/IP para serem transportadas por essa.

Figura. 2.3 – *Gateway de Sinalização SS7/IP*

A figura 2.3 exemplifica, de forma simplificada, a estrutura interna de um *Gateway* de sinalização (elemento SG da Figura 2.1), que atua como uma ponte entre a rede PSTN e a rede IP. As mensagens de sinalização padrão SS7 são recebidas da PSTN e convertidas para mensagens baseadas em TCP/IP, através do *NIF* – Função de Interconexão Nodal. As camadas superiores do SS7 podem permanecer da mesma forma como estão na PSTN (ISUP, SCCP/TCAP), mas o *Message Transfer Part* (MTP), que atua como o transportador para todas as mensagens SS7 nas redes TDM – *Time Division Multiplex* (Multiplexação por divisão de tempo), provendo transferência confiável de mensagens entre os nós SS7, não é usado na rede TCP/IP e deve ser eliminado.

Todavia, eliminar as três camadas inferiores da rede SS7 (MTP) significa perder as funcionalidades dessas camadas, uma vez que o comportamento da camada inter-rede (IP) independe da camada de enlace, e o TCP/IP não provê exatamente as mesmas funções e desempenho que a MTP, especificamente para o gerenciamento

do *enlace* (meio de comunicação entre os elementos), o gerenciamento do tráfego (Dados do usuário) e o gerenciamento de rota (caminho dos dados).

2.3 A Pilha SIGTRAN

O grupo de trabalho SIGTRAN do IETF foi fundado em 3 de novembro de 1998, com a missão de prover o transporte da sinalização PSTN baseada em pacotes sobre as redes IP. O primeiro passo do grupo foi produzir um documento informacional ([Ong99]), publicado em outubro de 1999, identificando os requisitos de funcionalidade e desempenho para suportar a sinalização sobre IP.

A forma de realizar tal tarefa foi manter ambas as pilhas SS7 e IP e definir uma interface que tornasse possível transportar ambos os *streams* de voz e os dados de sinalização SS7 através das redes IP. Esse documento, entre outras coisas, define o modelo de arquitetura de protocolos denominado SIGTRAN, que substituem (emulam) as funções das camadas mais baixas da rede de transporte SS7 (MTP). A Figura 2.4 mostra um SG com a nova estrutura de protocolos SIGTRAN.

Figura 2.4 – *Gateway SS7/ SIGTRAN*

Conforme se pode ver na arquitetura da Figura 2.4, definida pelo grupo SIGTRAN, algumas novas estruturas funcionais foram agregadas à estrutura IP e os protocolos de transporte TCP e UDP foram completamente substituídos por outro

protocolo, denominado SCTP [Stew00] (que é descrito no capítulo seguinte). De forma simplificada, os novos protocolos da Pilha SIGTRAN têm as seguintes funções:

O **M2UA** [Morn02] provê interface entre MTP3 e SCTP, tal que o padrão MTP3 pode ser utilizado na rede IP sem que o software de aplicação MTP3 saiba que as mensagens estão sendo transportadas sobre SCTP e IP em vez do MTP2. O M2UA já tem inclusive um número de porta padrão registrado de 2904.

O **M3UA** [Side02] provê uma interface entre SCTP e as aplicações que tipicamente utilizam serviços do MTP3, possibilitando comunicação ponto-a-ponto entre aplicações usuárias MTP3 sobre rede IP e aplicações idênticas na rede SS7. O M3UA tem o número de porta 2905 registrado.

O **SUA** [Loug03] provê acesso a serviços orientados a mensagem TCAP (MAP, AIN, INAP, LIDP...) e capacidade de roteamento estendida, como roteamento do número do subsistema (SSN), translação do título global (GTT) e roteamento do endereço IP.

O **IUA** [Morn01], é a especificação equivalente à Q.931 da ISDN para a sinalização de usuários, dentro da rede IP.

2.3.1 Por que um novo Protocolo de Transporte ?

Apesar dos protocolos de transporte TCP [Post81] e UDP [Post80] terem uma história de sucesso na Internet e passado, durante anos, por ajustes e estarem presentes em quase todos os sistemas operacionais, as funcionalidades esperadas do novo protocolo são:

- ✓ Transportar uma variedade de tipos de protocolos da **rede de comutação de circuitos** (***SCN***), tais como MTP3, ISUP, SCCP, TCAP, etc., com a

habilidade de prover uma forma de identificar o protocolo que está sendo transportado.

- ✓ Prover capacidade para estender a segurança e procedimentos para transporte da sinalização e suportar extensões para adicionar novos protocolos SCN, se necessário.
- ✓ Junto com o IP, prover funcionalidades relevantes como as das camadas MTP, incluindo:
    - o Controle de fluxo e Detecção de Erro.
    - o Entrega de mensagens de sinalização em seqüência dentro de *stream* (fluxos) controlados.
    - o Recuperação de falhas dos componentes no caminho da mensagem.
    - o Retransmissão e outros métodos de correção de erro.
    - o Detecção de indisponibilidade de entidades pares.
- ✓ Suportar a habilidade de multiplexar várias seções *SCN* sobre uma única sessão de transporte de sinalização. Em geral, entrega em seqüência é necessária para mensagens de sinalização dentro de um único *stream* controlado. O protocolo deve, se possível, tirar vantagem dessa propriedade para evitar bloqueio na entrega de mensagens de um *stream.*
- ✓ Ser capaz de transportar mensagens completas de tamanho maior do que os limites de segmentação e remontagem *SCN*.
- ✓ Permitir uma variação adequada de esquemas de segurança robusto para proteger as informações de sinalização sendo transportadas através das redes. O transporte da sinalização deve ser capaz de operar sobre sessões com mapeamento de endereços (*proxyable*) e através de *firewalls* (sistema de segurança que checa o conteúdo dos pacotes IP).
- ✓ Prover meios de evitar congestionamento e de reagir a congestionamentos da rede.

Diante desses requerimentos, o UDP não foi considerado, mas, inicialmente o TCP foi considerado para tornar-se o **Protocolo de Transporte Comum – *CTP***. Porém, após algumas análises detalhadas, foi mostrado que o TCP tem algumas deficiências que o tornam inadequado para o transporte da sinalização PSTN através das redes IP. Entre as deficiências identificadas em [Post81], destacam-se as seguintes:

- O TCP é um protocolo de transporte que provê tanto a transferência confiável de dados quanto a entrega de dados estritamente na ordem em que foram transmitidos. Isto é normalmente desejado, mas existem algumas aplicações que precisam de transferência confiável, mas não precisam que a seqüência seja mantida ou mesmo o ordenamento parcial dos dados é desejado. Uma aplicação com tal necessidade pode sofrer de ***Head-of-Line blocking*** (***HOL***)[14] que o TCP produz, causando um atraso que é desnecessário e indesejável.

- O TCP é orientado a *stream* (octetos, bytes) e isto também pode ser um inconveniente para algumas aplicações, visto que normalmente eles têm que incluir suas próprias marcas dentro do *stream* para identificar o início e o fim das mensagens. Além disso, eles devem fazer uso explícito da facilidade *push* (forçar o envio de dados) para assegurar que a mensagem completa seja transferida dentro de um tempo desejado.

- O TCP nunca foi projetado para ser *multihomed*[15]. O escopo limitado dos *sockets* TCP (estrutura lógica de envio e recepção de dados) torna difícil a tarefa de projetar qualquer mecanismo de transferência de dados no qual um *host multihomed* pudesse usar várias placas de rede ao mesmo tempo.

---

[14] O TCP gerencia mensagens como uma simples *string* de bytes sem estrutura interna. Assim, se usarmos uma conexão TCP simples para enviar muitas mensagens descorrelacionadas, o receptor as entregará a seu usuário na mesma ordem em que foram enviadas. Se um *datagrama* é perdido, isto afetará todas as mensagens subseqüentes, que serão retidas até a mensagem perdida chegar. Este é o bloqueio HOL.

[15] Um *host multihomed* é uma máquina (que hospeda um programa de aplicação) que tem muitas placas de rede e cuja aplicação pode fazer uso de mais de um endereço IP ao mesmo tempo.

Isto proveria alta disponibilidade, freqüentemente necessária em algumas aplicações.

- O TCP não é bem escalonável, ou seja, o número máximo de conexões TCP simultâneas é dependente das limitações do kernel (estrutura lógica central de controle do sistema operacional). Isto porque o TCP é geralmente implementado no nível de sistema operacional.

- No TCP não existe nenhuma possibilidade de controle dos tempos. O TCP geralmente não permite que as aplicações controlem sua iniciação, término e tempos de retransmissões.

- O TCP é relativamente vulnerável ao ***denial of service attacks***, que é o tipo de ataque que tenta tornar indisponíveis os serviços, normalmente tentando exaurir os recursos que eles usam. Um desses ataques, que é bem conhecido, é chamado **ataque SYN.**

O transporte da sinalização PSTN através das redes IP é um tipo de aplicação para a qual todas essas limitações do TCP são relevantes.

Após algum investimento em mudar o TCP para torná-lo capaz de atender aos requisitos para o transporte do SS7, o IETF deixou a idéia de lado. Muitas outras propostas foram levantadas. O ***Reliable UDP*** (***RUDP***) [Bova99], o ***UDP for TCAP (T/UDP)*** [MaG99], o ***Simple SCCP Tunneling Protocol (SSTP)*** [Sánc99] (uma evolução do ***Connectionless SCCP over IP Adaptation Layer (CSIP)*** [Sánd99]), o **PURDET** [Tone99], mas todas eram incompletas e foram abandonadas.

O grupo de trabalho do SIGTRAN estava não só procurando por novos protocolos, mas também levava em conta alguns protocolos do ITU-T que pudessem ser válidos, tais como o ***Service Specific Connection-Oriented Protocol (SSCOP)*** [ITUB96] ou ***H.323 Annex E*** [ITUH99] ou mesmo o ***RTP***. Nenhum deles foi considerado ideal para os propósitos do SIGTRAN.

No entanto, uma proposta submetida ao IETF, em 1998, por Randall R. Stewart e Qiaobing Xie, o ***Multi-Network Datagram Transmission Protocol (MDTP)*** [SteI00], atraiu a atenção daquele grupo de trabalho, principalmente por superar algumas deficiências do TCP, apesar de operar sobre o UDP. Esse protocolo, além de suportar operação *multihomed* e evitar o bloqueio ***HOL***, apresentava desempenho similar à do TCP.

Essas foram as razões para sua escolha pelo SIGTRAN, que, durante os dez meses seguintes, desenvolveu oito versões do MDTP, mas nunca tornou-se uma RFC. Por fim, ele foi abandonado e serviu de base para o ***SCTP – Simple Control Transport Protocol,*** que, após sua 9ª versão, foi apresentado como ***Stream Control Transport Protocol***.

A primeira versão do *Draft* Internet especificando o SCTP foi submetida em setembro de 1999 e, desde lá, muitas modificações foram feitas até outubro de 2000. Então, na 14ª versão do *Draft* Internet SCTP, ele recebeu o status de RFC e foi publicado no IETF como uma Proposta Padrão, a RFC 2960.

Durante os quatorze meses de trabalho, o projeto do protocolo SCTP foi discutido diariamente em uma lista de distribuição [Site9], que contém mais de 1.000 membros e alguns estágios do projeto, propondo mudanças e destacando erros de especificação em mais de 4.000 mensagens.

Em fevereiro de 2001, a discussão sobre o SCTP foi movida do SIGTRAN para o ***Transport Area Working Group (TSVWG)***, um outro grupo de trabalho da área de transporte do IEFT. Isso efetivamente significa que o grupo de trabalho SIGTRAN obteve muito sucesso no desenvolvimento do SCTP, a ponto de torná-lo um protocolo de transporte de propósito geral, não só para o transporte de sinalização PSTN.

Capítulo III – Especificações do Protocolo SCTP

3.1 Introdução

Assim como o TCP, o SCTP provê um serviço de transporte confiável para as aplicações sobre redes de pacotes sem-conexão, tal como a IP, assegurando que todos os dados sejam transportados sem erros. Ele realiza este serviço dentro de um contexto de uma associação entre dois *endpoints* SCTP.

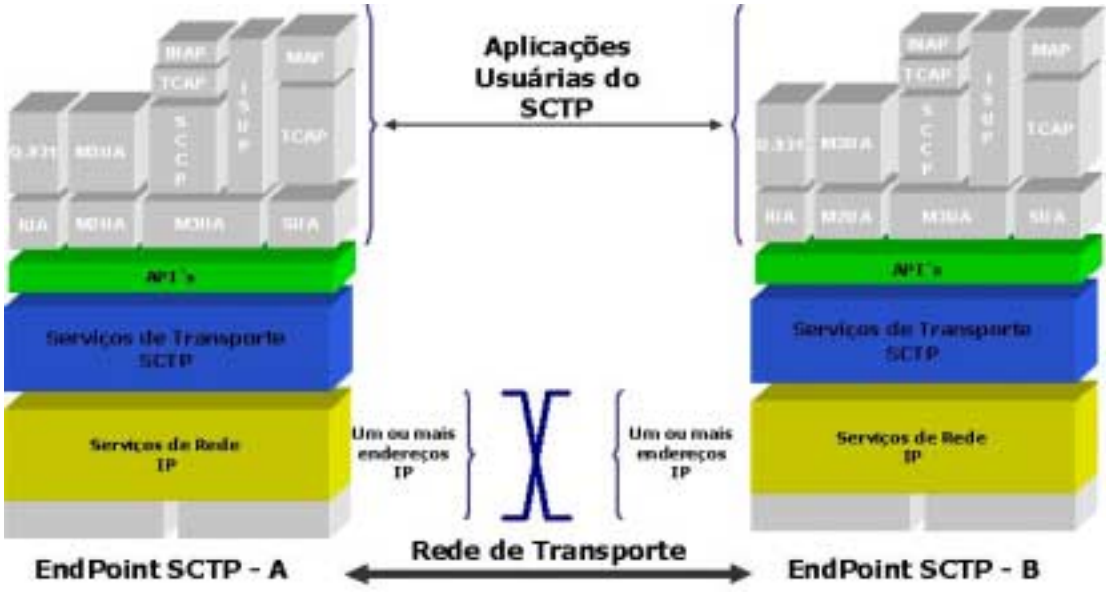


Figura 3.1 – *Associação entre endpoints SCTP*

O protocolo SCTP é orientado-a-conexão, mas o conceito de associação é mais amplo que o de conexão do TCP. O SCTP provê uma forma para que cada *endpoint* SCTP *multi-homed* informe ao *endpoint* par, durante a fase inicial da associação, uma lista de endereços de transporte (múltiplos endereços IP combinados com uma porta SCTP) através da qual o *endpoint* pode ser alcançado e pela qual ele originará pacotes SCTP.

3.2 Funcionalidades do SCTP

O serviço de transporte do SCTP pode ser decomposto em 7 blocos funcionais, conforme mostrado na Figura 3.2. Porém, nesta dissertação serão verificadas apenas 4 funcionalidades, relacionadas ao inicio e término da associação, ao empacotamento de mensagens, à validação de pacotes e ao gerenciamento de caminho.

Figura 3.2 – *Visão funcional do serviço de transporte SCTP*

Uma **associação é iniciada** por uma requisição de um usuário. Um mecanismo de *cookie*, similar ao descrito por Karn e Simpson em [RFC2522], é empregado durante a iniciação para prover proteção contra ataques de segurança. O mecanismo *cookie* usa um *four-way handshake* (Fig.3.5) e, nas últimas duas mensagens, é possível transmitir dados para uma rápida iniciação. O SCTP provê **encerramento** suave (*shutdown*) de uma associação ativa, a partir de uma requisição (primitiva SHUTDOWN) do usuário. O SCTP também permite encerramento abrupto (*abort*), ou a partir de uma requisição do usuário (primitiva ABORT) ou como resultado de uma condição de erro detectada dentro da camada SCTP. Esse protocolo não permite o *half-open* (conexão unidirecional), como o TCP, onde um lado pode continuar enviando dados enquanto o outro lado é encerrado. Quando um *endpoint* faz um *shutdown*, cada par da associação pára de aceitar novos dados de seus usuários (ULP) e somente entrega os dados que estiverem na fila.

Para o **empacotamento de *chunks*** (“fatias”, mensagens), os pacotes SCTP entregues à camada inferior são compostos de *header* (cabeçalho) comum seguido por um ou mais *chunks,* contendo dados de usuário ou informações de controle SCTP. O usuário SCTP tem a opção de requisitar o empacotamento de mais de uma mensagem em um único pacote SCTP. A função de empacotamento de *chunk* SCTP é responsável pela montagem de todo o pacote SCTP e sua desmontagem no lado de destino.

A **validação de pacotes** é feita através de um campo obrigatório chamado *Verification Tag* e um campo *checksum* de 32 bits, incluídos no *header* comum do SCTP. O valor da *Verification Tag* é escolhido por cada terminal no início da associação. Pacotes recebidos sem o valor *Verification Tag* esperado são descartados, como uma proteção contra ataques mascarados e contra pacotes SCTP antigos de uma associação anterior. O *checksum Adler-32* deve ser configurado pelo emissor a cada pacote SCTP para prover proteção adicional contra corrupção de dados na rede. O receptor do pacote SCTP com um *checksum Adler-32* inválido descarta, silenciosamente, o pacote.

Para **o gerenciamento de caminho**, o usuário SCTP emissor é capacitado para manipular o conjunto de endereços de transporte usados como destinos de pacote SCTP, através de primitivas. A função de gerenciamento de caminho escolhe o endereço de transporte para cada pacote SCTP de saída, baseado nas instruções do usuário SCTP e na situação de alcançabilidade correntemente do conjunto de endereços de destinos escolhidos. A monitoração da alcançabilidade de um endereço remoto é feita através de *heartbeats,* quando outro tráfego de pacotes é inadequado para prover esta informação e caso esse endereço não esteja alcançável o usuário SCTP é avisado. A função de gerenciamento de caminho também é responsável por informar o conjunto de endereços de transporte locais elegíveis para o terminal remoto, durante a fase inicial da associação, e reportar o endereço de transporte retornado do terminal remoto para o usuário SCTP.

Observa-se que o gerenciamento do caminho e a validação de pacotes são realizados ao mesmo tempo, apesar de descritos separadamente. Na verdade eles não podem ser realizados como ítens separados.

3.3 A Estrutura do Pacote SCTP

A estutura do pacote SCTP é muito diferente da estrutura do pacote TCP. Todo pacote SCTP tem um Cabeçalho Comum e um ou mais *chunks*. Pode-se ver na Figura 3.3 que múltiplos *chunks* podem ser empacotados em um pacote SCTP até o tamanho da MTU. Porém, isto não vale para alguns *chunks* (INIT, INIT ACK e SHUTDOWN COMPLETE).

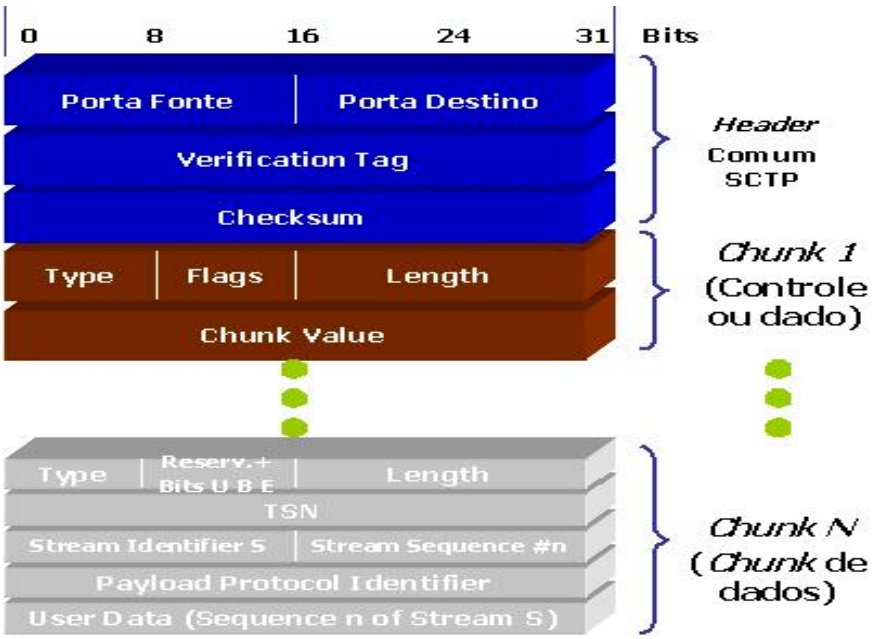


Figura 3.3 – *Formato do Pacote SCTP*

3.3.1 Os Campos do *Header* Comum

Porta Fonte (inteiro de 16 bits): Contém o número da porta do emissor. Ele pode ser usado pelo receptor em conjunto com o endereço IP fonte, a porta de destino SCTP e, possivelmente, o endereço IP de destino para identificar a associação a qual o pacote pertence.

Porta Destino (inteiro de 16 bits): Contém o número da porta para a qual o pacote é destinado. O *host* receptor usará esse número de porta para demultiplexar o pacote SCTP para o *endpoint*/aplicação receptor correto.

Verification Tag (inteiro de 32 bits): O receptor do pacote usa esse campo para validar o emissor desse pacote SCTP. Na transmissão, o valor desse campo deve ser fixado com o valor da *Initiate Tag* recebida do *endpoint* par durante a iniciação da associação, com as seguintes exceções: um pacote contendo um *chunk* INIT deve ter o campo *Verification Tag* igual a zero. O *chunk* INIT deve ser o único *chunk* transportado em um pacote SCTP; um pacote contendo um *chunk* SHUTDOWN-COMPLETE com o conjunto T-bit deve ter o *Verification Tag* copiado do pacote com *chunk* SHUTDOWN-ACK; um pacote contendo um *chunk* ABORT pode ter o *Verification Tag* copiado do pacote que causou o envio do ABORT.

Checksum (inteiro de 32 bits): Esse campo contém o *checksum* (verificação do número de bits que estão sendo transferidos para descobrir erros na transferência) do pacote SCTP. O SCTP usa o algoritmo *Adler-32* descrito no apêndice B da RFC2960.

3.3.2 Chunks SCTP (Mensagens SCTP)

Esta estrutura de *chunks* dá ao SCTP imensa capacidade de extensão e diminui o tempo de processamento do *datagrama* em função da estrutura TLV - Type-Length-Value.

Conforme a Figura 3.3, o *Chunk* inicia com um campo de identificação *Type* que é usado para distinguir *chunks* de dados ou controle, seguido por *flags* específicos (que, na prática, são ignorados) e pelo campo de tamanho, já que eles têm tamanhos diferentes.

Até o momento da elaboração deste trabalho, existiam 13 tipos de *chunks* definidos como padrão. Eles estão descritos na tabela abaixo.

Id Value	Chunk Type
0	Payload de dados (DATA)
1	Initiation (INIT)
2	Initiation Acknowledgement (INIT ACK)
3	Selective Acknowledgement (SACK)
4	Heartbeat Request (HEARTBEAT)
5	Heartbeat Acknowledgement (HEARTBEAT ACK)
6	Abort (ABORT)
7	Shutdow (SHUTDOWN)
8	Shutdow Acknowledgement (SHUTDOWN ACK))
9	Operation Error (ERROR)
10	State Cookie (COOKIE ECHO)
11	Cookie Acknowledgement (COOKIE ACK)
12	Reserved for Explicit Congestion Notification Echo (ECNE)
13	Reserved for Congestion Window Reduced (CWR)
14	Shutdown Complete (SHUTDOWN COMPLETE)
15 a 62	reservado pelo IETF
63	Definido pelo IETF para extenção de Chunks
64 a 126	reservado pelo IETF
127	Definido pelo IETF para extenção de Chunks
128 a 190	reservado pelo IETF
191	Definido pelo IETF para extenção de Chunks
192 a 254	reservado pelo IETF
255	Definido pelo IETF para extenção de Chunks

Tabela 3.1 – *Valor Id dos Chunks SCTP*

3.3.3 Descrição dos *Chunks* de Controle

A descrição dos *chunks* segue a especificação da RFC2960, porém, apenas com os campos que serão utilizados na criação dos modelos em Redes de Petri (RdPs) das especificações do protocolo SCTP.

***Chunk* Type 0 (Payload Data):** Esse *chunk* é usado para transportar dados de usuários entre dois *endpoints*. Ele possui basicamente 8 tipos de campos

funcionais, porém será utilizado apenas o ***TSN,*** que indica o número de seqüência de transmissão do *chunk* de dados.

***Chunk Type* 1 (*Initiation* - INIT):** Esse *chunk* é usado para iniciar uma associação entre dois *endpoints*. Os campos do *chunk* INIT utilizados aqui são:

- O campo ***Initiate Tag*:** O receptor do INIT armazena o valor deste parâmetro e coloca-o dentro do campo *Verification Tag* de todo pacote SCTP que o receptor do INIT transmitir dentro dessa associação. A *Initiate Tag* pode ter qualquer valor de $(2^{32} – 1)$, exceto 0. Se seu valor, recebido no INIT, for 0, o receptor deve tratar isso como um erro e encerrar a associação pela transmissão de um ABORT.

- O campo ***Initial TSN*** define o TSN inicial que o emissor usará. O valor varia de 0 a 4294967295. Este campo deve ser fixado com o valor do campo *Initiate Tag*.

- Não são utilizados campos de variáveis opcionais do INIT.

***Chunk Type* 2 (*Initiation Acknowledgement*- INIT ACK):** Esse *chunk* é usado para reconhecer a iniciação de uma associação. O formato do *chunk* INIT ACK é similar ao do *chunk* INIT, mas ele usa dois parâmetros variáveis extras:

- O ***State Cookie*** é um parâmetro cujo valor deve conter todos os estados necessários e informações de parâmetros requeridos pelo emissor desse INIT ACK para criar a associação, incluindo um código de autenticação de mensagem (MAC), um ***time stamp*** de quando o *State Cookie* é criado e o tempo de vida (***lifespan***) do *State Cookie*.

- O ***Unrecognized Parameter*** é uma cópia do parâmetro não reconhecido no INIT que é retornado ao emissor do INIT.

Chunk* Type 3 (SACK Data):** Esse *chunk* é usado para confirmar o recebimento de dados de usuários entre dois *endpoints*. Será utilizado apenas o campo ***TSN Ack, que informa o TSN recebido.

***Chunk Type* 4 (*Heartbeat Request* - HEARTBEAT):** Um *endpoint* deve enviar esse *chunk* para seu *endpoint* par para sondar a alcançabilidade de um endereço de transporte de destino particular definido na associação. O campo de parâmetro contém as informações *Heartbeat* que são entendidas somente pelo emissor. Contém também informações sobre o tempo corrente do emissor quando ele envia o *chunk* HEARTBEAT e o endereço de transporte de destino do *Heartbeat*.

***Chunk Type* 5 (*Heartbeat Acknowledgement* – HEARTBEAT ACK):** Um *endpoint* deve enviar esse *chunk* para seu *endpoint* par como uma resposta a um *chunk* HEARTBEAT para o qual esse reconhecimento está respondendo. O campo de *Heartbeat Information* deve conter o parâmetro do *Hertbeat Request* ao qual esse *Heartbeat Acknowledgement* está respondendo.

***Chunk Type* 6 (*Abort Association*- ABORT):** Esse *chunk* é enviado para o par na associação para encerrar a associação. Ele pode conter parâmetros de causa para informar ao receptor a razão do encerramento. *Chunks* de controle podem ser empacotados com o ABORT (exceto os *chunks* INIT, INIT ACK e SHUTDOWN COMPLETE), mas eles devem ser colocados antes do ABORT no pacote SCTP, ou eles serão ignorados pelo receptor. Se um *endpoint* recebe um ABORT com erro de formato ou de uma associação que não existe, ele deve silenciosamente descartá-lo. Além disso, sobre qualquer circunstância, um *endpoint* que recebe um ABORT não deve responder a esse ABORT pelo envio de outro ABORT. O *chunk* ABORT possui um bit T que é fixado para 0, se o emissor tem um TCB que ele destruiu. Se o emissor não tem um TCB, ele deve fixar o bit T em 1.

***Chunk Type* 7 (*Shutdown Association*- SHUTDOWN):** Um *endpoint* em uma associação deve usar esse *chunk* para iniciar um encerramento suave da

associação com seu par. Este *chunk* possui o parâmetro *Cumulative TSN* que informa o TSN do último *chunk* recebido em seqüência antes de qualquer lacuna.

***Chunk Type* 8 (*Shutdown Acknowledgement* – SHUTDOWN ACK):** Esse *chunk* é usado para reconhecer o *chunk* SHUTDOWN recebido no término do processo de *shutdown*.

***Chunk Type* 9 (*Operation Error* – ERROR):** Um *endpoint* envia esse *chunk* para seu *endpoint* par para notificá-lo de certas condições de erro. Ele contém uma ou mais causas de erro. Um erro de operação não é considerado fatal, mas pode ser usado com um *chunk* ABORT para relatar uma condição fatal.

***Chunk Type* 10 (*Cookie Echo* – COOKIE ECHO):** Esse *chunk* é usado somente durante a iniciação de uma associação. Enviado pelo iniciador da associação para completar o processo de iniciação, esse *chunk* deve preceder qualquer *chunk* de dados enviado dentro da associação, mas pode ser empacotado com um ou mais *chunks* de dados no mesmo pacote. O Parâmetro *Cookie* deve conter o exato *cookie* recebido no parâmetro *State Cookie* do INIT ACK.

***Chunk Type* 11 (*Cookie Acknowledgement* – COOKIE ACK):** Esse *chunk* é usado somente durante a iniciação de uma associação. Ele é usado para reconhecer o recebimento de um *chunk* COOKIE ECHO e deve preceder qualquer chunk DATA ou SACK enviado dentro de uma asociação, mas pode ser empacotado com um ou mais *chunks* DATA o SACK no mesmo pacote SCTP.

***Chunk Type* 14 (*Shutdown Complete* – SHUTDOWN COMPLETE):** Esse *chunk* deve ser usado para reconhecer o recebimento de um SHUTDOWN ACK no término do processo de *shutdown*. Ele possui o bit T que, se fixado em 1, o emissor tem um TCB que foi destruído. Se o emissor não tem TCB, T deve ser fixado em 0.

3.4 Diagrama de Estados da Associação SCTP

Os passos necessários para estabelecer e terminar a associação SCTP são representados pelo diagrama de estados na Figura 3.4, na qual podem ser identificados, claramente, 8 diferentes estados (retângulos), a progressão de um estado para outro em resposta aos vários eventos causadores (Primitivas de usuário, recepção de *chunks* de controle, ou alguns eventos de *timeout*) e as ações resultantes.

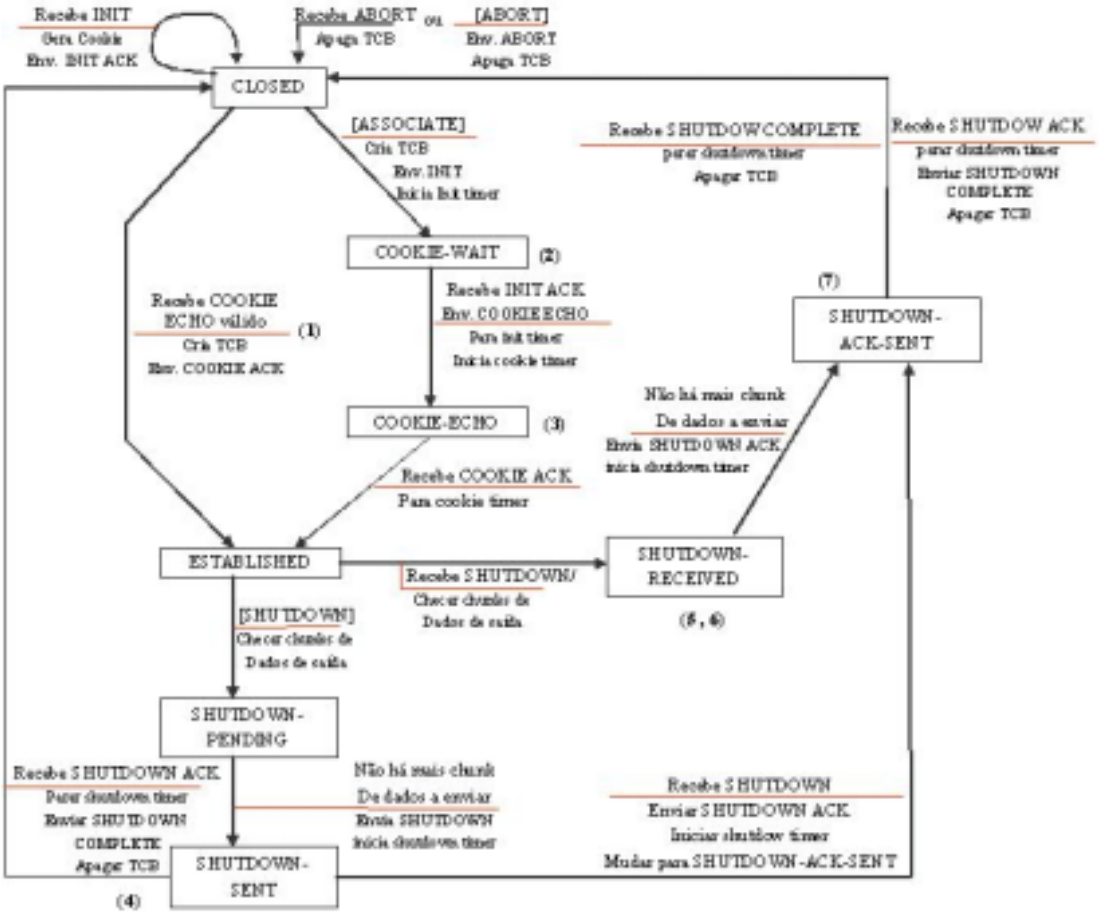


Figura 3.4 – *Diagrama de Estado da Associação SCTP*

O lado direito no diagrama de estado representa a parte ativa no estabelecimento e terminação da associação (cliente), enquanto o lado esquerdo representa o lado oposto (servidor). Os nomes dos *chunks* e das primitivas estão escritos em maiúsculo. As primitivas estão entre colchetes. O diagrama de estado não mostra as condições de erro (ERROR e ABORT) que podem ser geradas por qualquer um dos *endpoints* a qualquer instante, durante a associação.

3.5 Diagrama de Fluxo de Menssagens da Associação SCTP

Conforme se pode ver no diagrama de estados da Figura 3.4, e mais diretamente no diagrama de seqüência de mensagens da Figura 3.5, os processo de estabelecimento da associação, de troca de dados ULP e de desassociação.

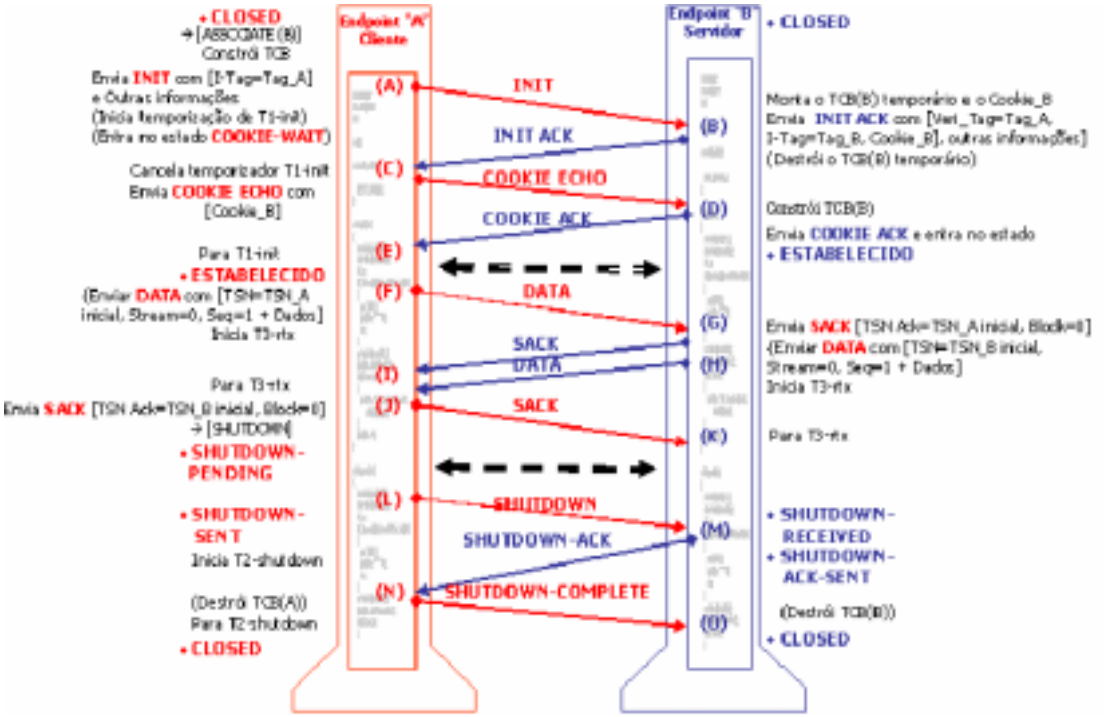


Figura 3.5 – *Diagrama Fluxo de mensagens da associação SCTP*

A fase de associação é completada após a troca de 4 mensagens (*Four-way-handshake*) e 5 passos:

A) Quando um usuário SCTP (ULP) deseja iniciar uma associação, ele solicita ao SCTP através da primitiva chamada ASSOCIATE, fornecendo os dados necessários para formar o *chunk* INIT. O *endpoint* "A" (Cliente) gera, então, um *chunk* INIT com seu *Verification Tag* (Tag_A) e o envia ao endereço de transporte (endereço IP e porta) do *endpoint* "B" (Servidor). Em seguida, "A" inicia o temporizador do INIT (*T1-init*) e vai para o estado COOKIE-WAIT (conforme visto no diagrama de estados).

B) O endpoint **"**B" (servidor), normalmente no estado CLOSED, recebe o INIT e deve responder imediatamente com um *chunk* INIT ACK. Esse INIT ACK inclui o *State Cookie*, com o código de autenticação da mensagem (MAC) e uma chave secreta (gerada pelos algoritmos MD5 ou SHA-1). Ele deve ainda configurar o campo *Verification Tag* com *Tag_A* e prover seu próprio *Verification Tag* (*Tag_B*) no campo *Initiation Tag*. O *endpoint* "B" não aloca recursos até que a terceira mensagem seja recebida e validada. Isso assegura que o pedido da associação realmente tenha sido originado pelo par correto.

C) Após receber o INIT ACK de "B", "A" deve parar o temporizador *T1-init* e deixar o estado COOKIE-WAIT. Em seguinda, deve imediatamente enviar o *State Cookie* recebido no *chunk* INIT ACK, em um *chunk* COOKIE ECHO, e iniciar o temporizador *T1-cookie*, que reenviará o *chunk* COOKIE ECHO quando *T1-cookie* se expirar e um COOKIE ACK não for recebido. Isto é repetido até o COOKIE ACK ser recebido ou o '*Max.Init.Retransmits*' ser alcançado, fazendo com que o *endpoint* par seja marcado como não alcançável (e assim a associação retorna ao estado CLOSED). Após o envio do primeiro COOKIE ECHO, a instância do protocolo entra no estado COOKIE ECHOED.

D) Após recepção de um *chunk* COOKIE ECHO, que contém a estrutura de dados do COOKIE como parâmetro, o servidor "desempacota" os dados contidos nesse COOKIE e usa o MAC contido nele para verificar se ele foi o originador do *cookie*. Se o MAC estiver OK, o *cookie* será validado (o servidor realmente o criou) e os dados contidos no *cookie* serão usados para iniciar a instância SCTP. O *endpoint* "B" responderá com um *chunk* COOKIE ACK após construir um TCB e ir para o estado ESTABELECIDO, estando apto então a enviar e receber dados.

E) Após a recepção de um COOKIE ACK do servidor, o *endpoint* "A" muda do estado COOKIE-ECHOED para o estado ESTABELECIDO, parando o temporizador *T1-cookie*. Ele deve também notificar a seu ULP o sucesso do estabelecimento da associação com uma notificação de *Communication UP*.

3.5.1 Geração do *State Cookie*

Os seguintes passos precisam ser dados para gerar o *State Cookie*: _criar um TCB para a associação, usando informações do *chunk* INIT e do INIT ACK de saída; no TCB, definir a hora da criação para a hora atual do dia e o *lifespan* para o parâmetro do protocolo '*Valid.Cookie.Life*'; do TCB, identificar e coletar o subconjunto mínimo de informações necessárias para re-criar o TCB e gerar um MAC, usando esse subconjunto de informações e uma chave secreta [RFC2104]; e gerar o *State Cookie* pela combinação desse subconjunto de informações e o MAC resultante.

Após enviar o INIT ACK com o parâmetro *State Cookie*, o enviador deve apagar o TCB e qualquer outro recurso local relacionado à nova associação, a fim de prevenir ataques a recursos. A severidade do método usado para gerar o MAC é um problema exclusivo do receptor do *chunk* INIT. O uso do MAC é obrigatório para prevenir ataques do tipo *denial of service*. A chave secreta deve ser aleatória ([RFC1750]) e mudada freqüentemente; o *timestamp* no *State Cookie* pode ser usado para determinar qual chave deve ser usada para verificar o MAC.

3.5.2 Autenticação do *State Cookie*

Quando um *endpoint* recebe um *chunk* COOKIE ECHO de um outro *endpoint* com o qual não tem associação, ele deve realizar as seguintes tarefas: computar um MAC usando os dados do TCB transportado no S*tate Cookie* e a chave secreta; autenticar o *State Cookie* como o que foi gerado anteriormente, comparando o MAC computado com o transportado no *State Cookie*. Se esta comparação falhar, o pacote SCTP, incluindo o COOKIE ECHO e todo *chunk* de dados que houver, deve ser silencionsamente descartado; comparar o instante de criação do *timestamp*, registrado no *State Cookie*, com a hora atual local. Se o tempo passado é maior que o *lifespan* transportado no *State Cookie*, então o pacote, incluindo o COOKIE ECHO e qualquer *chunk* de dados anexo, deve ser descartado e o *endpoint* deve transmitir um

chunk de ERROR com a causa de erro ”*Stale Cookie*” para o *endpoint* par; se o *State Cookie* é válido, criar uma associação com o enviador do *chunk* COOKIE ECHO, com as informações de dados do TCB transportadas no COOKIE ECHO, e entrar no estado ESTABELECIDO; enviar um *chunk* COOKIE ACK para o par reconhecendo a recepção do COOKIE ECHO. Se o COOKIE ECHO é recebido de um *endpoint* com o qual já tem uma associação, os procedimentos para tratar esse *chunk* são os descritos na seção 3.6 a seguir.

3.6 Mensagens de Iniciação Inesperadas ou Duplicadas

Conforme se pode ver nos diagramas de seqüências de mensagens da Figura 3.6, várias seqüências de mensagens são possíveis no sistema.

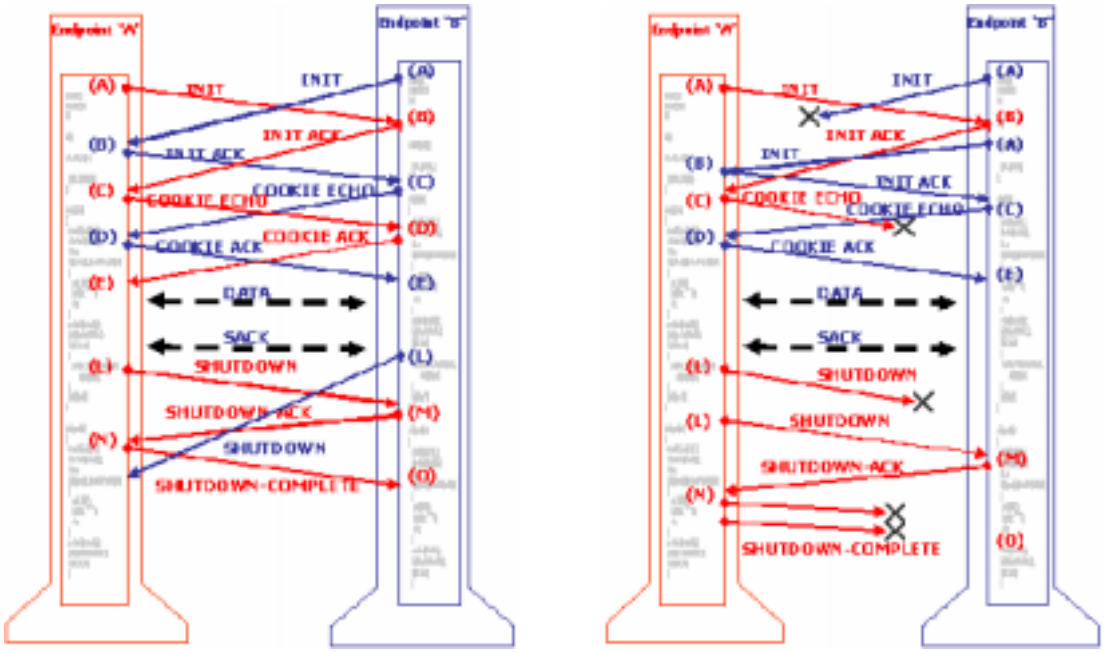


Figura 3.6 – *Diagramas de Fluxos com envio simultâneo e perdas de mensagens SCTP*

Durante o tempo de vida de uma associação, em um dos possíveis estados, um *endpoint* pode receber de seu par um dos *chunks* de controle (INIT, INIT ACK, COOKIE ECHO e COOKIE ACK). Esses *chunks* devem ser tratados como duplicados e processados conforme descrito a seguir. Nota-se que um *endpoint* não receberá o *chunk,* se esse não for enviado para um endereço de transporte SCTP

associado com esse *endpoint*. Então, o *endpoint* processa tal *chunk* como parte de sua associação corrente.

Os seguintes cenários podem causar *chunks* duplicados ou inesperados: (**A**) o par sofreu algum problema sem ser detectado, reinicia-se e envia um novo *chunk* INIT tentando restaurar a associação; (**B**) ambos os lados tentam iniciar a associação ao mesmo tempo; (**C**) o *chunk* é de um pacote antigo que foi usado para estabelecer a presente associação ou de uma associação passada não mais existente; (**D**) o *chunk* é de um pacote falso gerado por um ataque; ou, (**E**) o par nunca recebeu o COOKIE ACK e está retransmitindo seu COOKIE ECHO.

Para identificar e manusear, corretamente, esses casos, as regras seguintes devem ser aplicadas.

3.6.1 INIT Recebido no Estado COOKIE-WAIT ou COOKIE-ECHOED

Segundo [Stew00], normalmente isso indica uma colisão na iniciação, por exemplo, cada *endpoint* está tentando estabelecer uma associação, ao mesmo tempo. Após receber o INIT, no estado COOKIE-WAIT, um *endpoint* deve responder com um INIT ACK, usando os mesmos parâmetros que ele enviou em seu *chunk* INIT inicial, incluindo o mesmo *Initiation Tag*. Os parâmetros originais são combinados com aqueles do mais novo *chunk* INIT recebido. O *endpoint* deve também gerar um *State Cookie* com o INIT ACK. O *endpoint* usa os parâmetros enviados em seu INIT para calcular o *State Cookie*.

Após isso, o *endpoint* não deve mudar seu estado, o temporizador *T1-init* deve ser deixado com seu valor corrente e o correspondente TCB não deve ser destruído. Os procedimentos normais para manusear o ***State Cookie***, quando um TCB existe, resolverá os INITs duplicados para uma única associação.

Para um *endpoint* que está em um estado COOKIE-ECHOED, ele deve divulgar seu "*Tie-Tags*" com a informação ***Tag*** dele mesmo e de seu par, conforme descrito na seção seguinte.

3.6.2 INIT Inesperado em Estados diferentes de CLOSED, COOKIE-ECHOED, COOKIE-WAIT e SHUTDOWN-ACK-SENT

A menos que de outra forma declarado, após a recepção de um INIT inesperado para essa associação, o *endpoint* deve gerar um INIT ACK com um *State Cookie*. No INIT ACK, o *endpoint* deve copiar a *Verification Tag* atual e a *Verification Tag* de seu par para dentro de um lugar reservado no *state cookie*. Essas locações são chamadas de *Tie-Tag*_Par e *Tie-Tag*-Local. O pacote SCTP contendo esse INIT ACK deve transportar um valor *Verification Tag* igual a *Initiation Tag* encontrada no INIT inesperado. E o INIT ACK deve conter uma nova *Initiation Tag*. Outros parâmetros para o *endpoint* devem ser copiados dos parâmetros da associação existente dentro do INIT ACK e *cookie*.

Após o envio do INIT ACK, o *endpoint* não realizará mais nenhuma ação, ou seja, a associação existente incluindo seu estado corrente e o correspondente TCB, não serão mudados. Somente quando um TCB existe e a associação não está no estado COOKIE-WAIT, o *Tie-Tag* é divulgado. Para um INIT de uma associação normal, o *Tie-Tag* deve ser fixado em 0 (indicando que nenhum TCB prévio existe). O INIT ACK e o *State Cookie* são divulgados como foi especificado na seção 3.6.1.

3.6.3 INIT ACK Inesperado

Se um INIT ACK é recebido por um *endpoint* em qualquer estado que não seja o COOKIE-WAIT, o *endpoint* deve descartar o *chunk* INIT ACK. Um inesperado INIT ACK normalmente indica o processamento de um *chunk* antigo ou duplicado.

3.6.4 Manuseio do COOKIE ECHO quando um TCB existe

Quando um *chunk* COOKIE ECHO é recebido por um *endpoint* em qualquer estado de uma associação existente, a seguinte regra deve ser aplicada:

1) Computar um MAC como descrito no passo 1 da seção 3.5.2.

2) Autenticar o ***State Cookie*** como descrito no passo 2 da seção 3.5.2 (são os casos C ou D na seção 3.6).

3) Comparar o *timestamp* no ***State Cookie*** com a hora atual. Se o *State Cookie* é mais antigo do que o *lifespan* transportado no *State Cookie* e a *Verification Tag* contida no *State Cookie* é incompatível com a *Verification Tag* da associação atual, o pacote, incluindo o COOKIE ECHO e qualquer *chunk* de dados, deve ser descartado. O *endpoint* também deve transmitir um *chunk* de ERROR com uma causa de erro "*Stale Cookie*" para o *endpoint* par (Casos C e D).

   Se ambos ***Verification Tag*** no ***State Cookie*** combinam com a ***Verification Tag*** da associação atual, considerar o ***State Cookie*** válido (Caso E), mesmo se o ***lifespan*** estiver excedido.

4) Se o *State Cookie* provar ser válido, desempacotar o TCB dentro de um TCB temporário.

5) Consultar a tabela abaixo para determinar a ação correta a se tomar.

Local Tag	Tag do Par	Tie-Tag-Local	Tie-Tag-Par	Ação
X	X	M	M	(A)
M	X	A	A	(B)
M	O	A	A	(B)
X	M	O	O	(C)
M	M	A	A	(D)

Tabela 3.2 – *Manuseio do COOKIE ECHO quando um TCB existe*

Legenda: **X** → *Tag* incompatível com o TCB exitente
M → *Tag* compatível com o TCB existente
O → Nenhum *Tie-Tag* no *Cookie* (desconhecido)
A → Todos os casos, M, X ou O.

Ações:

A) Nesse caso, o par deve ser reiniciado. Quando o *endpoint* reconhece esse potencial '*restart'*, a sessão existente é tratada como se ela recebesse um ABORT seguido por um novo COOKIE ECHO, com as seguintes exceções:

- Qualquer *chunk* de Dados SCTP pode ser retido (opção específica da implementação).
- Uma notificação de RESTART deve ser enviada para o ULP ao invés de uma notificação "*COMMUNICATION LOST*".

Todos os parâmetros de controle de congestionamento (ex., cwnd, ssthresh) relacionados a esse par devem ser reiniciados. Após isto o *endpoint* deve entrar no estado ESTABELECIDO.

Se o *endpoint* está no estado SHUTDOW-ACK-SENT e reconhece que o par reiniciou (Ação A), ele não pode configurar uma nova associação; em vez disso reenvia o SHUTDOWN ACK e envia um chunk de ERROR com uma causa de erro "*Cookie Received while Shutting Down*" para seu par.

B) Nesse caso, ambos os lados podem estar tentando iniciar uma associação ao mesmo tempo, mas o *endpoint* par iniciou seu INIT após reponder ao INIT do *endpoint* local. Com isso ele deve ter selecionado uma nova Verification Tag, não estando ciente do *Tag* anterior que ele enviou para este *endpoint*. O *endpoint* deve manter o estado ou ir para o estado ESTABELECIDO, atualizando a *Verification Tag* do *State Cookie* do par, parando qualquer temporizador, *T-init* ou *T-cookie*, que estiver rodando, e enviar um COOKIE ACK.

C) Nesse caso, o ***cookie*** do *endpoint* local tem chegado tarde. Antes de sua chegada, o *endpoint* local enviou um INIT, recebeu um INIT-ACK e, finalmente, enviou um COOKIE ECHO com a mesma ***tag*** do par mais uma nova ***tag*** dele próprio. O ***cookie*** deve ser silencionamente descartado. O *endpoint* não deve mudar de estado e deve deixar qualquer temporizador rodando.

D) Quando ambas as ***tags*** remota e local combinam, o *endpoint* deve sempre entrar no estado ESTABELECIDO, se ele ainda não o fez. Ele deve parar qualquer temporizador, que devem estar rodando, e enviar um COOKIE ACK.

Deve-se observar que a "*Verification Tag* do Par" é a ***tag*** recebida no campo *Initiate Tag* do *chunk* INIT ou INIT ACK

3.6.5 Manuseio do COOKIE ACK Duplicado

Em qualquer outro estado diferente do COOKIE-ECHOED, um *endpoint* deve silenciosamente descartar um *chunk* COOKIE ACK recebido.

3.7 Gerenciamento do Temporizador de Retransmissão

O *endpoint* usa um temporizador de retransmissão para assegurar a entrega das mensagens na ausência de qualquer confirmação de recebimento de seu par. A duração dessa temporização é definida como RTO (*time-out* de retransmissão). Detalhes podem ser encontrados na seção 6.3 da RFC2960.

3.8 Fase de Transmissão de Dados

Após o término da fase de associação ser concluído, o SCTP inicia a fase de transmissão de dados e, de forma simplificada, podem ser enumeradas as seguintes

operações dessa fase: Os nós A e B trocam mensagens de dados; Após o recebimento de cada mensagem de dados, os pontos terminais retornam uma mensagem SACK para informar o recebimento; Dados são transmitidos até que um ponto terminal decida por encerrar a associação pelo envio de uma mensagem SHUTDOWN.

Transmissões de dados somente podem ocorrer nos estados: ESTABELECIDO, SHUTDOWN-PENDING e SHUTDOWN-RECEIVED. A única exceção é quando *chunks* de dados são empacotados juntamente com as mensagens COOKIE ECHO, quando no estado COOKIE-WAIT. As recepções de dados somente podem ocorrer nos estados: ESTABELECIDO, SHUTDOWN-PENDING e SHUTDOWN-SENT. Os SACKs podem ser processados nos estados ESTABELECIDO, SHUTDOWN- SENT e SHUTDOWN-RECEIVED. E, SACKs que chegam, no estado COOKIE-ECHOED, também.

Durante todo o processo de transmissão, HEARBEAT e HEARTBEAT ACK são trocados entre os nós, em intervalos de tempo regulares. Eles testam a conectividade entre os pontos terminais. Implementações do SCTP devem ter mecanismos de controle de congestionamento e fluxo de acordo com a RFC 2960, garantindo que o SCTP possa ser introduzido sem problemas em redes nas quais o TCP é amplamente usado. Tais mecanismos não são abordados nesta dissertação.

3.9 Gerenciamento de Falhas no *Endpoint* e no Caminho

Um *endpoint* deve manter um contador do número total de retransmissões consecutivas para seu par, incluindo retransmissões para todos os endereços de transporte de destino do par, se ele é *multihomed*. Se o valor do contador exceder o limite indicado no parâmetro do protocolo '*Association.Max.Retrans*', o *endpoint* deve considerar o *endpoint* par inalcançável e deve parar qualquer retransmissão de dados para ele, voltar para o estado CLOSED, relatar a falha para a camada superior e, opcionalmente, relatar todos os *chunks* de dados em sua fila de saída.

O contador deve ser zerado cada vez que um *chunk* de Dados enviado seja reconhecido, pela recepção de um SACK, ou um HEARTBEAT-ACK. No caso da detecção de falha no caminho, quando seu *endpoint* par é *multihomed*, um *endpoint* deve manter um contador de erro para cada endereço de transporte de destino do *endpoint* par. Cada vez que o temporizador expira, ou um HEARTBEAT enviado não é reconhecido dentro do RTO, há incremento do contador. Quando o contador exceder um limite definido no parâmetro do protocolo '*Path.Max.Retrans*', o *enpoint* deve marcar o endereço de transporte de destino como inativo e uma notificação deve ser feita à camada superior. A reiniciação do contador é feita sempre após o recebimento de uma mensagem enviada. O *endpoint* SCTP deve monitorar a alcançabilidade do(s) endereço(s) de transporte de destino livre(s) de seu par pelo envio periódico do *chunk* HEARTBEAT.

3.10 Fase de encerramento de uma Associação

O SHUTDOWN pode ser enviado por qualquer um dos *endpoints*. Basicamente, as operações são: O **nó A** envia um SHUTDOWN e inicia o temporizador. O **nó B** recebe o SHUTDOWN e envia o SHUTDOWN ACK. E o **Nó A** recebe o SHUTDOWN ACK e interrompe o temporizador. Gera então um SHUTDOWN COMPLETE que é enviado ao **nó B**.

Ambos os lados podem decidir terminar uma associação por diversas razões e o podem fazer praticamente em qualquer momento. Existe a possibilidade de um encerramento oportuno, assegurando que nenhum dado é perdido, e o encerramento (pode ser feito) de forma abrupta.

3.10.1 - Encerramento Suave

Após o recebimento da primitiva SHUTDOWN do processo do usuário da camada superior, uma instância SCTP deve parar de aceitar dados desse processo e

enviar um SHUTDOWN. Esse processo é garantido por um temporizador. O par irá então receber esse SHUTDOWN e responder com um SHUTDOWN ACK.

Quando o par que iniciou o procedimento de *shutdown* recebe o SHUTDOWN ACK, ele irá interromper o temporizador, enviar um SHUTDOWN COMPLETE, removendo todos os dados relacionados com aquela associação, e entrando no estado CLOSED. O par que recebe o SHUTDOWN COMPLETE também poderá remover todos os registros daquela associação, entrando também no estado CLOSED. Caso esse SHUTDOWN COMPLETE seja perdido, o par irá continuar enviando SHUTDOWN ACKS até que um contador de erros seja excedido, indicando que o outro par está inacessível.

3.10.2 - Abortando a Associação

Um *endpoint* pode também deliberar uma abortagem da associação. Ele tem que preencher o rótulo de verificação no pacote de saída e não deve empacotar nenhuma *chunk* de dados nesse pacote. O receptor não responde, mas valida este *chunk* de controle e remove a associação, se o ABORT contém o rótulo de verificação correto, reportando o encerramento aos processos das camadas superiores.

3.11 Conclusão

Conforme pôde ser verificado, os dois métodos formais utilizados (o grafo de estados e o diagrama de fluxo) para descrever as especificações não são suficientes para abordagem de todos os detalhes do SCTP, principalmente os eventos que envolvem concorrência. Portanto, é necessário o uso de uma ferramenta mais adequada para abordagem dos aspectos estruturais e comportamentais do SCTP de forma mais consistente.

Capítulo IV – Redes de Petri

4.1 Introdução

O grande motivador do estudo e utilização da ferramenta RdP é sua adequação à modelagem e análise de sistemas discretos à evolução paralela [MENA01]. Em comparação com os diagramas de estados, talvez este tenha sido o maior impulso que levou **Petri** a este tema em sua tese de doutorado, "Comunicação com autômatos": a insuficiência com a qual os autômatos tratavam o caso de processos concorrentes e paralelos. No diagrama de estado apresentado na Figura 4.1, a sua característica principal é a decisão em qual dos estados seguintes o sistema prossegue.

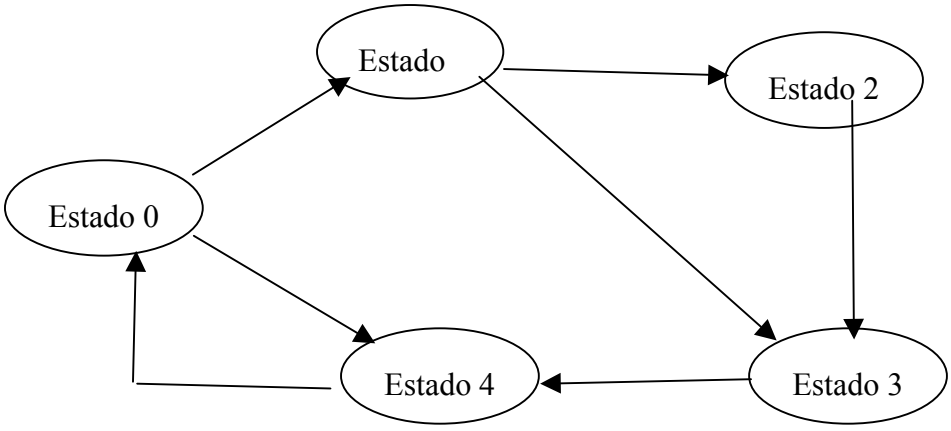


Figura 4.1 – *Diagrama de Estados.*

Como se pode verificar na Figura 4.1, a partir do "Estado 1" o sistema pode tomar a decisão de ir para o "Estado 2" ou de ir para o "Estado 3", e nunca para os dois paralelamente. O mesmo acontece com o "Estado 0" e as suas respectivas decisões do caminho a seguir.

Já a RdP é uma rede de lugares e transições interconectados, com regras que determinam quando uma transição pode ocorrer, e especificações de como sua ocorrência muda os estados dos lugares associados com eles, de forma que são, particularmente, usadas na facilitação do desenvolvimento e análise de sistemas distribuídos e complexos que operam fluxos discretos de objetos e/ou informações.

O grande atrativo é a forma na qual os aspectos dos sistemas são identificados e representados pelas RdPs, de maneira gráfica ou algébrica, tornando-se um modelo formal (análítico) de especificação e controle de fluxo de informações de sistemas discretos. A teoria da RdP permite que um sistema seja modelado, resultando numa representação matemática do sistema. A análise da RdP pode então revelar informações importantes sobre a estrutura e o comportamento dinâmico do sistema modelado. Essas informações podem então ser usadas para avaliar o sistema modelado e sugerir melhorias ou mudanças.

4.2 Definições Básicas

A RdP é um grafo bi-partido e orientado tal que os vértices da rede (nós) são identificados por círculos, ou elipses, e por barras, ou retângulos. Os círculos são identificados como lugares e os retângulos, como transições. A RdP contém um conjunto de flechas, ou setas, que são chamadas de arcos, conforme está ilustrado na Figura 4.2.

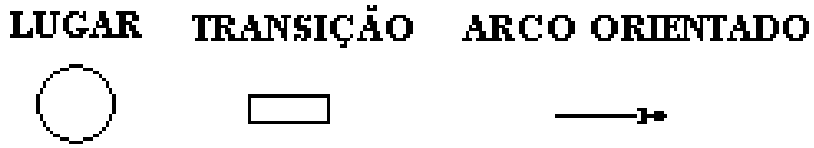


Figura 4.2 – *Elementos gráficos da rede de Petri.*

Cada arco conecta um lugar com uma transição ou uma transição com um lugar, mas nunca dois nós do mesmo tipo. Cada arco pode ter um valor inteiro positivo associado a ele. Este inteiro é chamado de uma expressão de arco. Por

convenção, omite-se aqui o termo expressão de arco e passa-se a identificar seu valor como **Peso**. Posteriormente, o termo expressão de arco volta a ser utilizado. Quando o peso do arco é igual a um normalmente pode-se omitir seu peso.

Um nó **x** é chamado um nó de entrada de outro nó **y**, se e somente se existir um arco direcionado de **x** para **y**. Analogamente, um nó **x** é chamado um nó de saída de um outro nó **y**, se e somente se existir um arco direcionado de **y** para **x**. Da mesma forma, fala-se sobre lugares de entrada, lugares de saída, transições de entrada, transições de saída, arcos de entrada e arcos de saída.

Os arcos direcionados relacionam os estados (lugares), ou sub-estados, a ações (transições) que podem ser realizadas a partir daquele estado. Relacionam também as transições aos lugares alcançáveis pela ocorrência dessas transições.

4.3 Eventos e Condições

Um evento em um sistema é uma ação ocorrendo nesse sistema. A ocorrência desses eventos é controlada pelo estado do sistema. O estado de um sistema pode ser descrito como um conjunto de condições. Uma condição é um predicado ou uma expressão lógica do estado do sistema que pode ser verdadeira ou falsa. Para que um evento possa ocorrer, é necessário que certas condições sejam verdadeiras. Essas condições são denominadas de pré-condições para os eventos. A ocorrência de um evento pode acarretar que as pré-condições deixem de ser verdadeiras e que novas condições o sejam. Essas novas condições são, por sua vez, pré-condições de outros eventos posteriores.

A modelagem de sistemas paralelos pode ser facilmente obtida através de uma rede de Petri, se as transições de uma rede de Petri forem associadas aos eventos e os lugares, a estados parciais do sistema. As pré-condições para a ocorrência dos eventos podem ser associadas a funções denominadas **Pré** e as pós-condições à funções denominadas **Pós**, que serão apresentadas posteriormente.

4.4 Formalismo matemático da Rede de Petri P/T

O formalismo das RdPs é baseado na teoria dos multi-conjuntos, ou das bolsas, que é uma extensão de teoria dos conjuntos. Um multi-conjunto é análogo a um conjunto, exceto que ele pode conter múltiplas aparências do mesmo elemento.

Uma Rede de Petri básica tem uma estrutura algébrica composta por 4 partes: um conjunto **P**, de **lugares**; um conjunto **T**, de **transições**; uma função **Pré**, entrada de transições; e uma função **Pós**, saída de transições. **Pré** e **Pós** são funções que mapeam o conjunto de transições sobre bolsas de lugares. Formalmente tem-se que uma estrutura de redes de Petri é uma quádrupla:

$$\mathbf{PN(P, T, Pré, Pós)} \qquad 4.1$$

Onde:

$$\mathbf{P} = \{\mathbf{p_1}, \mathbf{p_2}, ..., \mathbf{p_n}\} \qquad 4.1.1$$

Conjunto finito de lugares, $\mathbf{n} \neq 0$.

$$\mathbf{T} = \{\mathbf{t_1}, \mathbf{t_2}, ..., \mathbf{t_m}\} \qquad 4.1.2$$

Conjunto finito de transições, $\mathbf{m} \neq 0$. Os conjuntos **P** e **T** são disjuntos, ou seja, são entidades distintas:

$$\mathbf{P} \cap \mathbf{T} = \varnothing \qquad 4.1.3$$

Pré é uma função **entrada de transição** que mapeia transições em bolsas de lugares.

$$\mathbf{Pré : T \rightarrow P^{\infty}} \qquad 4.1.4$$

Pós é uma função **saída de transição** que mapea transições em bolsa de lugares.

$$\mathbf{Pós : T \rightarrow P^{\infty}} \qquad 4.1.5$$

$$\mathbf{|P| = n \quad e \quad |T| = m} \qquad 4.1.6$$

4.5 Grafo da Rede de Petri

Exemplificando, uma **PN(P, T, Pré, Pós)**, é tal que: **P = {p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}}** , **T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9}** e,

Pré$_{(t1)}$ = { p_1, p_6, p_6} **Pós$_{(t1)}$** = { p_2},
Pré $_{(t2)}$ = { p_2, p_7} **Pós$_{(t2)}$** = { p_3},
Pré $_{(t3)}$ = { p_3, p_7} **Pós $_{(t3)}$** = { p_4},
Pré $_{(t4)}$ = { p_4} **Pós $_{(t4)}$** = { p_1, p_6, p_6, p_7, p_7},
Pré $_{(t5)}$ = { p_5, p_6, p_8} **Pós $_{(t5)}$** = { p_9},
Pré $_{(t6)}$ = {p_6, p_9} **Pós $_{(t6)}$** = { p_{10}},
Pré $_{(t7)}$ = { p_{10}} **Pós $_{(t7)}$** = { p_5, p_{11}},
Pré $_{(t8)}$ = { p_7, p_{11}} **Pós $_{(t8)}$** = { p_{12}},
Pré $_{(t9)}$ = { p_{12}} **Pós $_{(t9)}$** = {p_6, p_6, p_7, p_8},

A descrição acima de uma rede de Petri **PN(P, T, Pré, Pós)** tem uma estrutura algébrica igual a de um grafo direcionado e bi-particionado, cujos nós são **P ∪ T** e cujas funções **Pré** e **Pós** definem os arcos direcionados entre os nós. A figura 4.3 é a representação gráfica da descrição algébrica acima.

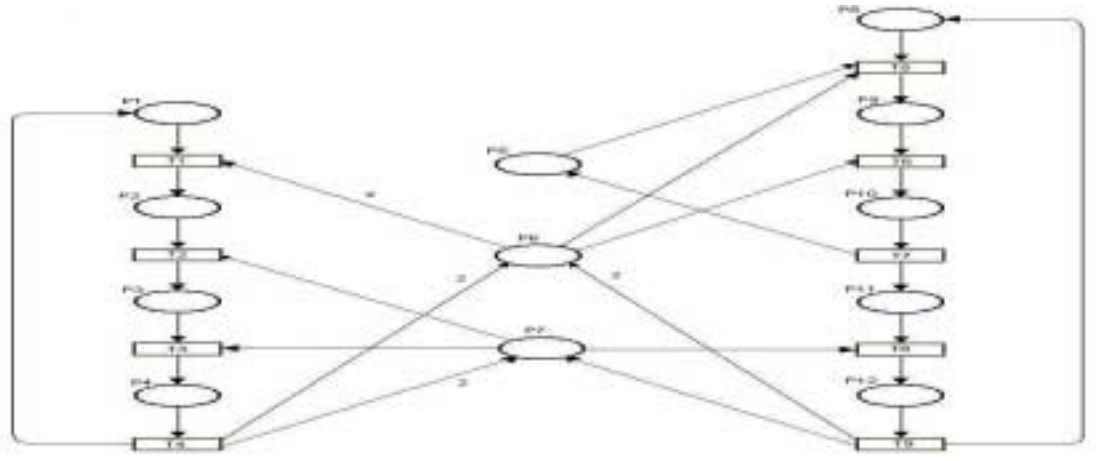


Figura 4.3 – *Rede de Petri P/T (Lugar/Transição)*

Embora não sejam necessárias para caracterizar uma rede de Petri, as funções **Pré** e **Pós** podem ser generalizadas de modo a incluir também os mapeamentos:

Pré: P → T$^{\infty}$ e Pós : P → T$^{\infty}$ 4.1.7

Estes mapeamentos são definidos de modo que:

$$\#(t_j, \textbf{Pré}\ (p_i)) = \#(p_i, \textbf{Pós}\ (t_j)) \qquad 4.1.8$$

Ler-se: o número de transições $\mathbf{t_j}$ em **Pré** de $\mathbf{p_i}$ é igual ao número de $\mathbf{p_i}$ em **Pós** de $\mathbf{t_j}$, ou,

$$\#(t_j, \textbf{Pós}\ (p_i)) = \#(p_i, \textbf{Pré}\ (t_j)) \qquad 4.1.9$$

o número de transições $\mathbf{t_j}$ em **Pós** de $\mathbf{p_i}$ é igual ao número de $\mathbf{p_i}$ em **Pós** de $\mathbf{t_j}$.

Assim, para o exemplo anterior, têm-se as funções mapeando lugares em função de bolsas de transições:

Pré $_{(p1)} = \{ t_4\}$	**Pós** $_{(p1)} = \{ t_1\}$	**Pré** $_{(p8)} = \{ t_9\}$	**Pós** $_{(p8)} = \{ t_5\}$
Pré $_{(p2)} = \{ t_1\}$	**Pós** $_{(p2)} = \{ t_2\}$	**Pré** $_{(p9)} = \{ t_5\}$	**Pós** $_{(p9)} = \{t_6\}$
Pré $_{(p3)} = \{ t_3\}$	**Pós** $_{(p3)} = \{ t_3\}$	**Pré** $_{(p10)} = \{ t_6\}$	**Pós** $_{(p10)} = \{t_7\}$
Pré $_{(p4)} = \{ t_3\}$	**Pós** $_{(p4)} = \{ t_4\}$	**Pré** $_{(p11)} = \{ t_7\}$	**Pós** $_{(p11)} = \{t_8\}$
Pré $_{(p5)} = \{ t_7\}$	**Pós** $_{(p5)} = \{ t_5\}$	**Pré** $_{(p11)} = \{ t_7\}$	**Pós** $_{(p11)} = \{t_8\}$
Pré $_{(p6)} = \{t_4, t_4, t_9, t_9\}$	**Pós** $_{(p6)} = \{ t_1, t_1, t_5, t_6\}$	**Pré** $_{(p12)} = \{ t_8\}$	**Pós** $_{(p12)} = \{t_9\}$
Pré $_{(p7)} = \{ t_4, t_4, t_9\}$	**Pós** $_{(p7)} = \{ t_2, t_3, t_8\}$		

4.6 Marcações em uma Rede de Petri

Cada lugar pode conter um número, variável dinamicamente, de pequenos pontos, que são chamados de fichas. Uma distribuição arbitrária de fichas sobre os lugares é chamada de uma marcação. Portanto, a marcação em uma RdP é uma designação de fichas aos lugares de uma RdP.

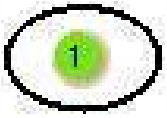


Figura 4.4 – *Lugar Marcado.*

Formalmente, uma marcação **μ** de uma rede de Petri **PN(P, T, Pré, Pós)** é uma função do conjunto de lugares **P** sobre o conjunto de inteiros não negativos **N**.

$$\mu : P \rightarrow N \qquad 4.2$$

Uma marcação pode ser representada por uma bolsa ou pelo seu vetor de **Parikh** correspondente [Mena01]. O vetor de **Parikh** descreve a quantidade de elementos em um dado multiconjunto. Exemplo: Dado um domínio **D** = {a, b, c, d, e} e o multiconjunto $\mathbf{A}_{|\mathbf{D}}$ = {a, a, a, b, b, c, e, e, e, e}, o vetor de **Parikh** de **A** é (3, 2, 1, 0, 4).

Se em uma marcação **μ**, **μ**(**p**) = **n**, para algum **p**, então se diz que **p** contém **n** fichas. Em uma representação gráfica, as fichas de uma marcação são representadas por pontos no interior dos lugares, conforme Figura 4.4.

Portanto, uma rede de Petri marcada **M(PN, μ)**, é uma estrutura de redes de Petri **PN(P, T, Pré, Pós)** e uma marcação **μ**. Observe a distribuição inicial de fichas sobre os lugares $\mathbf{p_1}$, $\mathbf{p_5}$, $\mathbf{p_6}$, $\mathbf{p_7}$ e $\mathbf{p_8}$, na figura 4.5. Essa marcação inicial do modelo e é normalmente denotada por μ_o, onde μ_0 = (2, 0, 0, 0, 1, 3, 2, 3, 0, 0, 0, 0).

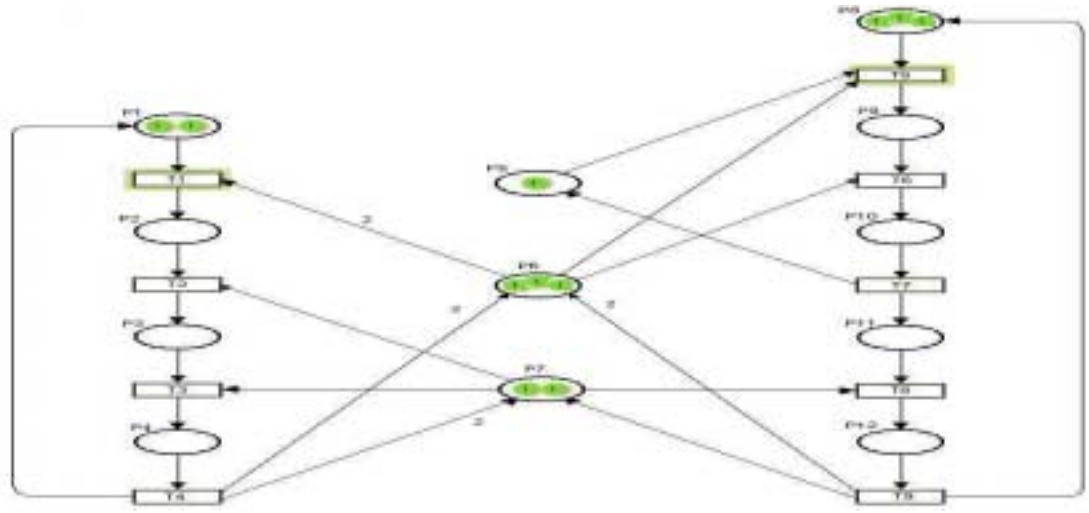


Figura 4.5 – *RdP P/T na marcação inicial* μ_0 *e transições* $\mathbf{t_1}$ *e* $\mathbf{t_5}$ *habilitadas.*

Após essa descrição sintática do modelo, deve-se agora considerar a sua semântica, ou seja, as regras de execução da rede.

4.7 Dinâmica das Redes de Petri

Conforme [Jens92], uma rede de Petri ordinária, ou lugar/transição, pode ser considerada como um jogo de tabuleiro, onde as fichas são marcas que somente podem ser posicionadas sobre lugares. Cada transição representa um potencial movimento no jogo "Jogo de rede de Petri". Um movimento é possível se e somente se cada lugar de entrada de uma transição contiver pelo menos o número de fichas preestabelecido pela expressão, ou peso, do arco de entrada correspondente. Diz-se então que a transição está habilitada.

Na marcação inicial μ_o da Figura 4.5, a transição t_1 está habilitada porque existe pelo menos uma ficha no lugar p_1 e duas fichas em p_6. Analogamente, a transição t_5 está habilitada porque existe pelo menos uma ficha nos lugares p_5, p_6 e p_8. Todas as outras transições estão desabilitadas porque há muito poucas, ou nenhuma, fichas em seus lugares de entrada. Quando uma transição está habilitada, uma mudança de estado correspondente pode acontecer. Se isso acontecer, diz-se que a transição ocorreu ou disparou. O efeito de uma ocorrência, ou disparo, é que fichas são removidas dos lugares de entrada e fichas são adicionadas aos lugares de saída. O número de fichas removidas/adicionadas é especificado pela expressão, ou peso, dos arcos de entrada/saída correspondentes. Como um exemplo, a ocorrência da transição t_1 transforma a marcação inicial μ_o (Fig. 4.5) na marcação μ_1 que é mostrada na Figura 4.6.

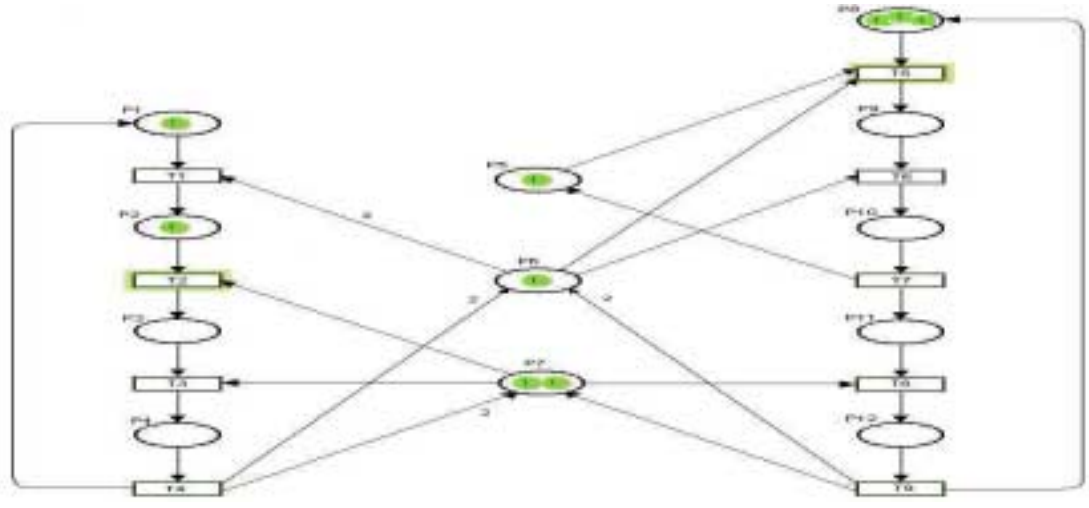


Figura 4.6 – *RdP P/T na marcação* μ_1 *e transições* t_2 *e* t_5 *habilitadas.*

Freqüentemente se pensa nessa transformação como se uma ficha fosse movida de $\mathbf{p_1}$ para $\mathbf{p_2}$, enquanto duas fichas são removidas de $\mathbf{p_6}$. Porém, deve ficar claro que o formalismo da RdP não fala sobre fichas sendo movidas de lugares de entrada para lugares de saída. Ao invés disso, fichas são removidas de lugares de entrada, e fichas completamente novas são adicionadas aos lugares de saída. Isso significa que não existe nenhum relacionamento específico entre a ficha removida de $\mathbf{p_1}$ e as fichas adicionadas em $\mathbf{p_2}$, nem entre as fichas removidas de $\mathbf{p_6}$ e as fichas adicionadas em $\mathbf{p_2}$.

Analogamente à ocorrência da transição $\mathbf{t_1}$, que transforma $\boldsymbol{\mu_o}$ (Fig.4.5) na marcação $\boldsymbol{\mu_1}$ (Fig. 4.6), a ocorrência da transição $\mathbf{t_5}$ transforma $\boldsymbol{\mu_1}$ na marcação $\boldsymbol{\mu_2}$ = (1, 1, 0, 0, 0, 0, 2, 2, 1, 0, 0, 0).

O comportamento dinâmico de um sistema é caracterizado pelo disparo de transições. Uma transição habilitada $\mathbf{t_j \in T}$ em uma RdP marcada **M(P,T, Pré,Pós, $\boldsymbol{\mu}$)** é dita habilitada ou sensibilizada se e somente se $\forall \mathbf{p} \in \mathbf{P}$:

$$\boldsymbol{\mu}(\mathbf{p}) \geq \#(\mathbf{p}, \mathbf{Pré}(\mathbf{t_j})) \qquad 4.2.1$$

Nessa condição a transição pode ser disparada. O disparo de uma transição, $\mathbf{t_j}$, em uma RdP marcada **M(P, T, Pré, Pós, $\boldsymbol{\mu}$)**, transforma a marcação corrente $\boldsymbol{\mu}$ em uma nova marcação $\boldsymbol{\mu}'$ tal que, $\forall \mathbf{p_i \in P}$:

$$\boldsymbol{\mu}'(\mathbf{p_i}) = \boldsymbol{\mu}(\mathbf{p_i}) - \#(\mathbf{p_i}, \mathbf{Pré}(\mathbf{t_j})) + \#(\mathbf{p_i}, \mathbf{Pós}(\mathbf{t_j})) \qquad 4.2.2$$

Uma vez que uma marcação define o estado de uma RdP e, em conseqüência, o estado de um sistema por esta modelado, a mudança de marcação de uma RdP corresponde, então, à mudança de estado do sistema.

4.7.1 Espaço de estados de uma Rede de Petri

Conforme [Mena01], formalmente podemos chamar de espaço de estados de uma RdP com **n** lugares ao conjunto de todas as marcações: $\mathbf{N^n}$. A mudança de estado causada pelo disparo de uma transição é definida pela função de estado seguinte, δ.

$$\delta : N^n \times T \rightarrow N^n \qquad 4.3$$

Essa função é definida apenas quando: $\mu(p_i) \geq \#(p_i, Pré(t_j))$ e é tal que se $\delta(\mu, t_j)$ é definido, seu valor é igual a μ' onde:

$$\mu'(p_i) = \mu(p_i) - \#(p_i, Pré(t_j)) + \#(p_i, Pós(t_j)), \forall p_i \in P \qquad 4.3.1$$

Dada uma RdP **PN(P, T, Pré, Pós)** e uma marcação inicial μ_o, uma execução de **PN** corresponde aos disparos sucessivos de transições habilitadas pelas sucessivas marcações alcançáveis μ_o, μ_1, μ_2, ... Esse processo pode ser feito enquanto houver marcações com transições habilitadas. Ao atingir uma marcação onde não existem transições habilitadas, a execução da rede pára. Portanto, a execução de uma RdP produz duas seqüências: Uma seqüência de marcações alcançadas (μ_o, μ_1, μ_2, ...), e uma seqüência de transições disparadas (t_{j0}, t_{j1}, t_{j2}, ...). Essas duas seqüências estão relacionadas por:

$$\delta(\mu_k, t_j) = \mu_{k+1} \quad \textbf{para } k = 0, 1, 2, \ldots \qquad 4.3.2$$

Dada uma marcação μ_o e uma seqüência de transições disparadas, é possível obter univocamente a seqüência de marcações alcançadas correspondente. Segundo [Mena01] "*o inverso também, quase sempre, é possível de obter, exceto em alguns casos degenerados*".

O conjunto de alcançabilidade **R(PN, μ_o),** onde **PN(P,T, Pré, Pós)** é uma RdP e μ_o é sua marcação inicial, é o conjunto de marcações tais que:

$\mu_o \in$ R(PN, μ_o)

Se **$\mu' \in$ R(PN, μ_o)** e $\exists$ **$t_j \in T$,** tal que **$\delta(\mu', t_j) = \mu''$,** então: 4.3.3

$\mu'' \in$ R(PN, μ_o)

Do exemplo 1, Figura 4.5, aplicando-se a expressão 4.3.3, tem-se que **R(PN, μ_o)** = {μ_o, μ_1, μ_2, μ_3, ..., μ_{12}}, onde:

μ_o = (2, 0, 0, 0, 1, 3, 2, 3, 0, 0, 0, 0),

μ_1 = (1, 1, 0, 0, 1, 1, 2, 3, 0, 0, 0, 0),

μ_2 = (1, 1, 0, 0, 0, 1, 2, 2, 1, 0, 0, 0),

μ_3 = (1, 0, 1, 0, 0, 1, 1, 2, 1, 0, 0, 0),

... **...**

μ_{12} = (2, 0, 0, 0, 1, 1, 1, 2, 0, 0, 0, 1)

Dada a RdP abaixo:

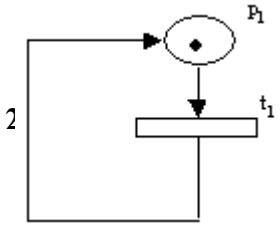


Figura 4.7 - *RdP Não-Limitada*

R(PN, μ_o) = {μ_o, μ_1, μ_2, μ_3, ...}, onde:

μ_o = (1)

μ_1 = (2)

μ_2 = (3)

μ_3 = (4)

...

Observa-se que, dependendo da rede de Petri, o conjunto de alcançabilidade pode ser finito, como é o caso da RdP do exemplo da Fig.4.5, ou infinito, como é o caso da RdP da Figura 4.7. Porém, sistemas não limitados não têm existência no mundo real.

Esses conceitos e definições constituem uma técnica formal capaz de representar de forma algébrica os elementos estruturais e comportamentais de qualquer sistema, e possibilitam a aplicação de métodos formais autômatos para a análise das propriedades e verificação da consistência do sistema especificado.

4.8 Propriedades das Redes de Petri

As propriedades de uma rede de Petri revelam o comportamento do sistema por ela modelado, permitindo assim que se chequem as propriedades do sistema real. Segundo [Gira98], elas são classificadas em dois tipos:

a) As relativas a RdP marcada, ou propriedades dinâmicas, (chamadas também de boas propriedades): são as que dependem da marcação inicial e estão ligadas à evolução da rede. Sua verificação se faz geralmente pela construção do grafo de alcançabilidade, ou máquina de estados, que será apresentado posteriormente.

b) As relativas a RdP não marcada, conhecidas como propriedades estruturais: são aquelas que dependem da estrutura topológica da RdP. Elas são independentes da marcação inicial μ_o. Assim, sua análise está baseada na teoria da álgebra linear e essas propriedades podem ser caracterizadas em termos da matriz de incidência "**C**".

4.8.1 As Boas Propriedades e as Propriedades Específicas

As boas propriedades das redes de Petri possibilitam a análise do sistema modelado. As utilizadas neste trabalho são:

a) Limitação: Uma RdP marcada **M(P, T, Pré,Pós, μ**) é dita ser **k limitada**, ou simplesmente limitada, se o número de fichas em cada lugar não excede um número **k**, inteiro e positivo, para qualquer marcação alcançável desde μ_o. Se **k** = 1 diz-se que a RdP é segura, ou binária. Ou seja, **PN(P, T, I, O,** μ_0**)** é limitada se e somente se, $\forall$ **p** $\in$ **P** e $\forall$ $\mu_0 \in$ **R(PN,** μ_o**)** existir **k** $\in$ **N** tal que μ**(p)** $\leq$ **k**.

b) Vivacidade: O conceito de vivacidade está relacionado com o possível disparo das transições. Uma transição **t** é viva em **μ,** se for possível sempre encontrar uma marcação **μ` ∈ R(PN, μ)** que a habilite. Se todas as transições da RdP forem sempre vivas, diz-se que a RdP é viva.

c) Reiniciação. ***Definição:*** Uma RdP marcada **M(P,T,Pré,Pós,μ_0)** é reiniciável para toda marcação se e somente se **∀ μ ∈ R(PN, μ_0), μ_o ∈ R(PN, μ),** ou seja, "*se o grafo de marcações alcançáveis for fortemente conexo"* [Mena01]. Assim, conclui-se que a reiniciação depende da marcação inicial e da estrutura da rede.

Dentre as propriedades específicas interessantes para análise de protocolo de comunicação, estão o paralelismo ou concorrência, o conflito (escolha ou decisão), a natureza assíncrona e a possibilidade de hierarquia.

O **Paralelismo** ou concorrência pode ser facilmente expressado em termos de redes de Petri. O **Conflito**, numa RdP, é uma situação que pode levar a uma disputa não determinística de fichas (recursos). A **Natureza Assíncrona**, onde não existe medida inerente ao tempo, por exemplo, quando mais de uma transição está habilitada ao mesmo tempo, então qualquer uma delas pode ser a próxima a disparar. Portanto, o disparo de uma transição é feito de uma maneira não determinística. Dada a natureza assíncrona do modelo de RdP clássica, a única propriedade importante do tempo está em definir uma ordem parcial da ocorrência das transições. E a **Possibilidade de Hierarquia,** em que numa rede de Petri um lugar ou uma transição pode ser substituído por uma sub-rede, o que leva a uma abstração maior, ou uma sub-rede pode ser substituída por um lugar ou uma transição, permitindo assim refinar a rede [Jens92].

4.9 Métodos de Análise das Redes de Petri

Uma vez apresentadas algumas das principais propriedades que podem ser encontradas em uma RdP, faz-se necessário determinar se uma RdP possui ou não tais propriedades. Para isso algumas metodologias de análise (verificação das propriedades) foram desenvolvidas e são implementadas em ferramentas computacionais, como as utilizadas nesta dissertação. Genericamente, as metodologias podem ser classificadas em três grandes grupos:

- **Análises por Enumeração das Marcações** (feita através da árvore de alcançabilidade/cobertura): é dinâmica, simula a execução da RdP a partir da marcação inicial e encontra todas as marcações possíveis. É o método mais utilizado para verificação das propriedades da RdP. Do exemplo da Figura 4.5, aplicando-se o método acima, obtém-se o grafo de alcançabilidade mostrado na Figura 4.8.

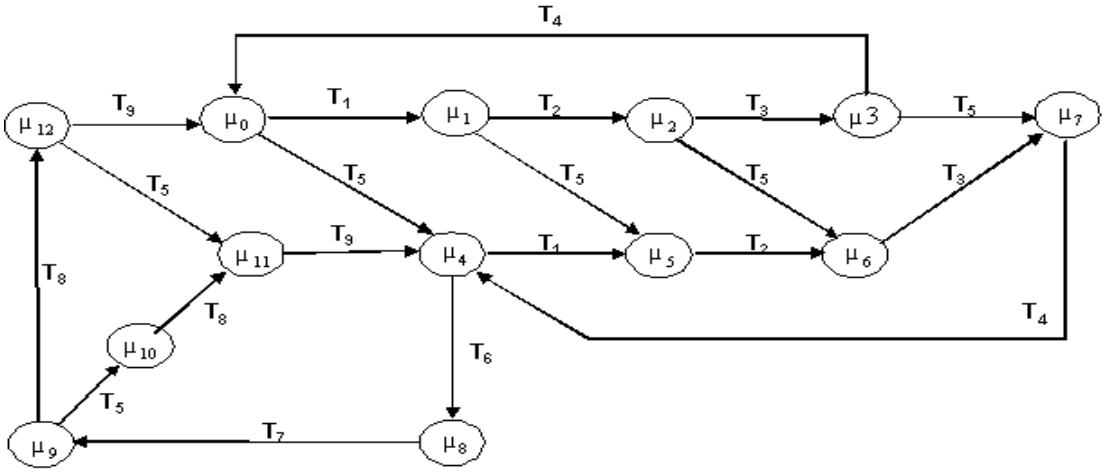


Figura 4.8 – *Grafo de alcançabilidade da RdP da Figura 4.6.*

- **Análise Estrutural**: (baseada na equação fundamental das RdP), é estática, isto é, depende apenas da topologia da rede.

- **Análise por redução**: procura a eliminação de lugares e/ou transições que não afetam as propriedades das RdPs e, posteriormente, aplica uma das técnicas anteriores.

4.10 Parâmetros Temporais nas Redes de Petri

Os parâmetros temporais em sistemas concorrentes e sistemas distribuídos têm uma fundamental implicação em seus funcionamentos e isso tem motivado o desenvolvimento de extensões nas muitas linguagens de modelagem e métodos de análises para verificação e validação desses sistemas. Para a modelagem de sistemas que apresentam parâmetros temporais explícitos em sua especificação, os modelos de redes de Petri convencionais são inadequados. Por isso, algumas extensões do formalismo básico de RdP foram desenvolvidas.

4.10.1 Redes de Petri Temporizadas (Timed Petri Nets)

Um modelo proposto por Ramchandani [Ramc74] define uma RdP temporizada como um par **(PN, τ)**, onde **PN** é uma RdP convencional e **τ** é uma função que associa um numero real, não negativo τ_i, a cada transição $t_i \in T$, denominado duração de disparo da transição t_i.

O disparo de uma transição habilitada **t** provoca uma mudança de marcação em dois passos. O primeiro é instantâneo com o disparo de **t** e provoca um decréscimo na marcação, como é mostrado abaixo:

$$\forall p \in P : \mu'(p) = \mu(p) - \#(p, Pré(t)) \qquad 4.4$$

Onde, **μ′(p)** é a marcação resultante do lugar **p** após a conclusão da primeira fase.

O segundo passo ocorre **τ**(t) unidades de tempo depois e provoca incremento na marcação como mostrado abaixo:

$$\forall p \in P : \mu''(p) = \mu'(p) + \#(p, Pós(t)) \qquad 4.4.1$$

Onde, **μ"(p)** é a marcação final do lugar **p** após a segunda fase.

A marcação final após o disparo de uma transição **t** é a mesma que seria obtida na rede de Petri convencional associada sem os parâmetros temporais.

4.10.2 Redes de Petri com Temporização (Time Petri Nets)

Conforme [Mena01], a RdP com Temporização, definida por MERLIN [Merl74], é um par **(PN, θ)**, onde **PN** é uma RdP convencional e **θ** é uma função associando um intervalo fechado sobre os reais não negativos [$\mathbf{a_i}$, $\mathbf{b_i}$] à cada transição $\mathbf{t_i} \in \mathbf{T}$, denominado intervalo de disparo estático da transição $\mathbf{t_i}$. Não existem restrições aos limites inferiores e superiores desses intervalos, exceto naturalmente que $\mathbf{a_i} \leq \mathbf{b_i}$. Isso significa que $\mathbf{a_i}$ pode ser zero e $\mathbf{b_i}$ pode ser infinito. Denomina-se $\mathbf{a_i}$ e $\mathbf{b_i}$ respectivamente de **Limite Estático Inferior (LEI) e Limite Estático Superior** (**LES**).

Nessa técnica, as transições devem ocorrer em um tempo **θ(t)** situado entre os limites de tempo definidos pelo seu intervalo de disparo estático, contado a partir do instante em que **t** foi habilitada.

Tanto as redes de Petri temporizadas quanto as redes com Temporização permitem a modelagem de sistemas em que é necessário especificar a duração de eventos, a realização de tarefas que devem ocorrer em tempos pré-determinados.

Deve-se salientar que, embora tais modelos tenham aumentado o poder de modelagem em relação a uma rede de Petri ordinária, convencional, seu poder de análise ficou reduzido. Por exemplo, não é mais possível encontrar um algoritmo para produzir alguma coisa equivalente à árvore de alcançabilidade, nem é mais possível decidir, para uma rede de Petri dotada de parâmetros temporais, se ela é limitada ou não.

4.11 As Redes de Petri de alto-nível

Conforme foi visto nas descrições das redes de Petri ordinárias (P/T) anteriores, seu uso traz importantes vantagens no controle de sistemas concorrentes, porém não permite a descrição do fluxo de informações dos sistemas do mundo real.

Isso claramente demonstra uma necessidade de tipos de redes de Petri mais poderosas para descrever sistemas complexos de uma forma gerenciável.

Conforme [Jens92], a passagem da RdP de baixo-nível para a RdP de alto-nível pode ser comparado à passagem da linguagem *assembly* para as modernas linguagens de programação. A principal razão apontada para o grande sucesso desses tipos de modelos de redes está no fato de que elas têm **uma representação gráfica** e uma **semântica bem definida** permitindo análises formais. As RdPs Coloridas (CPN) pertencem a essa classe de redes de alto-nível. Uma seleção de artigos de referência, descrevendo aplicações industriais das RdP, é encontrada em [Site2].

4.11.1 Redes de Petri coloridas (CPN)

Basicamente, a vantagem das CPN´s está no fato de que as fichas são estruturas complexas de dados que podem ser exploradas pelas transições. O valor do dado pode ser de um tipo arbitrariamente complexo (ex., um registro onde o primeiro campo é um real, o segundo uma *string* texto, enquanto o terceiro é uma lista de pares inteiros).

Para um dado lugar, todas as fichas devem ser fichas coloridas que pertençam a um tipo especifico. Esse tipo é chamado de conjunto de cor de um lugar. O uso dos conjuntos de cores na CPN é totalmente análogo ao uso de tipos de dados nas linguagens de programação. Conjuntos de cores determinam os valores possíveis de fichas (analogamente a forma na qual "Tipos" determinam os valores possíveis de variáveis e expressões).

Nas redes convencionais, indica-se a marcação corrente pela colocação de pequenos pontos dentro dos lugares, individualmente. Porém, isto não é pratico para CPNs, porque além do número de fichas tem-se que representar as cores das fichas, que podem ser valores arbitrariamente complexos. Por isso deve-se, ao invés de representar a marcação corrente de um dado lugar por meio de um pequeno círculo,

representá-lo com um inteiro, explicitando quantas fichas existem, e um texto próximo ao circulo, com um multi-conjunto informando quais são as cores das fichas individuais e quais coeficientes eles têm.

Conceber marcações mais complexas também significa que os movimentos **no "jogo rede de Petri"** tornam-se mais complexos. As fichas coloridas podem ser inspecionadas pelas transições, o que significa que a habilitação de uma transição pode depender das cores de suas fichas de entrada. Isso também significa que as cores das fichas de entrada podem determinar as cores das fichas de saída, produzindo, por exemplo, o efeito da transição. Para descrever essa situação mais complexa, é preciso elaborar expressões de arcos que especifiquem uma coleção de fichas, cada uma com uma cor bem definida, a serem adicionadas/removidas. Para fazer isto usam-se expressões de arco sobre multi-conjuntos.

A descrição coerente dos aspectos teóricos e práticos das redes de Petri coloridas é encontrada em [Jens92] e [Jens97], onde é apresentado o desenvolvimento da CPN, desde os conceitos básicos até seu emprego profissional, com estrutura de linguagem para desenvolvimento, especificação, simulação, validação e implementação de sistemas de considerável nível de complexidade.

4.11.2 CPN Hierárquica

Conforme [Jens97] as RdPs podem ser inter-relacionadas, usando construções hierárquicas baseadas na construção de lugares de substituição e transições de substituição, lugares de fusão e transições de fusão.

A idéia básica por trás das CPNs hierárquicas, com as transições e lugares de substituição, é permitir ao modelador substituir um lugar ou transição e os arcos conectados a este para construir um grande modelo através da combinação de pequenas redes CP, resultando em uma grande rede. Isto é similar à situação de um programador que constrói várias sub-rotinas até chegar no programa final.

A sintaxe e semânticas das CPN hierárquicas têm definições formais similar às definições da CPN não hierárquicas [Jens92]. Cada CPN hierárquica tem uma CPN não hierárquica equivalente, e vice versa. Os dois tipos de redes têm o mesmo poder computacional, mas as CPN hierárquicas têm maior poder de modelagem.

Conforme [Jens92], para criar grandes sistemas é necessário desenvolver estruturas fortes e conceitos de abstração. O primeiro passo é substituir redes de Petri por redes de alto nível (coloridas). O segundo passo é introduzir redes hierárquicas. Comparativamente à programação, o primeiro equivale à declaração de tipos e o segundo, à criação de sub-rotinas. Esses conceitos são implementados pela ferramenta Design/CPN e aplicados na fase de modelagem do protocolo, nesta dissertação.

4.12 Ferramentas Computacionais para Redes de Petri

Encontram-se em [Site6] muitas ferramentas que implementam redes de Petri. Após verificação dos recursos que algumas delas oferecem, optou-se por utilizar duas que oferecem recursos de edição e análise de redes temporizadas (o PAREDE) e redes coloridas e hierárquicas (o Design/CPN).

O PAREDE é um programa de análise de redes de Petri com temporizações. A versão para PC deste programa foi desenvolvida no Departamento de Engenharia Elétrica da PUC/RJ, a partir da versão para "mainframe" desenvolvido por [Mena85]. A versão utilizada neste trabalho data de 1995 e está disponível no CD-ROM do apêndice, onde se encontra também um artigo sobre o PAREDE, publicado num workshop internacional sobre RdP temporizadas, realizado pelo IEEE, no qual o seu autor destaca a utilização da ferramenta na análise de redes de Petri com temporização.

Importa destacar que poucas ferramentas implementam análise formal de RdPs com parâmetros temporais, devido à impossibilidade de se obter um algoritmo

que gere o grafo de classes de estados. No entanto, o PAREDE gera o grafo de alcançabilidade a partir de um procedimento com condição de parada. Assim, o PAREDE permite visualizar sucessivamente e em seqüência as diversas classes de estados alcançadas através de uma árvore de alcançabilidade. No caso de redes de Petri limitadas, ele resolve os problemas de vivacidade, segurança, reiniciação, alcançabilidade, persistência e o problema da cobertura. Mesmo para redes não limitadas, o PAREDE pode pesquisar se determinada marcação pode ser alcançada ou coberta por uma outra marcação alcançada durante a fase de enumeração das classes de estados. Detecta a existência de bloqueios e, se existirem, qual é a seqüência de disparos responsável pelos mesmos. Trata de redes de Petri com temporizações que possuam simetria circular e trata de redes de Petri convencionais (P/T) de maneira imediata.

Para o desenvolvimento do modelo **CPN** do protocolo **SCTP,** é empregado o programa **Design/CPN 4.0**, uma ferramenta desenvolvida pela **Universidade de *Aarhus***, Dinamarca, em 1996, que facilita a construção e análise dos modelos CPNs pelo provimento de uma representação gráfica da rede e pela análise de alcançabilidade automatizada. O espaço de estado de um modelo CPN é denominado de grafo de ocorrência (**OG**).

A versão utilizada (4.0) foi obtida junto ao grupo CPN da Universidade de Aarhus, Dinamarca, com licença acadêmica cedida ao INATEL, para uso do autor desta dissertação, conforme [Site1]. O Design/CPN provê: um editor para criação e manipulação de CPNs; um verificador de sintaxe para validação das CPNs; um simulador para execução das CPNs; monitoração interativa dos processos e capacidade de debugamento; facilidades para organizar uma rede dentro de uma hierarquia de módulos e facilidades de animação e formatação para mostrar resultados da simulação.

O tutorial do Design/CPN pode ser encontrado no CD-ROM do apêndice, bem como o arquivo de instalação do programa, na pasta /FERRAMENTAS.

Capítulo V - Modelagem e Verificação de funcionalidades do Protocolo SCTP

5.1 Introdução

Por ser o SCTP um protocolo padrão Internet, publicado no documento de Requisição para Comentários, RFC 2960, observa-se que sua especificação, descrita por diagramas de seqüências de mensagens e um diagrama de estados alcançáveis, é insuficientemente formal para garantir a consistência do protocolo. Além disso, constata-se que várias investigações sobre o SCTP concentram-se em objetivos relacionados à desempenho e os problemas funcionais são deixados para serem descobertos na implementação [Site6].

Nesse capítulo, apresenta-se um trabalho de modelagem e verificação formal, empregando duas classes de RdPs consideradas adequadas para abstração das especificações e dos eventos concorrentes do SCTP. Dessa forma, esse trabalho é dividido em duas fases.

Na primeira fase, os modelos são desenvolvidos, empregando-se RdPs com temporização, afim de que se possa ter uma visão funcional em alto nível do protocolo, das restrições temporais inerentes às três principais fases da associação SCTP e das ações que podem ocorrer concorrentemente.

Análises formais dos modelos em RdPs com temporização são realizadas com o emprego do PAREDE para geração da árvore de alcançabilidade e verificação das propriedades dos modelos e da consistência do protocolo.

O PAREDE funcionou em um ambiente DOS de uma máquina 486 DX2 de 66 MHz com 32MB de RAM, o que limitou a capacidade de processamento a cerca de 40.000 estados.

Na segunda fase, o diagrama de estados alcançáveis do SCTP, presente na RFC 2960, que descreve sem muitos detalhes a associação SCTP, é modelado e analisado através da rede de Petri colorida e hierárquica (CPN) [Jens97]. Algumas limitações do diagrama são superadas, nos modelos desenvolvidos, pela incorporação de detalhes como: informações de cabeçalho (*header*) do pacote e dos *chunks* (mensagens de controle e de dados de usuário); e descrição da atualização das variáveis de estados do SCTP, fundamentais na descrição dos procedimentos da associação.

A validação e verificação do modelo CPN do SCTP são feitas pelo uso do pacote de programa chamado Design/CPN desenvolvido na Universidade de *Aarhus* em 1996 [Site2]. O Design/CPN rodou num ambiente Unix, de uma máquina com um processador de 1.3 GHz, 192 MB de RAM e sistema operacional LINUX Connectiva 8, e não ocorreram problemas com a quantidade de estados do sistema.

5.2 Modelamento e Análise do Mecanismo de Retransmissão por *Time-out*

Conforme foi descrito no capítulo IV, a RdP com temporização permite que se manipule, com certa facilidade, os eventos em que não se tem certeza do momento de suas ocorrências, mas se sabe de uma janela de tempo em que tais eventos devem ocorrer. Portanto, utilizar-se-á essa extensão na modelagem dos eventos de retransmissão de mensagens do sistema por estouro de *time-out.*

Pode-se ver, na rede da Figura 5.1, a seqüência de eventos ou evolução do sistema, que pode ser acompanhada pela seqüência das figuras de (a) a (f) e pelas setas que apontam a seqüência. A partir do estado inicial (a), o envio e confirmação

do recebimento da mensagem ou a perda de mensagem enviada e a ação do sistema de retransmissão por *time-out*, até o retorno ao estado inicial.

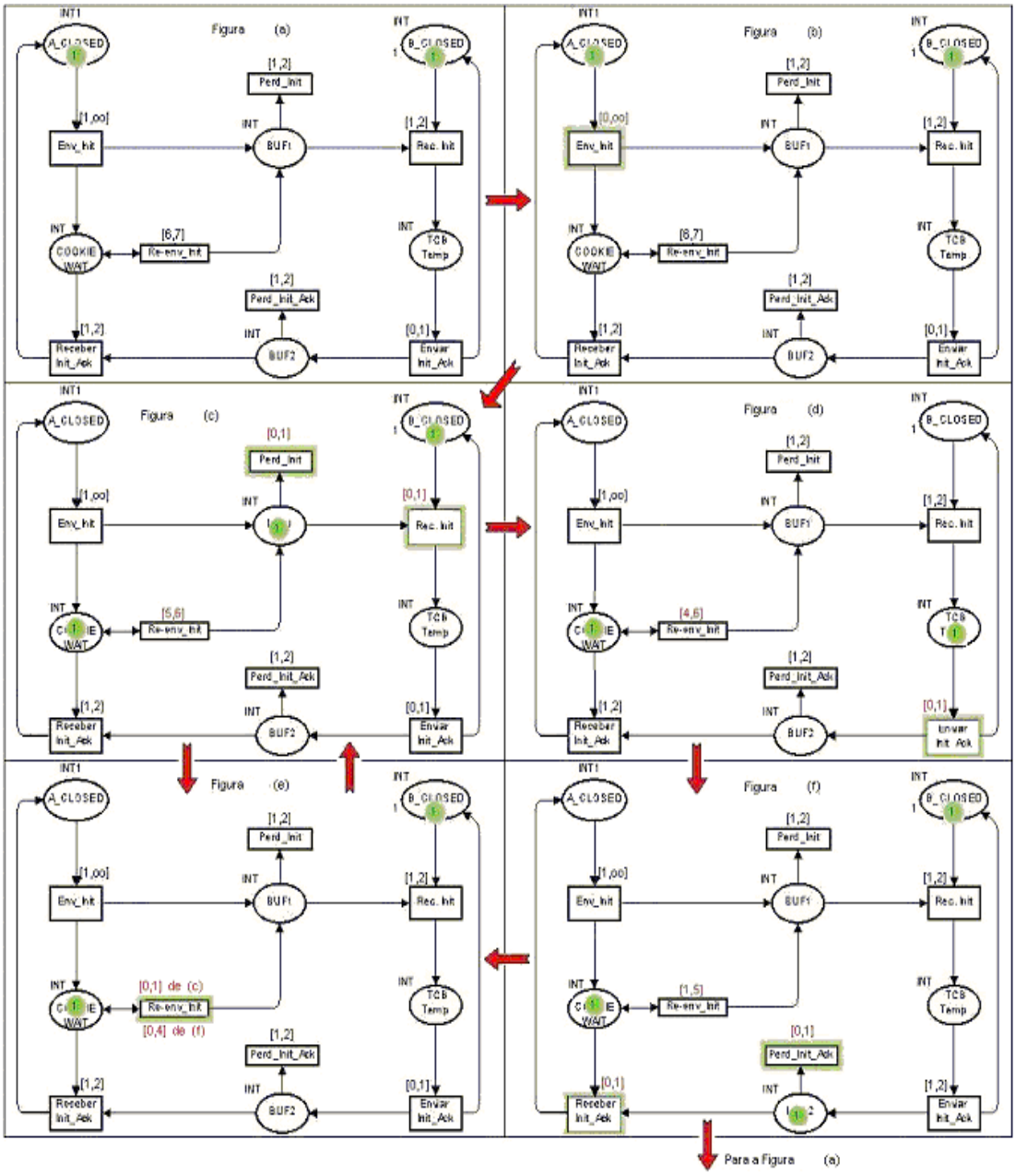


Figura 5.1 – *Simulação do two-way handshake com retransmissão por time-out.*

Observa-se que o ajuste do tempo de espera para retransmissão de mensagens é fundamental para o bom funcionamento e desempenho do sistema. Caso ele seja ajustado indevidamente, o sistema tem seu funcionamento comprometido. Na prática, esse ajuste é constantemente verificado e atualizado pela variável RTT (*Round Trip Time*), conforme descrito na seção 3.7.

Para o modelo da Figura 5.1 obtém-se o grafo de classes de estados da Figura 5.2, na qual podem ser observados eventos concorrentes em Q_1 e Q_4.

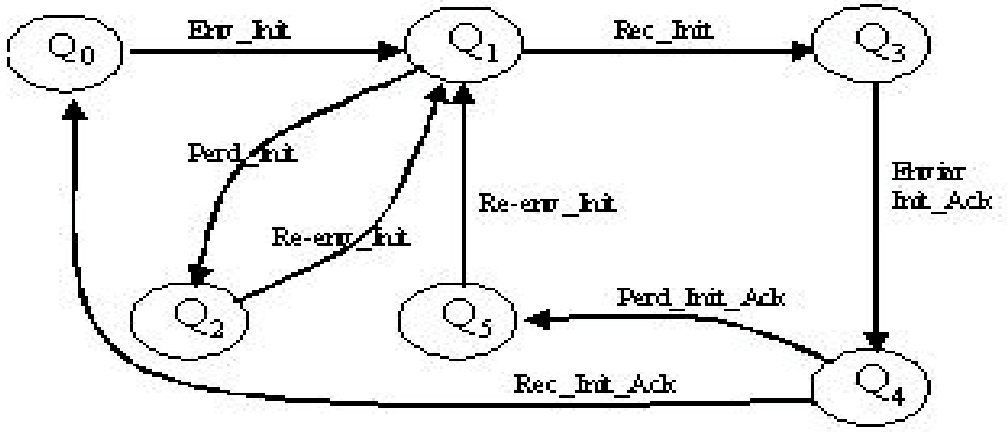


Figura 5.2 *– Grafo de Classes de Estados da rede da Figura 5.1 – RTO de [6,7]*

Esse grafo de classes de estados mostra 6 classes de estados alcançáveis, sendo Q_0 = (A_CLOSED, B_CLOSED), Q_1 = (COOKIE_WAIT, BUF1, B_CLOSED), Q_2 = (COOKIE_WAIT, B_CLOSED), Q_3 = (COOKIE_WAIT, TCB_TEMP), Q_4 = (COOKIE_WAIT, BUF2, B_CLOSED) e Q_5 = (COOKIE_WAIT, B_CLOSED), com intervalo de disparo para a transição **Re_env_Init** diferente do encontrado em Q_2. Por checagem do grafo acima podem ser observados os seguintes comportamentos do sistema:

- ✓ Os *loops* Q_1-Q_2-Q_1 **...** e Q_1-Q_3-Q_4-Q_5-Q_1 **...** podem ser interpretados como situações não desejadas como, por exemplo, no estabelecimento da associação SCTP. Conforme esses *loops* se apresentam, pode significar que o usuário (ULP) que solicitou a associação fique indefinidamente sem resposta do protocolo. Por isso, conforme especificado na RFC2960, o sistema deve limitar o número de retransmissões e isso é implementado com o acréscimo de um contador para limitar o número de retransmissões. Essa implementação é feita na rede da Figura 5.5, a ser apresentada e analisada na seção seguinte.

- ✓ A segunda situação observada diz respeito à especificação do valor da janela de retransmissão de mensagem (**Re_env_Init**). Caso esse valor seja menor que um valor mínimo (cálculo do RTT), a retransmissão da mensagem será

feita sem necessidade e o resultado é o comportamento incorreto do sistema. No grafo de classes de estados abaixo, pode ser visto o comportamento da rede da Fugura 5.1 para o caso de RTO = [5,7]. Para um sistema mais complexo, esse inconveniente poderia acarretar em uma seqüência tal que levaria o sistema a um bloqueio, conforme será verificado no item 5.3.1.

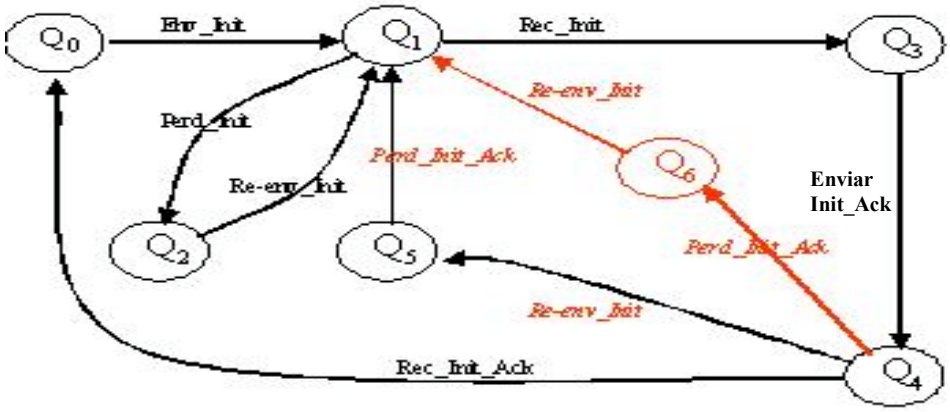


Figura 5.3 – *Grafo de Classes de Estados da rede da Figura 5.1, com RTO de [5,7]*

Para esse caso, apesar da alteração do RTO para [5,7] gerar o estado **Q_6**, não desejável ao sistema, por inspeção visual pode-se ver que se trata de um sistema limitado, vivo e reiniciável. Tais propriedades são confirmadas pelo PAREDE.

Para a análise dos casos onde o valor do RTO é menor que [5,7], onde são geradas mais de uma ficha nos lugares de entrada de **Perd_Init** e **Perd_Init_Ack**, foi necessário o acréscimo de arcos elementares à rede, ou seja, lugares de entrada e saída a estas transições, marcados com uma ficha, para que o PAREDE pudesse tratar o modelo. Assim, pode-se analisar o modelo para diversos valores de RTO. Para o caso de RTO definido para [4,7] obteve-se um grafo com 48 classes de estados. Para o caso de um RTO igual a [3,7], foram encontradas 41 classes de estados. Para RTO de [2,7] foram encontradas 105 classe de estados. Para todos esses valores de RTO, no modelo da Figura 5.1, as propriedades encontradas indicam um sistema limitado, vivo e reiniciável. Para RTO de [1,7] foram encontradas mais de 40.000 classes de estados. Nesse ponto, o PAREDE não continuou a análise devido à falta de recursos computacionais. Já para o RTO de [0,7] pode-se inferir, pelo valor mínimo de **Re_env_Init,** que seu disparo pode ser contínuo, levando o sistema a ser não-limitado.

Por outro lado, deixando o valor máximo do RTO igual a infinito, [7,∞], implicou a não existência da classe de estado $\mathbf{Q_5}$, conforme pode ser visto no grafo de classes de estados apresentado abaixo.

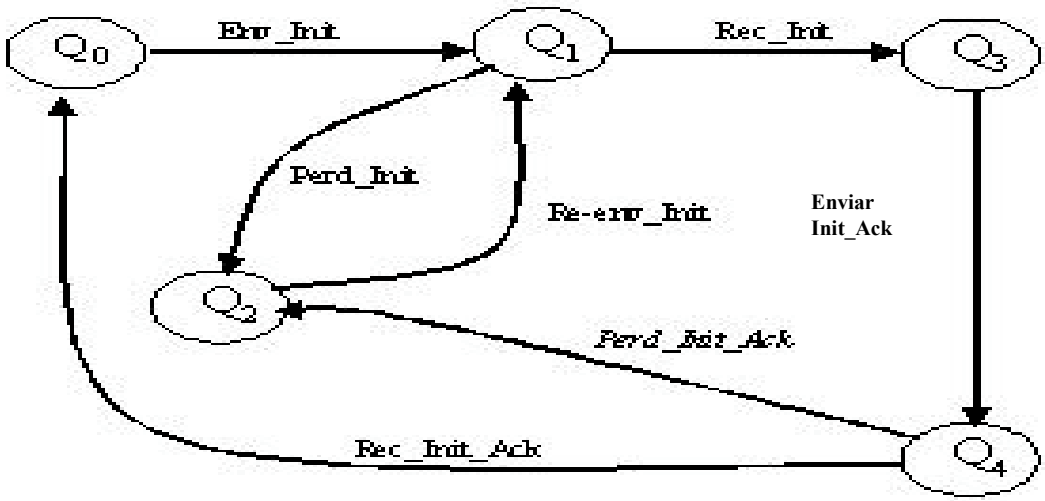


Figura 5.4 *– Grafo de Classes de Estados da rede da Figura 5.1, com RTO de [7,∞]*

Esse grafo, no qual a quantidade de estados é menor que do grafo anterior, pode aparentar que a resposta do sistema é mais rápida. No entanto, a definição do valor máximo de disparo de **RTO_Init** para infinito implica em perda do controle do desempenho do sistema. Ou seja, deve-se ter cuidado em associar um grafo com menor quantidade de estados com um sistema otimizado.

Essa análise comprova a importância do cálculo dos valores mínimo e máximo adequados para a retransmissão de mensagens por *time-outs (RTT)***.** O problema é encontrar o valor correto para RTO, tendo em vista a dependência do estado da rede (carga) e suas mudanças dinâmicas.

Convém notar que o PAREDE não considera a marcação inicial [1,∞], do quadro (a) da Figura 5.1. Portanto, para ele, os quadros (a) e (b) são idênticos, com Env_Init = [0, ∞].

5.3 Estabelecimento Normal da Fase de Associação em Alto nível

A partir da idéia de dividir a funcionalidade global do protocolo em três fases interdependentes e seqüênciais, inicia-se aqui a modelagem da primeira fase (estabelecimento da associação), seguindo as especificações da RFC 2960 [Stew00], que indicam que deve ser concluído um *four-way handshake*, visando a segurança da associação entre os *endpoints* contra ataques.

A RdP da Figura 5.5 descreve visualmente (em alto nível) os estados e eventos do SCTP para o controle de envio e recebimentos das mensagens do *four-way handshake*, num cenário Cliente-Servidor, além dos mecanismos de retransmissão de mensagens por *time-out* e dos mecanismos contadores de mensagens retransmitidas. A descrição dessa rede é apresentada em seguida.

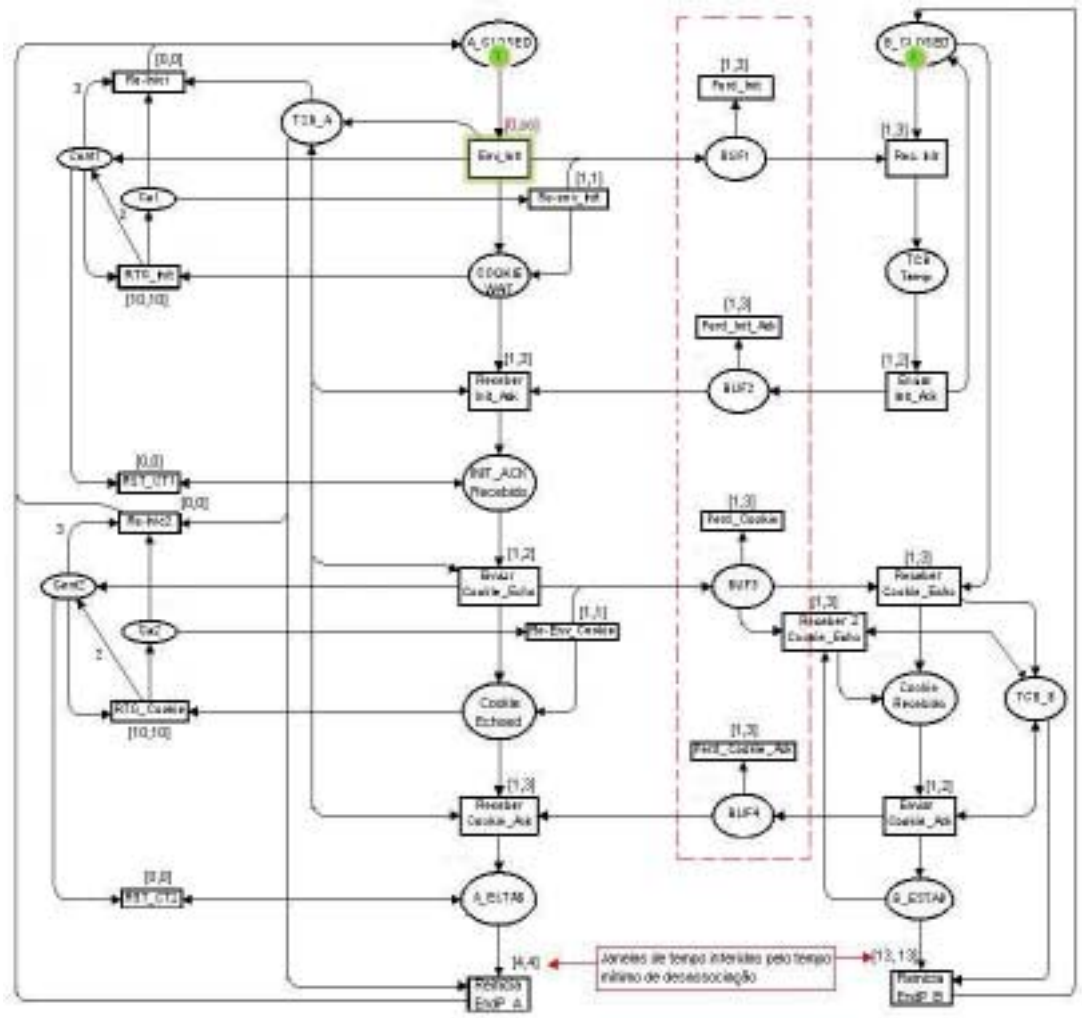


Figura 5.5 - *RdP do estabelecimento normal da Associação SCTP – (**Q_0**)*

A descrição dessa rede é apresentada em duas etapas. A primeira é uma descrição estática (sintática) do modelo e a outra etapa é a descrição da dinâmica da

rede (semântica). A rede estática é composta por duas sub-redes interconectadas, que são os dois *endpoints* que descrevem um sistema assíncrono, porém com alguns momentos onde essas sub-redes devem sincronizar-se para troca de mensagens.

A sub-rede do lado esquerdo representa estados e procedimentos realizados por um *endpoint* cliente, e a sub-rede do lado direito representa estados e procedimentos do *endpoint* servidor. Essa RdP é composta por 19 lugares (estados parciais do sistema) e 23 transições de estados (eventos do sistema). Os nós que estão dentro da área pontilhada indicam ocorrências no sistema no nível do meio de comunicação, ou seja, na rede IP, que devem ser tratados pelo protocolo.

A sub-rede do ***endpoint* cliente** tem os seguintes **lugares (estados)**:

1) **A_CLOSED** indica que o *endpoint* A (Cliente) encontra-se em seu estado inicial.
2) **COOKIE_WAIT** indica que o cliente está aguardando uma resposta à mensagem INIT enviada (a mensagem INIT_ACK).
3) **INIT_ACK_Recebido** indica que o cliente recebeu a mensagem de resposta do servidor.
4) **TCB_A** indica um bloco de controle de transmissão, um lugar onde são armazenadas as variáveis de estado da conexão SCTP para o *endpoin*t A.
5) **Cookie_Echoed** indica que o cliente ecoou o *cookie* recebido, conforme especificação do protocolo *four-way handshake,* e aguarda pela confirmação.
6) **A_ESTAB** indica que o cliente recebeu a mensagem COOKIE_ACK, confirmando que está devidamente conectado ao servidor.
7) **Cont1** indica um contador de mensagens INIT enviadas.
8) **CA1** indica uma cópia da mensagem INIT gerada por *time-out* pronta para ser reenviada ao destino.
9) **Cont2** indica um contador de mensagens COOKIE_ECHO enviadas.
10) **CA2** indica uma cópia da mensagem COOKIE_ECHO gerada por *time-out*, pronta para ser reenviada ao destino.

A sub-rede do ***endpoint* cliente** tem as seguintes **transições (ações)**:

1) **Env_Init** indica a ação de envio da mensagem INIT.
2) **Receber_Init_Ack** indica a ação de receber a mensagem INIT_ACK.
3) **Env_Cookie_Echo** indica a ação de envio da mensagem COOKIE_ECHO.
4) **Receber_Cookie_Ack** indica a ação de receber a mensagem COOKIE_ACK.
5) **RTO_Init** indica a ação de gerar uma cópia da mensagem INIT e disponibilizá-la para reenvio ao destino.
6) **Re-Inic1** indica a ação de reiniciar o sistema devido ao contador de mensagens INIT sem resposta ter chegado ao valor preestabelecido.
7) **RST_CT1** indica a ação de reiniciar o contador de mensagens INIT enviadas.
8) **Re-env_Init** indica a ação de reenvio da mensagem INIT.
9) **RTO_Cookie** indica a ação de gerar uma cópia da mensagem COOKIE_ECHO e disponibilizá-la para reenvio ao destino.
10) **Re-Inic2** indica a ação de reiniciar o sistema devido ao contador de mensagens COOKIE_ECHO sem resposta ter chegado ao valor máximo preestabelecido.
11) **RST_CT2** indica a ação de reiniciar o contador de mensagens COOKIE-ECHO enviadas.
12) **Re_Env_Cookie** indica a ação de reenvio da mensagem COOKIE_ECHO.
13) **Reinicia_EndP_A** indica a função de reiniciar o *endpoint* cliente após todos os procedimentos da fase de conexão terem sidos realizados, de forma que o modelo de rede de Petri possa manter a representatividade do sistema real sem perda de propriedades.

A sub-rede do ***endpoint* servidor** tem os seguintes **lugares (estados):**

11) **B_Closed** indica que o *endpoint* B (Servidor) encontra-se em seu estado inicial.
12) **TCB_Temp** indica que o servidor recebeu uma mensagem INIT e está gerando um *cookie* para esta mensagem.

13) **Cookie_Recebido** indica que o servidor recebeu uma mensagem COOKIE_ECHO.
14) **TCB_B** indica um bloco de controle de transmissão, um lugar onde são armazenadas as variáveis de estado da conexão SCTP para o *endpoint* B.
15) **B_ESTAB** indica que o servidor confirmou a validação da mensagem COOKIE_ACK, já enviou a confirmação da conexão para o cliente e está pronto para enviar e receber mensagens de usuários.

A sub-rede do ***endpoint* servidor** tem as seguintes **transições (ações):**

14) **Rec_Init** indica a ação de receber a mensagem INIT.
15) **Enviar_Init_Ack** indica a ação de envio da mensagem INIT_ACK.
16) **Receber_Cookie_Echo** indica a ação de receber a mensagem COOKIE_ECHO.
17) **Receber_Cookie_Echo2** indica a ação de receber a mensagem COOKIE-ECHO estando o servidor já no estado B_ESTAB.
18) **Enviar_Cookie_Ack** indica a ação do servidor de enviar a mensagem COOKIE_ACK para o cliente.
19) **Reinicia EndP_B** indica a ação de reiniciar o *endpoint* servidor após todos os procedimentos da fase de conexão terem sidos realizados, de forma que modelo de rede de Petri possa manter a representatividade do sistema real sem perda de propriedades.

Os estados do meio de comunicação são:

16, 17, 18 e 19 - São os *Buffers* (**BUF1, BUF2, BUF3 e BUF4**) indicando canais lógicos de comunicação unidirecionais, onde uma mensagem pode estar em trânsito, do cliente para o servidor, ou do servidor para o cliente, pode estar em qualquer lugar na rede ou mesmo num buffer da entidade local.

As transições do meio de comunicação são:

20) **Perd_Init** indica a ação de perder a mensagem INIT.

21) **Perd_ Init_Ack** indica a ação de perder a mensagem INIT_ACK.

22) **Perd_Cookie** indica a ação de perder a mensagem COOKIE_ECHO.

23) **Perd_Cookie_Ack** indica a ação de perder a mensagem COOKIE_ACK.

Após essa descrição sintática do modelo, será considerada agora a semântica, ou seja, o comportamento do modelo.

Conforme o modelo de RdP com temporização que a Figura 5.5 mostra, os *endpoints* podem assumir diversos estados e realizar certas funções nesses estados. Os estados dos *endpoints* são sub-estados do protocolo. No estado inicial do sistema, ambos os *endpoints* estão no estado **CLOSED** e, conforme o modelo expressa, a única ação que pode ocorrer no sistema, quando no estado inicial, é o *endpoint* cliente enviar uma mensagem ao *endpoint* destino chamada **INIT**. As mensagens de controle trocadas entre os *endpoints* podem ser perdidas e o protocolo SCTP é inteligente o suficiente para superar esse problema e garantir a continuidade do processo de associação.

Conforme especificação do protocolo SCTP, RFC2960, o protocolo deve ter um controle sobre a quantidade máxima de vezes que um *endpoint* deve enviar mensagens a outro *endpoint* sem receber respostas. Caso esse valor máximo, *Threshold,* seja alcançado, o sistema deve abortar o processo de iniciação e retornar para seu estado inicial.

O término do processo da associação se dá quando ambos os *endpoints* envolvidos estão no estado "ESTABELECIDO", prontos para iniciar a fase de troca de dados de usuários (ULP - *Upper Layer Protocol*) ou para retornar para CLOSED.

Os passos a seguir descrevem a simulação do modelo. Equivale à seqüência de eventos do sistema.

1 - O Disparo de **Env_Init**. Essa é a primeira ação possível do sistema a partir do estado inicial. Iniciar o *four-way handshake*. O resultado do disparo de **Env_Init** é mostrado na RdP apresentada pela figura 5.6.

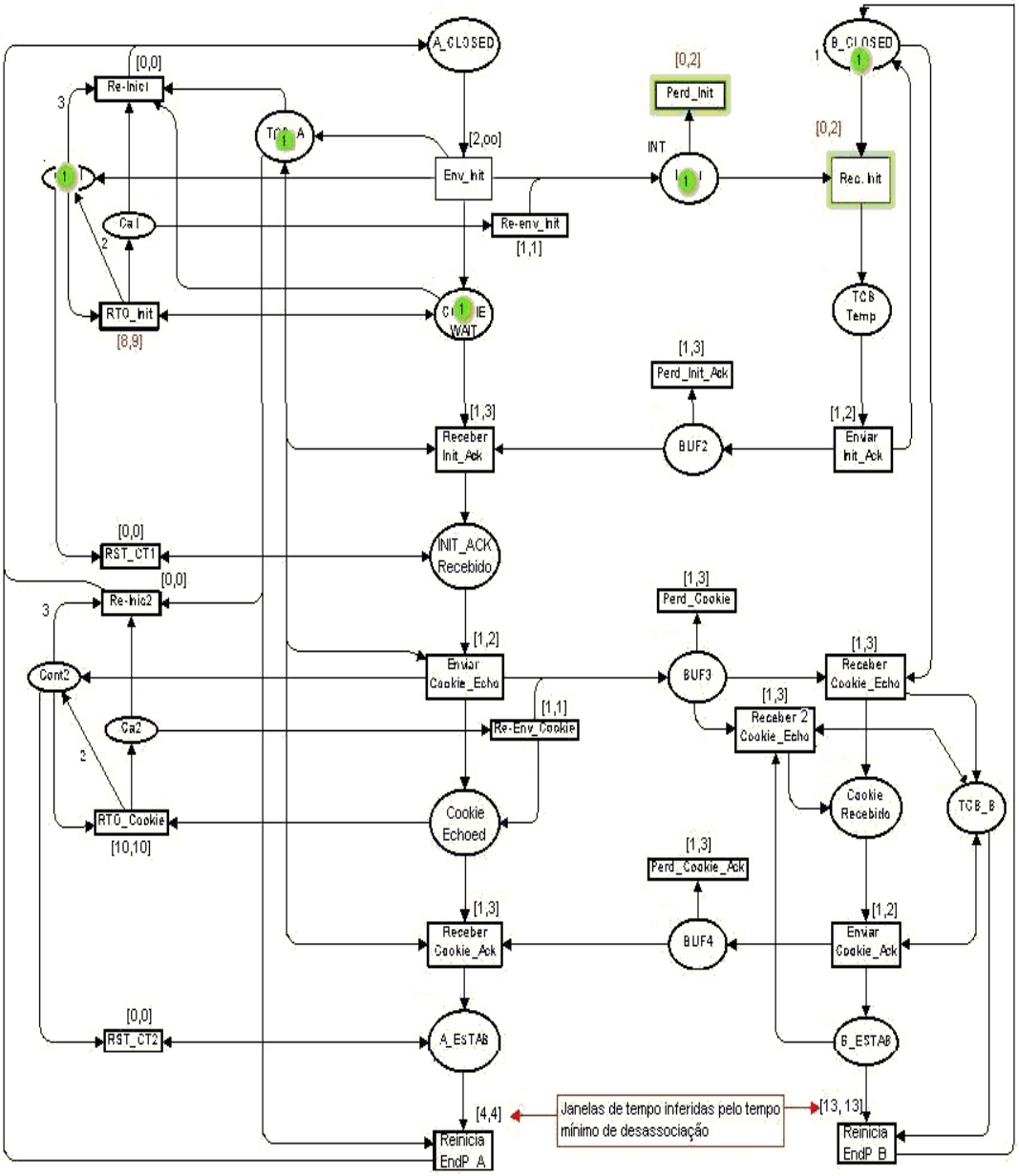


Figura 5.6 – *Modelo de RdP do Four-way Handshake (**Q_1**).*

Conforme a RdP da Figura 5.6 mostra, no estado **Q_1** há dois eventos passíveis de ocorrência, o passo 2 **ou** o passo 3, detalhados a seguir.

2 - Disparo de **Perd_Init**. O que ocorre nesse caso é a perda da mensagem pelo meio de transmissão. O *endpoint* cliente deve executar o procedimento de retransmissão da mensagem, conforme especificação. Após a retransmissão de **INIT**,

o sistema retorna para a classe de estado $\mathbf{Q_1}$. Caso o valor máximo de *Threshold* seja alcançado, o *endpoint* destino é considerado não alcançável. Neste caso, o sistema é reiniciado, retornando o *endpoint* cliente para o estado inicial (**CLOSED**), retornando a execução do sistema para o passo 1.

3 - Disparo de **Rec_Init**. Significa que o *endpoint* destino recebe a mensagem do endpoint cliente. O servidor sai do estado **CLOSED** e vai para o estado **TCB_Temp**. Dessa forma, o próximo evento possível no sistema só pode ser o descrito no passo 4.

4 - Disparo de **Env_Init_Ack**. Significa que o *endpoint* destino enviou uma mensagem de resposta ao cliente e retornou para o estado **CLOSED**. Nesse caso os próximos eventos possíveis podem ser os descritos pelos passos 5 ou 6.

5 - Disparo de **Perd_Init_Ack**. Nesse caso, o sistema comporta-se semelhantemente ao passo 2.

6 - Disparo de **Receber_Init_Ack**. Significa que o *endpoint* cliente recebe a mensagem que estava esperando do servidor e pode dar prosseguimento à fase de conexão. O disparo de **Receber_Init_Ack** também realiza o procedimento de desativação da retransmissão da mensagem **INIT** por "*time-out*". Antes de prosseguir com o procedimento de associação, é necessário ainda que o contador de mensagem **INIT** enviada seja zerado. Isso se dá pelo disparo de **RST_CT1**, tantas vezes quanto o número de mensagens **INIT** enviadas. Após estes procedimentos o próximo passo é o descrito em 7.

7 - Disparo de **Env_Cookie_Echo**. Significa que o cliente envia a mensagem **Cookie-Echo** para o servidor e fica aguardando resposta. Um procedimento de controle de mensagem, idêntico ao da mensagem **INIT** é iniciado. Os próximos passos do sistema são o 8 ou o 9.

8 - Disparo de **Perd_Cookie_Echo**. Nesse caso, o sistema comporta-se de modo idêntico ao descrito no passo 2, porém agora com a retransmissão da

mensagem **Cookie_Echo**. Caso o número de mensagens enviadas, sem resposta, chegue ao valor máximo definido, o sistema é reiniciado, voltando para o passo 1.

9 - Disparo de **Receber_Cookie_Echo**. Significa que o Destino recebeu a mensagem. Nesse caso, o Destino muda de estado **CLOSED** para **Cookie_Recebido**. Dessa forma, o próximo evento possível é definido como o passo 10.

10 - Disparo de **Enviar_Cookie_Ack**. Significa que o *endpoint* servidor envia uma mensagem **Cookie_Ack** e muda seu estado para **B_ESTAB**. Ou seja, o servidor está pronto para trocar dados de usuários com o *endpoint* cliente. Após o passo 10, os passos seguintes passíveis de ocorrência são os passos 11 e (12 ou 13).

11 - Disparo da transição **Reinicia_EndP_B** (após um certo tempo). Esse disparo leva o *endpoint* servidor ao seu estado Inicial, **B_CLOSED**. Observe-se que esta transição substitui, em parte, os subsistemas de troca de dados de usuário e de desassociação.

12 - Disparo de **Perd_Cookie_Ack**. Nesse caso o sistema toma o procedimento conforme descrito no passo 8.

13 - Disparo de **Receber_Cookie_Ack**. Nesse caso, o *endpoint* cliente recebe a mensagem que estava esperando do servidor e muda seu estado para **A_ESTAB**. Também realiza o procedimento de desativação da retransmissão da mensagem **Cookie_Ack** por "*time-out*". Antes de iniciar qualquer procedimento de envio de dados de ULP, é necessário ainda que o contador de mensagem **Cookie_Echo** enviada seja zerado. Isto se dá pelo disparo de **RST_CT2**, tantas vezes quanto o número de mensagens **Cookie_Ack** enviadas. Após esses procedimentos o próximo passo é o descrito em 14.

14 - Disparo de **Reinicia_EndP_A**. Essa transição substitui os subsistemas de troca de dados de ULP e de desassociação. Após o disparo dessa transição, o *endpoint* cliente retorna ao estado **CLOSED**.

É oportuno observar que os disparos de algumas transições são concorrentes e de outras, são paralelos. Em seguida, são apresentadas algumas análises sobre o modelo anterior, de forma a se verificar as propriedades comportamentais do sistema.

5.3.1 Análises da RdP da Fase de Estabelecimento

A RdP da Figura 5.5 foi implementada e analisada pelo PAREDE, que encontrou todas as classes de estados possíveis, e o resultado está apresentado na forma de um grafo de alcançabilidade, conforme Figura 5.7.

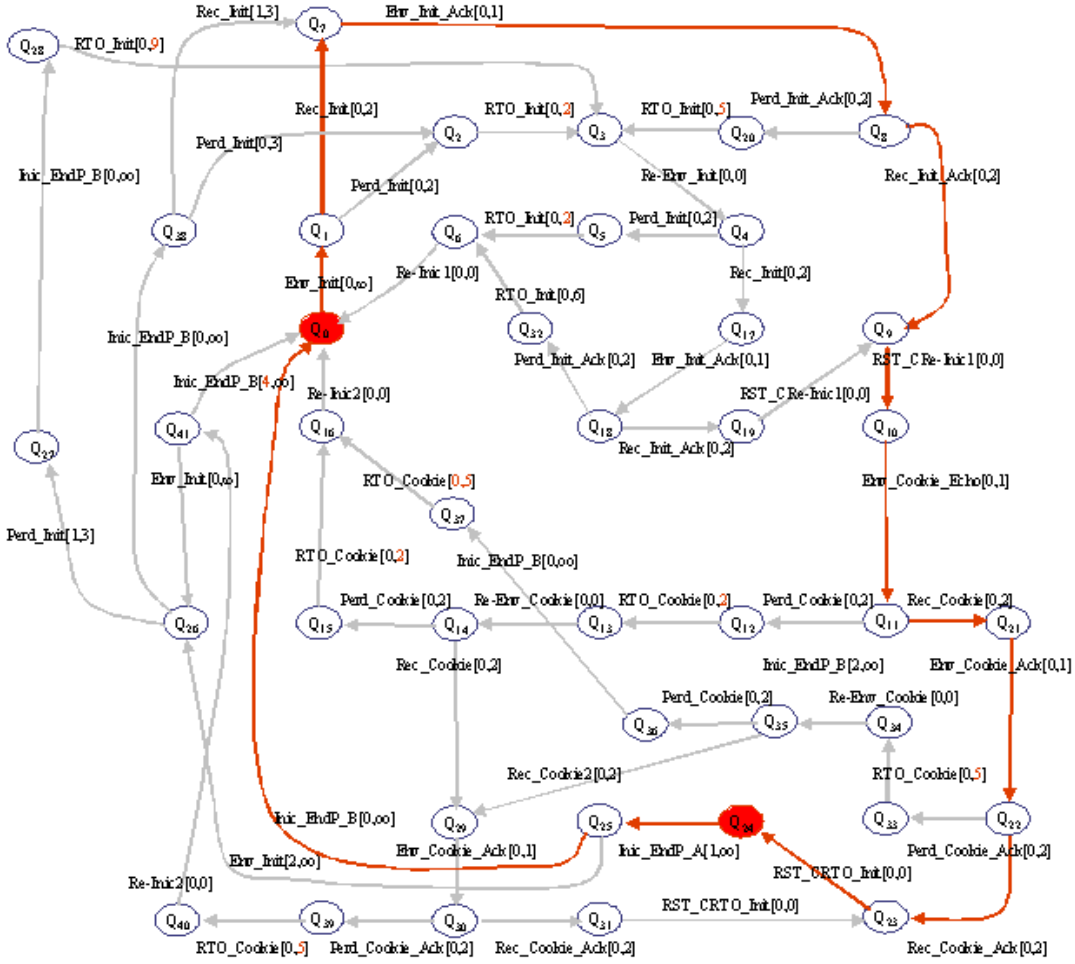


Figura 5.7 – *Grafo de Classe de estados da fase de associação normal*

A primeira informação interessante que se pode obter do grafo é quanto à grande quantidade de estados relativos ao sistema para a troca de apenas quatro mensagens. Os arcos em vermelho indicam a seqüência de disparo mais curta que leva o protocolo do estado CLOSED ao estado ESTABELECIDO e deste de volta a CLOSED. Todos os demais arcos e marcações do grafo revelam comportamentos aleatórios do sistema.

Pode-se perceber também que tais eventos aleatórios estão sob controle, pois, apesar de serem inerentes e causarem atrasos ao sistema, não causam outros prejuízos. A quantidade limitada de classes de estados (42) indica um sistema limitado. A não existência de transições não vivas indica um sistema vivo, e o fato de Qo ser alcançável de qualquer marcação indica um sistema reiniciável (Observar que não existe nenhum loop que não passe por Qo).

As janelas de tempo de **Reinicia_EndP_A** e de **Reinicia_EndP_B** não podem ser menores que um mínimo, pois podem levar o sistema a um estado indesejável. Essa é uma situação normalmente crítica, onde os tempos dos eventos precisam ser bem definidos para que o sistema não tenha crise em seu comportamento. Alguns valores de janelas de disparo de **Reinicia_EndP_A** e **Reinicia_EndP_B** foram simulados e os resultados são mostrados na tabela seguinte, onde se pode observar que, para alguns valores definidos, o sistema tem sua capacidade de controle comprometida.

CENÁRIOS(C) / PROPRIEDADES	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Reinicia_EndP_A	[1,1]	[1,1]	[1,1]	[1,1]	[1,1]	[1,1]	[2,2]	[3,3]	[4,4]	[5,5]	[11,oo]
Reinicia_EndP_B	[1,1]	[3,3]	[6,6]	[7,7]	[8,8]	[17,17]	[13,13]	[13,13]	[13,13]	[13,13]	[13,oo]
Numero de Classes de Estados Distintas Analisadas	40	44	45	55	57		46	45	42	45	51
Rede Limitada	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Rede Viva	Nao	Nao	Nao	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Rede Reinicializavel	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Tabela 5.1 – *Resultados das Analises dos Grafos de Estados da Rede da Figura 5.5 para alguns valores de Janela de disparo das transicoes Reinicia_EndP_A e Reinicia_EndP_B.*

Conforme a tabela 5.1, os três primeiros cenários apresentaram a propriedade de rede não viva. Isto devido à transição **Receber_Cookie_Echo2** ser não-viva quando aqueles valores de Janela de disparo das transicoes **Reinicia_EndP_A** e **Reinicia_EndP_B** são definidos. Para o último cenário, o grafo de alcançabilidade é mostrado na Figura 5.8.

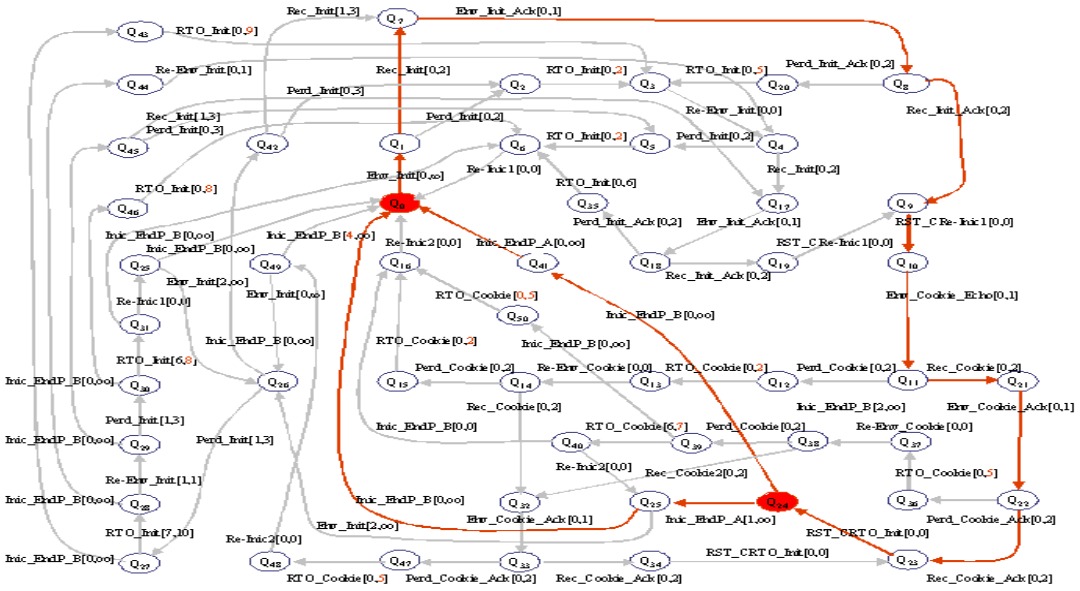


Figura 5.8 – *Grafo com 51 Classes de estados da fase de associação normal(C11)*

Pode ser visto, na rede da Figura 5.5, que as duas últimas mensagens do *four-way handshake* somente iniciam após o término das duas primeiras. A análise realizada, no PAREDE, para as duas primeiras mensagens indicam 16 classes de estados, considerando apenas uma retransmissão. Como o grafo das quatro mensagens tem 42 classes de estados, concluímos que as duas últimas mensagens têm 26 classes de estados. A diferença do número de classes de estados das primeiras duas mensagens para as duas últimas é devida ao comportamento assíncrono inerente ao sistema. Por exemplo, os disparos de **Reinicia_EndP_A** e de **Env_Init**, sem o disparo de **Reinicia_EndP_B**, leva o sistema a seguir uma seqüência maior (mais demorada) para retornar a $\mathbf{Q_0}$.

No caso da RdP apresentada na Figura 5.5, as janelas de tempo definidas para o disparo de **Reinicia_EndP_A** e **Reinicia_EndP_B,** dimensionadas respectivamente para [4, 4] e [13, 13], viabilizam o comportamento adequado para o modelo e minimizam a quantidade de estados aleatórios do sitema.

5.4 RdP para Tratamento de Mensagens Duplicadas/Inexperadas

A RdP da Figura 5.9 especifica o comportamento do protocolo SCTP para o tratamento de eventos relacionados à iniciação simultânea da associação por ambos os *endpoints*. A Rede foi desenvolvida seguindo as especificações contidas na RFC 2960 [Stew00], seção 5.2.1, e apresentadas no capítulo III, seção 3.6.2, desta dissertação.

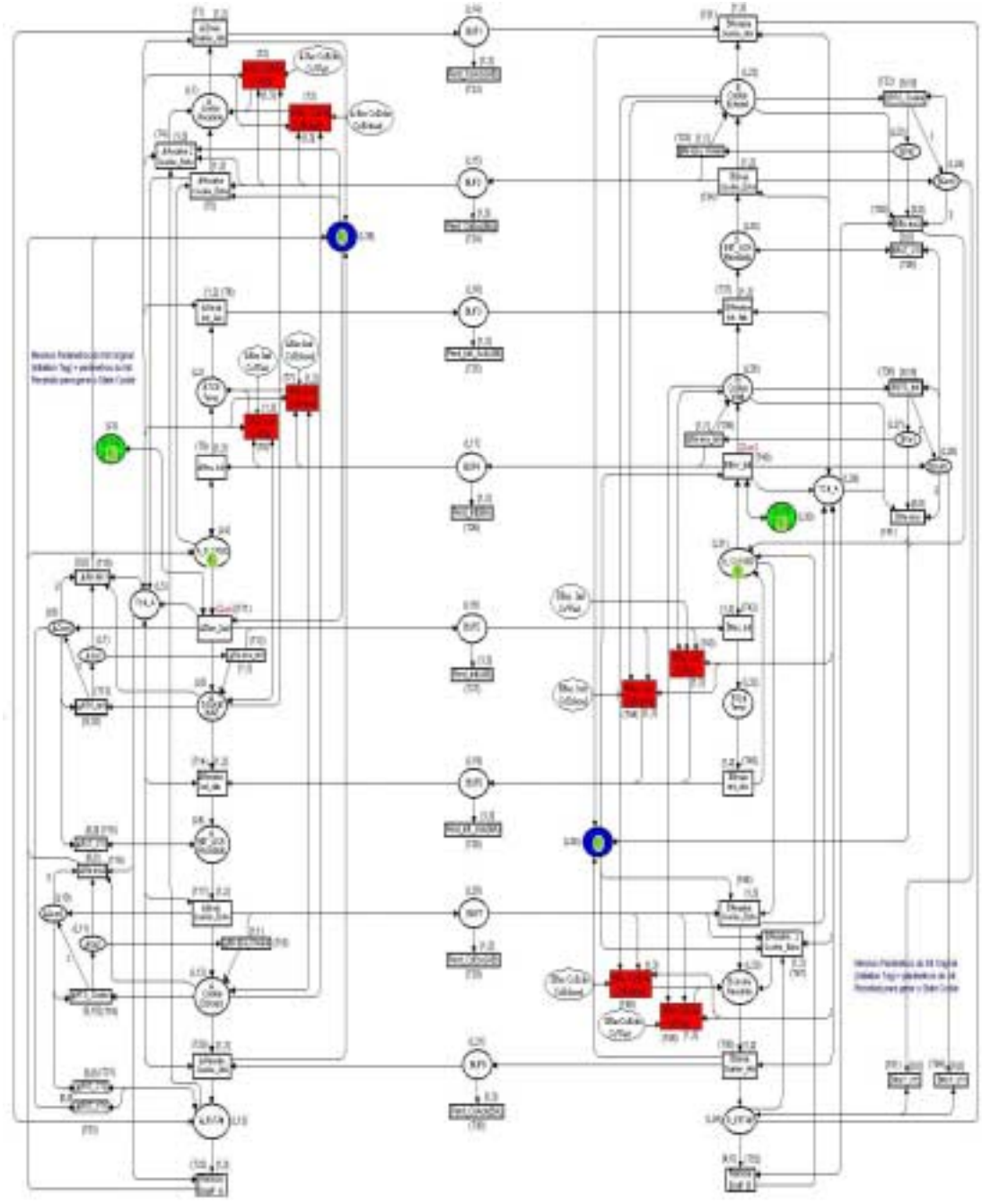


Figura 5.9 – *RdP da fase de Associação, abordando eventos duplicados ou inesperados*

O modelo de RdP da Figura 5.9 tem a mesma estrutura sintática e semântica apresentada na seção 5.3, porém com a diferença que essa rede é composta de duas sub-redes simétricas, demonstrando a idêntica estrutura lógica e funcional dos *endpoints*. Essa RdP tem, basicamente, três dispositivos não existentes na rede anterior (Figura 5.5). O primeiro é formado pelas transições em vermelho (T2, T3, T7, T8, T43, T44, T48 e T49), que tratam as mensagens aleatórias do sistema. O segundo é implementado pelos lugares em azul (L35 e L36), que fazem o controle da associação para o caso de TCB existente. E o terceiro é a presença dos lugares que representam as primitivas dos usuários SCTP (L3 e L30) que, quando marcados, fazem com que os *endpoints* operem também como cliente. Do contrário, o *endpoint* opera apenas como servidor. Outras descrições serão apresentadas na análise do modelo.

5.4.1 Análises da RdP da seção 5.4

A análise da RdP da Figura 5.9 foi realizada em três etapas. Na primeira etapa, o lugar que representa a primitiva do usuário, L30, foi deixado sem marcação. Isso faz com que a rede tenha o comportamento idêntico à rede da Figura 5.5, inclusive quanto à quantidade de classes de estados. No entando, metade de sua estrutura fica inoperável. O relatório do PAREDE confirmou que, para esse cenário, a rede é: Limitada, Não-viva e Reiniciável em Qo. A lista de transições não-vivas é a que se segue: T1, T2, T3, T4, T5, T6, T7, T8, T9, T23, T24, T25, T26, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T43, T44, T48, T49, T51, T53 e T54. Não existem classes de estados com bloqueio e o número de classes de estados distintas analisadas é 47.

Na segunda etapa da análise da RdP da seção 5.4, o lado oposto do sistema foi simulado, com a marcação de L30 e a desmarcação de L3. Conforme esperado, devido à simetria do sistema, o comportamento foi basicamente igual ao anterior. O relatório do PAREDE confirma que, para esse cenário, a rede é: Limita, Não-viva e Reiniciável em Qo. A lista de transições não-vivas é a que se segue: T2, T3, T7, T8,

T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T27, T28, T29, T30, T42, T43, T44, T45, T46, T47, T48, T49, T50, T53 e T54. Não existem classes de estados com bloqueio e o número de classes de estados distintas analisadas é 48.

Na terceira etapa, ambos os lugares L3 e L30 (em verde) foram marcados. Isso ocasionou os eventos de mensagens duplicadas e/ou inesperadas. O relatório do PAREDE confirma que, para esse cenário, a rede é Limitada, Viva e Reiniciável em Qo. O número de classes de estados distintas analisadas foi de 3549. Devido à grande quantidade de classes de estados, não é exeqüível a apresentação do grafo de alcançabilidade.

Esse relatório sobre as propriedades da RdP da Figura 5.9 é apresentado na tabela 5.2.

L30	L3	Qtd. Classes	Limitada	Viva	Reinicializável	
Não	Sim	47	Sim	Lista de Transições não vivas: T1, T2, T3, T4, T5, T6, T7, T8, T9, T23, T24, T25, T26, T31, T32, T33, T34, T35, T36, T37, T38, T39, T40, T41, T43, T44, T48, T49, T51, T53 e T54	Sim	
Sim	Não	48	Sim	Lista de Transições não vivas: T2, T3, T7, T8, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T27, T28, T29, T30, T42, T43, T44, T45, T46, T47, T48, T49, T50, T53 e T54.	Sim	
Sim	Sim	3.548	Sim	Sim	Sim	
Sim	Sim	6.722	Sim	Sim	Sim	Com 2 retransmissões

Tabela 5.2 – *Resultados das Analises da RdP da Figura 5.9, realizadas em três etapas.*

Para a pesquisa desse Grafo de Estados, o PAREDE utilizou 1284.6 Kbytes e demandou 00:15:46.60 h. Para a verificação das Propriedades 92.7 Kbytes e 00:01:43.20 h, totalizando 1377.3 Kbytes e 00:17:29.80 h.

5.5 Fase de Transferência de Dados de Usuários SCTP

Após o estudo em alto nível da fase de associação, inicia-se agora o estudo da segunda fase do protocolo SCTP: a transferência de dados, que inicia após o término da primeira fase. A transmissão de dados se dá nos dois sentidos. Assim, essa fase é representada pelas duas redes independentes da Figura 5.10.

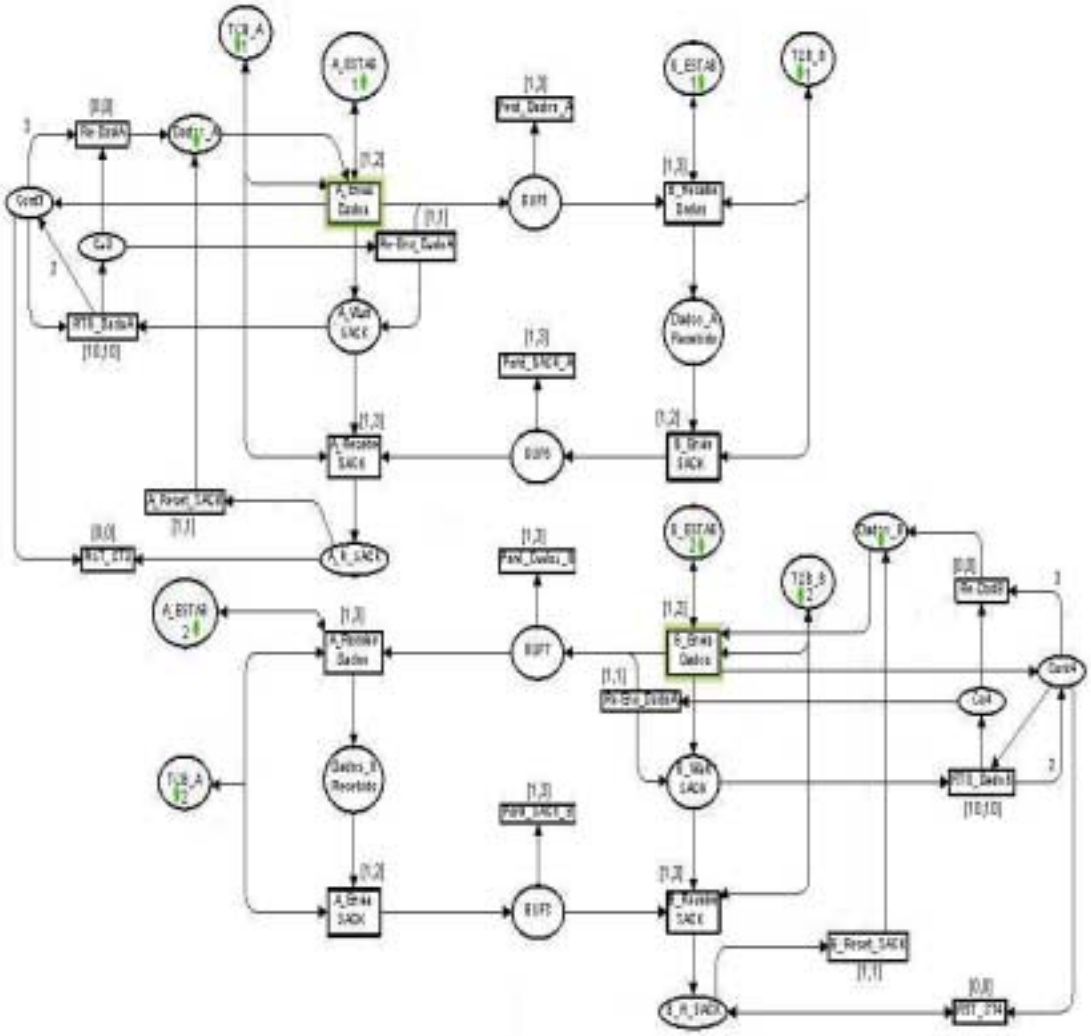


Figura 5.10 - *RdP da fase de transferência de dados ULP (**Q_0**)*

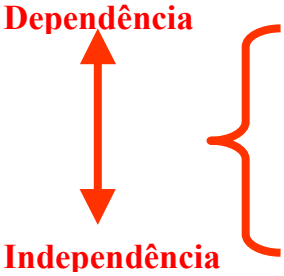
A RdP elaborada para a transmissão de dados (Fig. 5.10) segue a especificação contida na seção 9 do protocolo [Stew00], com o mesmo nível de abstração apresentado na fase de associação anterior, onde se pretende apenas abordar o controle de fluxo *fullduplex* de uma mensagem de dados e a quantidade de estados que envolvem o sistema nesta fase, para esse nível de abstração. A RdP

mostra que os *endpoints*, a partir do estado ESTABELECIDO, podem iniciar a transferência de dados de usuários (ULP) em ambos os sentidos ao mesmo tempo. Pode-se ver o paralelismo entre as ações de enviar e receber dados, e entre as ações de confirmação de dados recebidos.

Destaca-se, na RdP, o fato dos lugares **A_ESTAB, B_ESTAB, TCB_A** e **TCB_B** estarem duplicados, existindo um para a transmissão e outro para a recepção. Isso é necessário para se evitar que o disparo de uma transição que compartilha um desses lugares possa reiniciar o intervalo de disparo de outra transição. Isso ocorre, por exemplo, com as transições **A_Envia Dados** e **A_Recebe Dados**, que compartilham o lugar **A_ESTAB**. Comportamento semelhante ocorre com as transições relacionadas com os demais lugares citados.

5.5.1 Análises da RdP da Fase de Transferência de Dados

Percebe-se que essa RdP tem a mesma estrutura que a RdP da primeira fase do *four-way handshake* com uma retransmissão, que tem 16 classes de estados. Assim, poder-se-ia concluir que o grafo de alcançabilidade desse modelo teria 16 x 16 = 256 classes de estados. Porém, isso não se confirma, conforme a tabela 5.3 mostra e é explicada em seguida.



TCB-A	TCB-B	ESTAB-A	ESTAB-B	Qtd.Classes
Sim	Não	Sim	Não	2936
Sim	Sim	Sim	Não	3100
Sim	Sim	Sim	Sim	3444

Tabela 5.3 – *Resultados das Analises da RdP da Figura 5.10, relação de dependência entre as variáveis TCB e ESTAB ne transmissão e recepção de dados de usuários .*

Através do PAREDE, verificou-se os grafos de alcançabilidade para três estruturas dessa RdP. Na primeira, com um único lugar **TCB** e um único lugar **ESTAB** para transmissão e recepção, totalizando 20 lugares e 22 transições, foram encontradas 2.936 classes de estados. Na rede com dois TCB's e um único lugar **ESTAB,** para transmissão e recepção, totalizando 22 lugares e 22 transições, o PAREDE encontrou 3.100 classes de estados. E, finalmente, para a rede com dois subsistemas totalmente independentes (Fig.5.10), o PAREDE encontrou 3.444 classes de estados. Todos os modelos estudados possuem as propriedades de limitação, vivacidade e são reiniciáveis para $\mathbf{Q_0}$.

Ainda nessa RdP da Fig. 5.10, o PAREDE efetuou análises, em um de seus subsistemas (um lado da transmissão de dados), com o fim de validar o dispositivo contador de retransmissões de mensagens. O resultado mostra que para uma retransmissão obtém-se 16 classes de estados; para duas retransmissões, 23 classes de estados; para três retransmissões, 30 classes de estados; para quatro retransmissões, 37 classes de estados; para 10 retransmissões, 79 classes de estados; para 60 retransmissões, 429 classes de estados. Em todos os casos, o PAREDE confirmou as propriedades de limitação, vivacidade e reiniciação para $\mathbf{Q_0}$. Esse relatório é mostrado na tabela 5.4.

Qtd. Retransm.	Qtd. Classes
1	16
2	23
3	30
4	37
10	79
60	429

Tabela 5.4 – *Resultados das Analises da RdP da Figura 5.10. Quantidade de retransmissões x Quantidade de Classes de Estados alcançadas.*

O modelo de transmissão de dados foi desenvolvido sem muitos detalhes, tendo em vista que, nessa dissertação, não há preocupação com detalhes dessa fase, por exemplo, o controle de fluxo. O modelo dessa fase é desenvolvido com estrutura suficiente para se verificar as relações entre essa e sua antecessora e dessa com sua sucessora. Antes da apreciação do modelo da fase de desassociação é conveniente verificar o modelo resultante das duas primeiras fases já vistas.

5.6 RdP e Análises das duas primeiras Fases do SCTP

A primeira RdP, apresentada na Figura 5.11, representa a fase de associação seguida do envio de dados pelos clientes. A rede é composta de 34 lugares (sub-estados) e 47 transições.

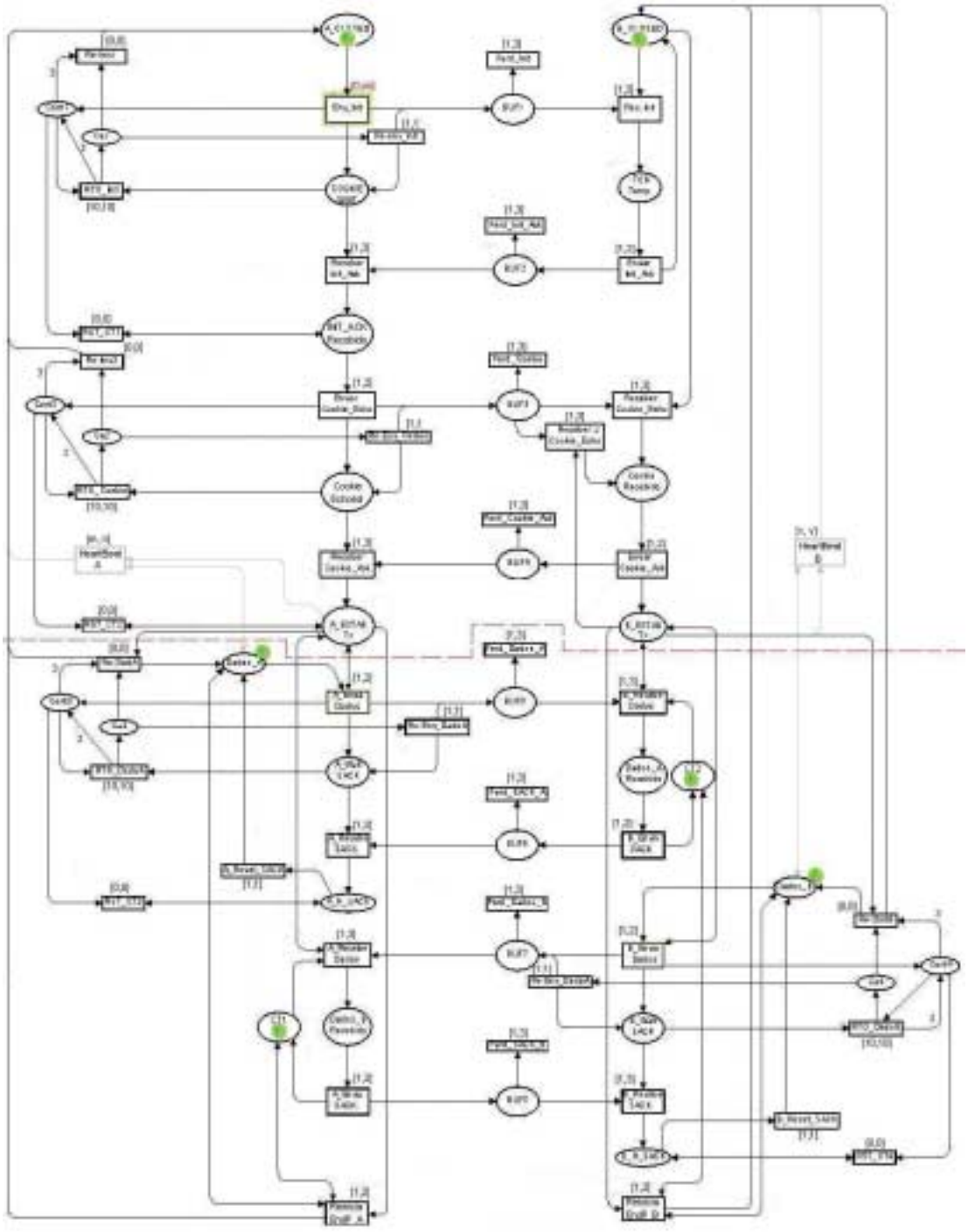


Figura 5.11 – *RdP das duas primeiras fases do SCTP*

A análise da RdP das duas primeiras fases do SCTP é feita em três etapas. Na primeira etapa considera-se que apenas o *endpoint* Cliente tem dados a enviar. Na segunda etapa, apenas o *endpoint* Servidor tem dados de usuários a enviar. Na terceira etapa ambos os endpoint têm dados de usuários a enviar. O resultado da análise é apresentado na tabela 5.5.

Dados_A	Dados_B	Qtd. Classes	Limitada	Viva	Reiniciável	Valores HeartBeats
Sim	Não	85	Sim	Quantidade de Transições não vivas: 12	Sim (Qo)	HeartBeat_B=[8,8]
Não	Sim	437	Sim	Quantidade de Transições não vivas: 12	Sim (Qo)	HeartB_A=[20,20]
Sim	Sim	13.471	Sim	Transições não vivas: HeartB_A e HeartB_B	Sim (Qo)	HeartB_A=[20,20] HeartB_B=[50,50]

Tabela 5.5 – *Resultados das Analises em três etapas da RdP da Figura 5.11.*

Conforme a tabela mostra, considerando que apenas o *endpoint* Cliente tem dados a enviar, foram encontradas 85 classes de estados. É uma rede limitada. No entanto, ela é não-viva**.** A rede é não viva porque 12 transições são potencialmente não-disparáveis a partir de determinadas classe de estados.

Quanto a propriedade de reinicição, de forma prática, sem a transição *HeartBeat,* observa-se no modelo que o *endpoint* cliente pode retornar para o estado **CLOSED** enquanto o *endpoint* servidor pode permanecer no estado **ESTABELECIDO**. Nesse caso, o servidor não tem nenhuma informação disponível para retornar para o estado inicial, **CLOSED**. Por isso a RdP agrega a transição denominada *HeartBeat* que, na prática, existe para evitar, justamente, que o sistema tenha um comportamento de não-reiniciação. Assim, mesmo sob falha, o sistema pode reiniciar seu processamento normal.

Considerando que apenas o servidor tenha dados a enviar, a rede tem comportamento semelhante ao anterior. Sendo que foram alcançadas 437 Classes de

Estados, parte do sistema ficou inoperável (o que é normal para o cenário verificado) e, para que a propriedade de reiniciação seja obtida, há necessidade do *HeartBeat*.

Na simulação considerando ambos os *endpoint* tendo dados a enviar, foram encontradas 13.471 Classes de Estados. Devido ao mecanismo de contador de retransmissão reiniciar o sistema antes do mecanismo *HeartBeat*, esses ficaram sem funcinalidade na RdP, para a janela de disparo definida..

5.7 RdP e Análise da Fase de Desassociação

A RdP da fase de desassociação é apresentada na Figura 5.12. Trata-se de um *Three-way handshake*.

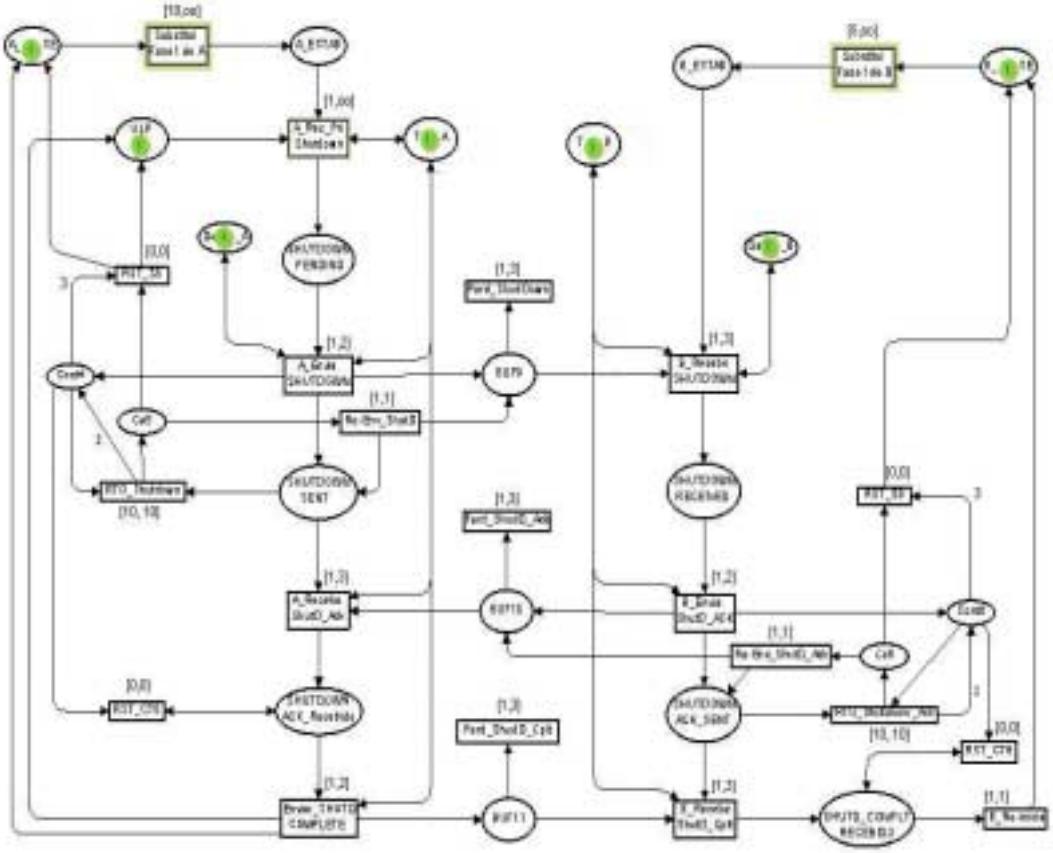


Figura 5.12 - *RdP da fase de desassociação SCTP*

A análise efetuada pelo PAREDE, nessa rede, mostra que ela tem 149 classes de estado, portanto é limitada, e também é viva e reinicializável, o que

demonstra a consistência dos eventos normais dessa fase para o nível de detalhes abordado.

5.8 RdP das Três Fases do Protocolo SCTP

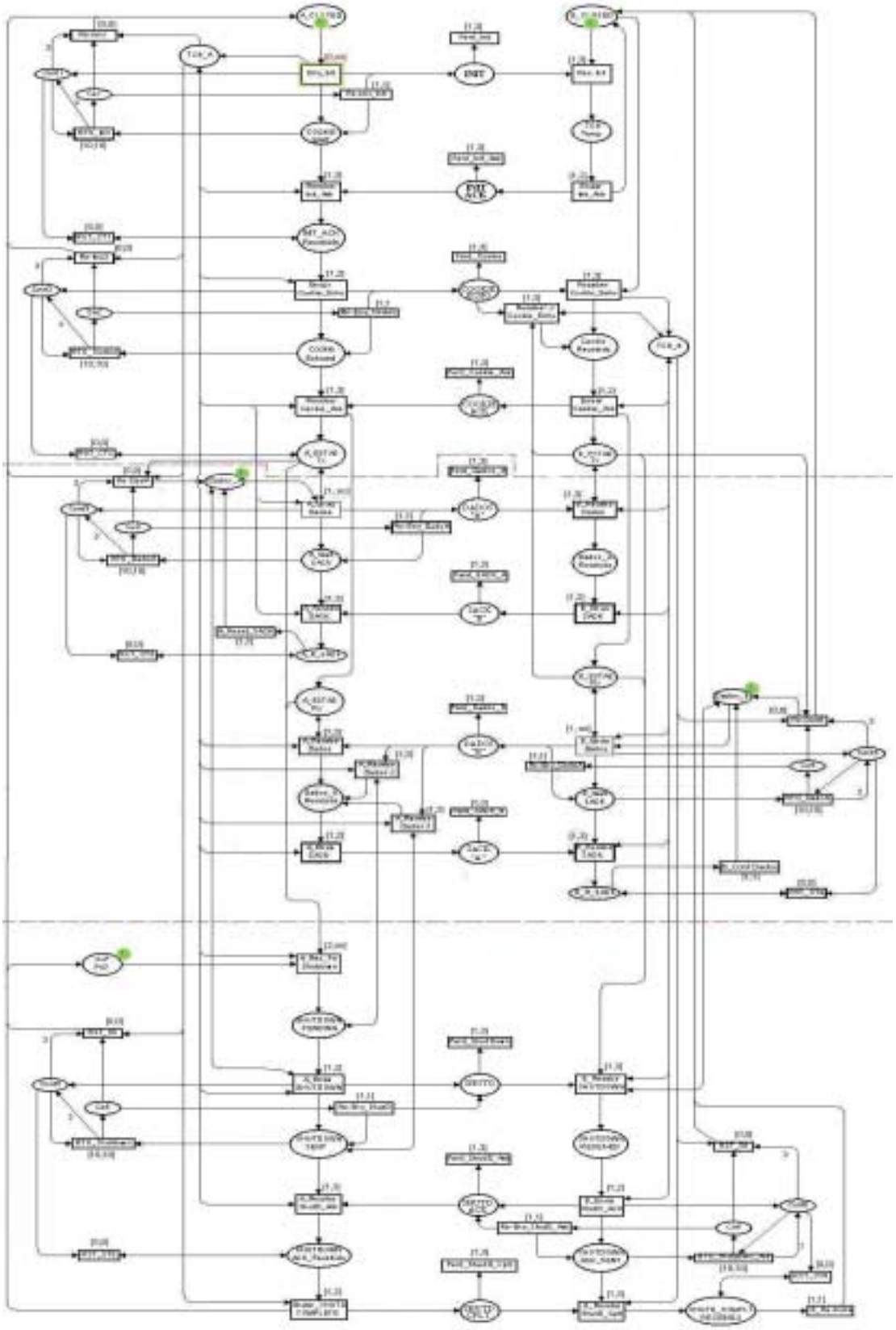


Figura 5.13 - *RdP das três fase do protocolo SCTP*

Essa RdP une as três fases do SCTP: o estabelecimento da associação, a troca de dados de usuários e a fase de desassociação, vistas anteriormente. A análise

dessa rede foi feita em duas etapas. Na primeira considerou-se não haver perdas de mensagens no sistema e o resultado é apresentado na tabela 5.6.

Dados_A	Dados_B	Qtd. Classes	Limitada	Viva	Reiniciável	Valores HeartBeats
Sim	Sim	202	Sim	Lista de Transições não vivas: Receber CookieEcho2, ...	Sim (Qo)	Sem HeartBeat

Tabela 5.6 – *Resultados da Analise da RdP das três fases do SCTP, sem perda de mensagens.*

Na segunda fase consideram-se as perdas e retransmissões de mensagens, Nesse caso, verificaram-se situações críticas relacionadas ao grande paralelismo no instante em que ambos os *endpoints* estão no estado ESTABELECIDO: os eventos de enviar dados, receber dados e receber primitiva *shutdown* são possíveis e a ocorrência de um desses eventos afeta diretamente os demais, podendo ocorrer falta de sincronismo sistêmico nesses instantes, sendo necessária uma configuração adequada das janelas de tempo dos eventos (sincronismo) para minimizar a explosão de estados no sistema.

O PAREDE não chegou à quantidade de classes de estados devido a pequena disponibilidade de memória no PC em que o PAREDE funcionou, limitando o processamento a cerca de 40.000 classes de estados. O PAREDE não rodou em máquinas mais rápidas, com S.O. Windows. Problema: *Runtime error* 200 *at* 0006:3996.

Esse resultado mostra que a RdP das três fases precisa ser melhor elaborada de forma a se poder obter as propriedades do SCTP neste nível de abstração e validar tais especificações do SCTP. Além disso, alguns outros eventos no sistema ainda precisam ser verificados, como é o caso de um *endpoint* tornar-se inalcançável (por exemplo, por desligamento do *host*). Isso pode ocorrer a qualquer instante, a partir de qualquer classe de estado e, após o retorno do *endpoint* à operação, o protocolo deve continuar a comunicação a partir do estado interrompido. Outra situação do

protocolo que precisa ser modelada e verificada é quanto ao ataque à segurança da associação, quando os *hosts* são *multihomed*.

Pode-se perceber que os modelos de RdP P/T começam a tornar-se muito grandes, necessitando de abstrações maiores (recursos de hierarquia). Emprega-se em seguida as CPNs para a continuação do estudo formal do protocolo.

5.9 Especificações do Protocolo SCTP em CPN

A CPN desenvolvida segue a especificação da RFC 2960, permitindo a checagem das principais fases do SCTP em um nível significante de detalhes para conduzir a uma análise do seu comportamento funcional, incluindo a modelagem do cabeçalho do pacote SCTP e das mensagens (*chunks*), (por exemplo, endereços, número de seqüências, etc) e a descrição das atualizações das variáveis de estado TCB.

Os campos do cabeçalho comum do pacote SCTP usados no modelo CPN incluem: Endereço fonte, endereço destino e Verification Tag (Ver_Tag), que especifica um valor gerado pelo enviador para validar seus pacotes na recepção. O campo *Checksum* é deixado de fora nesta modelagem. O cabeçalho das mensagens é formado basicamente pelos campos de tipo de mensagem e propriedades das mensagens. Os tipos de mensagens variam conforme Tabela 3.1 e as propriedades das mensagens dependem do tipo de mensagem.

Para a mensagem INIT (tipo 1), o campo utilizado no modelo é o **Initiate Tag**, que serve para o controle do receptor sobre os pacotes SCTP recebidos. Caso o valor do *Initiate Tag* mude após o recebimento do INIT, os pacotes são tratados como errados, a associação é encerrada e é enviado um ABORT. Os demais campos não são considerados no modelo nem os campos opcionais.

Para a mensagem INIT_ACK (tipo 2), que é utilizada para reconhecimento da iniciação de uma associação SCTP (do INIT), o campo utilizado é o mesmo do INIT mais um campo chamado *State Cookie*. Esse parâmetro é composto por um conjunto de valores necessários para que o enviador do INIT_ACK possa criar a associação. Nesse conjunto de valores, tem-se o MAC – Código de Autenticação de mensagem e a identificação do *endpoint* que gera o INIT_ACK.

Na mensagem COOKIE_ECHO (tipo10), os campos que a compõe são o tipo de mensagem e o campo *Cookie*, cujo valor deve ser uma cópia do *State Cookie*. Na mensagem COOKIE_ACK (tipo 11), há apenas o campo tipo de mensagem. É usado para reconhecer o recebimento do COOKIE_ECHO.

Da mensagem SHUTDOWN (tipo 7), utiliza-se apenas o campo de tipo de mensagem.. Da mensagem SHUTDOWN_ACK (tipo 8), utiliza-se apenas o campo de tipo de mensagem. E, da mensagem SHUTDOWN_COMPLETE (tipo 14), são utilizados os campos de tipo de mensagem e o campo T, para informar se o enviador tem ou não um TCB destruído.

Da mensagem *DATA* (tipo 0), são considerados os campos de Tipo e TSN para controle das mensagens de dados enviadas. Da mensagem SACK (tipo 3), enviada pelo receptor da mensagem de dados para informar sobre mensagens de dados recebidas, os campos utilizados são o de tipo de mensagem e o *Cumulative TSN Ack*, a fim de confirmar as mensagens de dados recebidas.

Essas são as mensagens (*chunks)* inicialmente consideradas para a modelagem do cenário básico, dos eventos normais do SCTP e eventos não normais da associação. Posteriormente, serão descritas outras mensagens utilizadas em outros eventos do SCTP e tratadas por ele. Dessa forma são modelados os pacotes SCTP num nível de abstração necessário para análise funcional.

Também é utilizado no modelo um registro denominado TCB (Bloco de controle de transmissão), que é uma estrutura de dados interna criada por um

endpoint SCTP para cada associação SCTP existente com o *endpoint* SCTP par. Serve para manter toda informação operacional do estado da associação, permitindo ao *endpoint* manter e gerenciar a associação correspondente. No modelo, as variáveis que compõem o TCB são: Estado do *Endpoint*, Origem, Destino, *V_Tag*, Tipo de mensagem (TC), *I_TAG* e *State Cookie*, composto de MAC e *EndP*.

Comandos de usuários (ULP), primitivas, também são utilizados no modelo. No modelo CPN usado aqui são empregadas as primitivas *Initialize*, *Associate, Close, ShutDown, Abort e Send*. Os procedimentos de iniciação, troca de dados e terminação foram descritos no capítulo III e modelados em alto nível nas RdP sem cor, anteriores, onde o comportamento essencial do protocolo nessas fases é descrito pela mudança de estados do SCTP, de acordo com os eventos de entradas e saídas. Para uma descrição completa ver [Stew00], seção 4 e 5.

Para essa rede CPN são feitas as seguintes **considerações,** por questão de limitações de recursos da ferramenta:

1. O canal de comunicação não perde, não corrompe, nem duplica pacotes SCTP, mas pode atrasar pacotes. Duas razões são apresentadas para iniciar a modelagem com um canal sem perda: uma é que um canal com perda pode mascarar qualquer bloqueio no protocolo; outra é que o Design/CPN não analisa redes temporizadas.

2. Não há retransmissões.

3. Considerou-se uma única instância de uma associação SCTP. Uma associação SCTP é identificada por um endereço IP local, um número de porta local, um endereço IP remoto e um número de porta remota. Uma única instância significa que a associação não é reusada.

4. Considera-se, que o *Buffer* do receptor é grande o suficiente para armazenar todos os pacotes que chegam.

5. O usuário emite comandos para o *endpoint* SCTP e recebe respostas. A razão é que devemos capturar as partes essenciais do SCTP para análise de certas propriedades (ex., ausência de bloqueios).

5.9.1 Arquitetura da CPN

A arquitetura inicial do modelo aproveita as vantagens do instanciamento de páginas [Jens92], que reduz o número de páginas (módulos) CPN. A arquitetura torna mais fácil manter e modificar o modelo CPN. Um outro beneficio é a redução de tempo necessário para checagem da sintaxe e mudanças no Design/CPN.

A CPN é formada por 7 lugares e 29 transições. Ela é organizada em três níveis hierárquicos, como é mostrado na Figura 5.14.

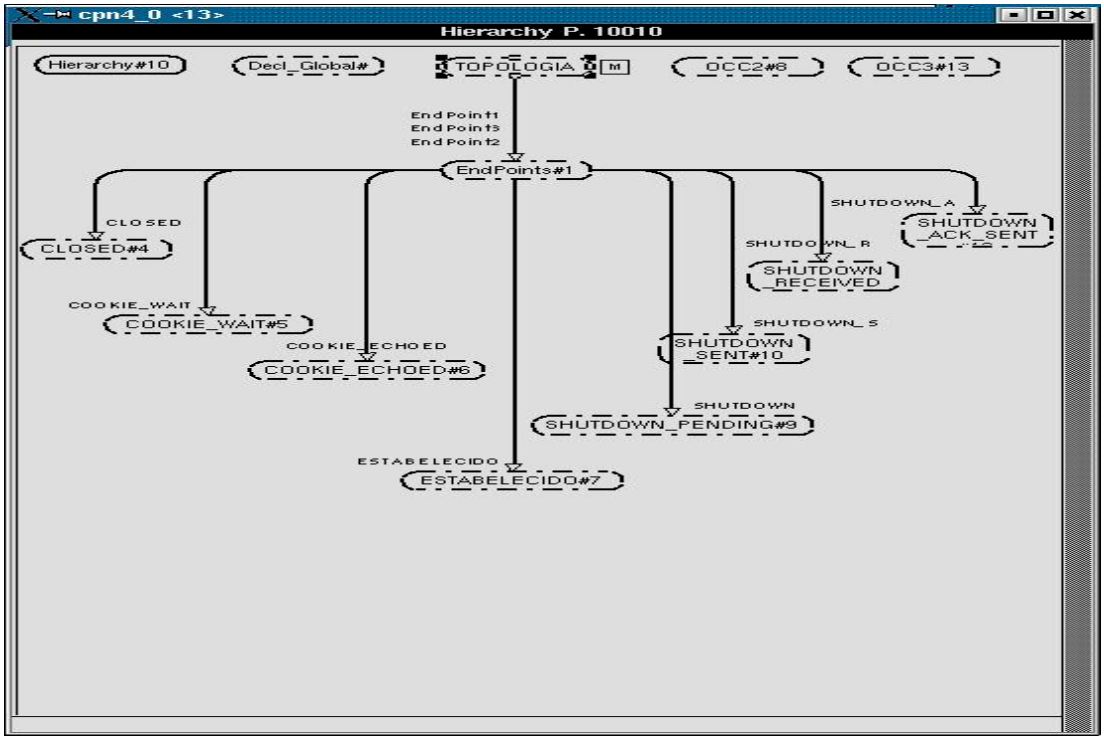


Figura 5.14 – *Hierarquia de páginas da CPN*

O primeiro nível tem uma página nomeada de topologia. O segundo nível tem uma página chamada de *endpoint*. O terceiro nível tem 8 páginas, cada uma é nomeada pelo estado parcial do SCTP.

Assim, descreve-se o comportamento do SCTP para cada estado, o que é consistente com a forma com que o SCTP é especificado na seção 4 da RFC 2960 [Stew00].

5.9.2 Declarações Globais

As declarações globais, como mostrado na Figura 5.15, incluem os conjuntos de cores e variáveis para cabeçalho SCTP, *chunk*s, TCB, comandos ULP (primitivas), e respostas do SCTP para ULP.

Figura 5.15 – *Declarações Globais*

O conjunto de cor HC, na linha 2, define o cabeçalho do SCTP com os campos EndO, EndD e VTag, explicados na seção 5.9. Os campos EndO e EndD podem assumir os valores definidos na linha 1. O VTag é definido como um inteiro. Observar que, apesar do espaço de número de seqüência e número de reconhecimento ser finito, podendo variar de 0 a 2^{32} -1, não foi implementada esta "geração aleatória" deste valor, no modelo, portanto o valor de ITag é estipulado em cada *endpoint*. Módulos aritméticos não são necessários para o procedimento de gerenciamento da associação, pois somente uma pequena quantidade de números de seqüências é necessária. O conjunto de cor dos *chunks* de controle são definidos nas linhas 5 a 17. Na linha 25 é declarada a estrutura do pacote SCTP, definido com a mesma estrutura do INIT.

O bloco de controle de transmissão TCB (linha 23) é definido como um produto do conjunto de cor STATE (linha 20) e V_ESTADO (linha 21), que modelam o estado do SCTP e as variáveis de estado, respectivamente. O conjunto de cor V_ESTADO define as variáveis explicadas na seção 5.9: EndO, EndD, VTag_Env, VTag_Rec, ITag_Env, ITag_Rec, OS, IS, TSN e TSN_Ack. A variável v (linha 22) e a variável para o registro das variáveis SCTP podem assumir qualquer valor pertencente a V_ESTADO.

O conjunto de cor PRI (linha 19) define os comandos que são enviados para a entidade SCTP de seus usuários. O conjunto de cor que define as respostas enviadas pelo *endpoint* aos usuários ULP, não foi declarado.

As funções associando valores aos pacotes SCTP/*Chunks* são mostradas nas linhas 24 – 36. Utiliza-se a função PAC_Init que modela o pacote INIT, como um exemplo para explicar a associação. O número de seqüência inicial, especificado em ML como #ITAG_ENV(v), é associado com o Init_Tag do *chunk* INIT e ITSN do *chunk* DATA. O VTag é iniciado com valor 0, porque no *chunk* INIT esse valor tem que ser zero, conforme especificado em [Stew00], seção 5.1. Outros *chunks* SCTP são modelados de forma similar.

5.9.3 As páginas Principal e de nível 2

A página principal, denominada de topologia e mostrada na Figura 5.16, provê uma visão abstrata em alto nível de uma arquitetura de comunicação do protocolo.

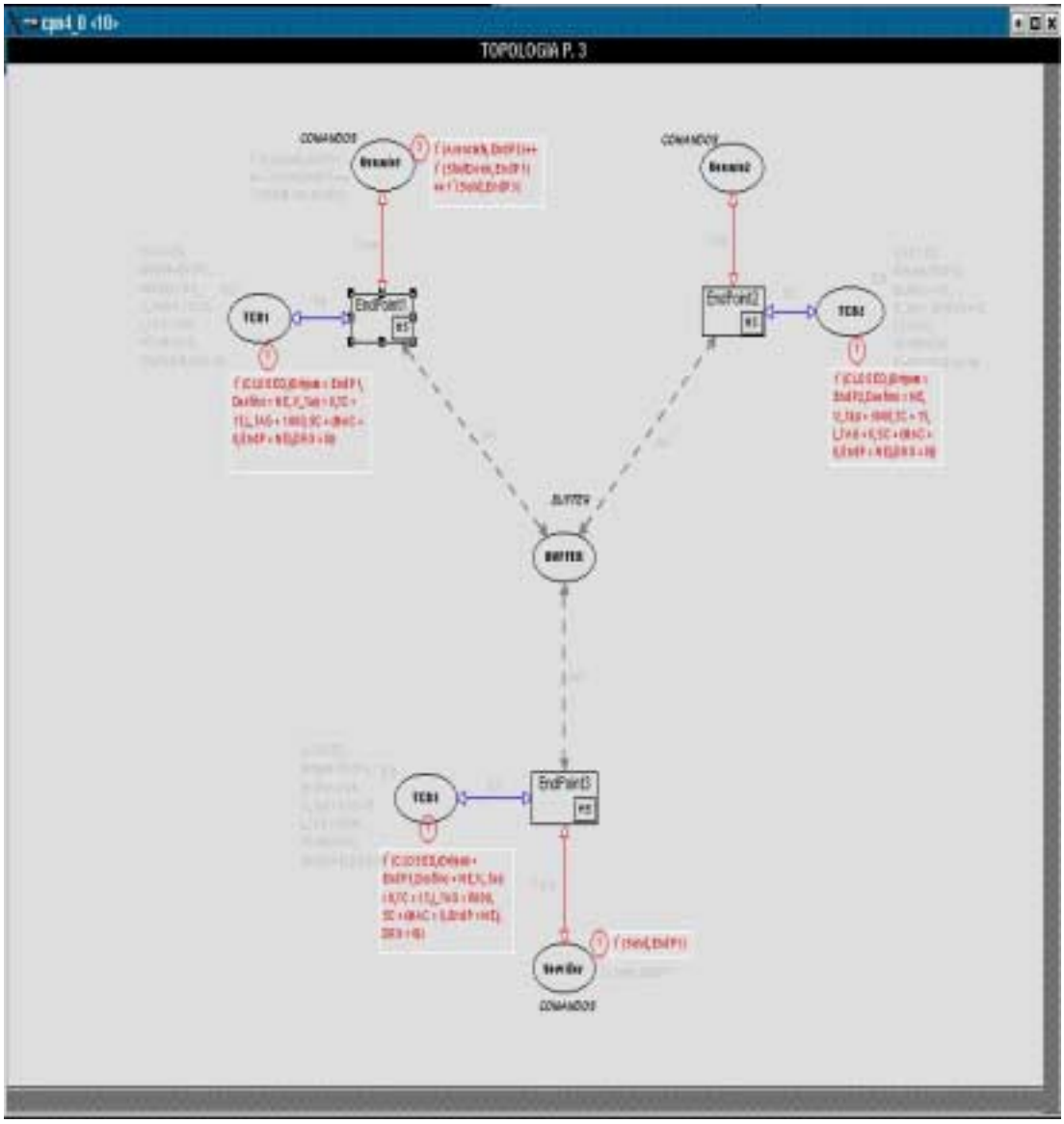


Figura 5.16 – *Nível superior de página CPN: Visão geral da Topologia*

Existem 7 lugares na figura acima. Os lugares **Usuário1-3**, cujos conjuntos de cores associados é o **COMANDOS**, representando as primitivas de usuário SCTP. Mudando a marcação inicial de um lugar **Usuário** mudará o comando emitido para o *endpoint* SCTP, resultando em diferentes casos de modelagem. Os lugares **TCB1-3** do tipo **TCB** modelam as informações de estado dos *endpoints* SCTP.

Uma ficha nesse lugar representa seus estados. O lugar **BUFFER,** do tipo **BUFFER,** modela um canal de comunicação *fullduplex*, onde pacotes enviados de um *endpoint* a outro podem permanecer por algum tempo. Uma ficha nesse lugar, canal de comunicação, representa um segmento em trânsito de um *endpoint* ao seu *endpoint* par, e pode estar em qualquer lugar na rede ou em um buffer do *endpoint*. Há também, na Figura 5.16, três transições de substituição chamadas de **EndPoint1, EndPoint2** e **EndPoint3**, cada qual representando um *endpoint* com as funcionalidades SCTP, estabelecimento da associação, transmissão de dados e os procedimentos de término da associação.

A transição de substituição é uma transição criada para ser associada com uma outra página CPN (sub-página) que é realmente simulada, quando a transição de substituição executa. Os *EndPoints* são todos associados com a página de segundo nível (Figura 5.17), que serve como uma instância de página quando chamada por eles, alternadamente.

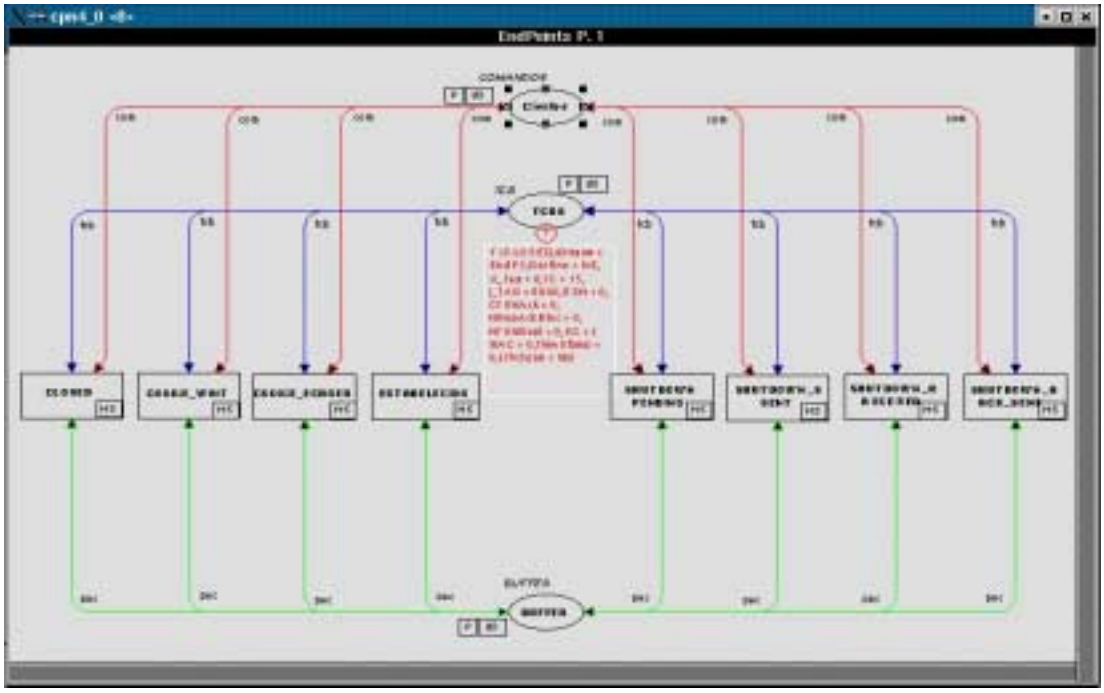


Figura 5. 17 – *Página do segundo nível CPN: EndPoint SCTP*

Os lugares associados com uma transição de substituição precisam ser associados a lugares com a mesma função na sub-página. Por exemplo, os lugares **Usuário1** e **Usuário2**, na Figura 5.16, são associados ao lugar **Clientes** na Figura

5.17 para os **EndPoint1** e **EndPoint2** respectivamente. O lugar ***BUFFER***, na Figura 5.16, é associado com o lugar ***BUFFER***, na figura 5.17, para **EndPoint1** e **EndPoint2**.

A página de segundo nível organiza os detalhes do processo de associação, transmissão de dados e desassociação em 8 transições de substituição. Cada transição é nomeada por um estado do *endpoint* e é associada com uma página no terceiro nível, que modela o comportamento do *endpoint* SCTP para este estado.

5.9.4 As oito Páginas do terceiro nível

Para cada estado do *Endpoint* é criada uma página. Portanto, têm-se as páginas, conforme Figura 5.17.

- **CLOSED;**
- **COOKIE_WAIT;**
- **COOKIE_ECHOED;**
- **ESTABELECIDO;**
- **SHUTDOWN_PENDING;**
- **SHUTDOWN_SENT;**
- **SHUTDOWN_RECEIVED e**
- **SHUTDOWN_ACK_SENT.**

Em seguida, são apresentadas todas as oito páginas do terceiro nível.

5.9.4.1 A página CLOSED

A página CLOSED (Figura 5.18) modela o comportamento do SCTP para o estado CLOSED. Ela tem 6 transições: Primitivas, Env_INIT, Recebe pac_Init,

Envia Init_Ack, Receber Cookie_Echo e enviar Cookie_Ack. E, 3 lugares que já foram descritos.

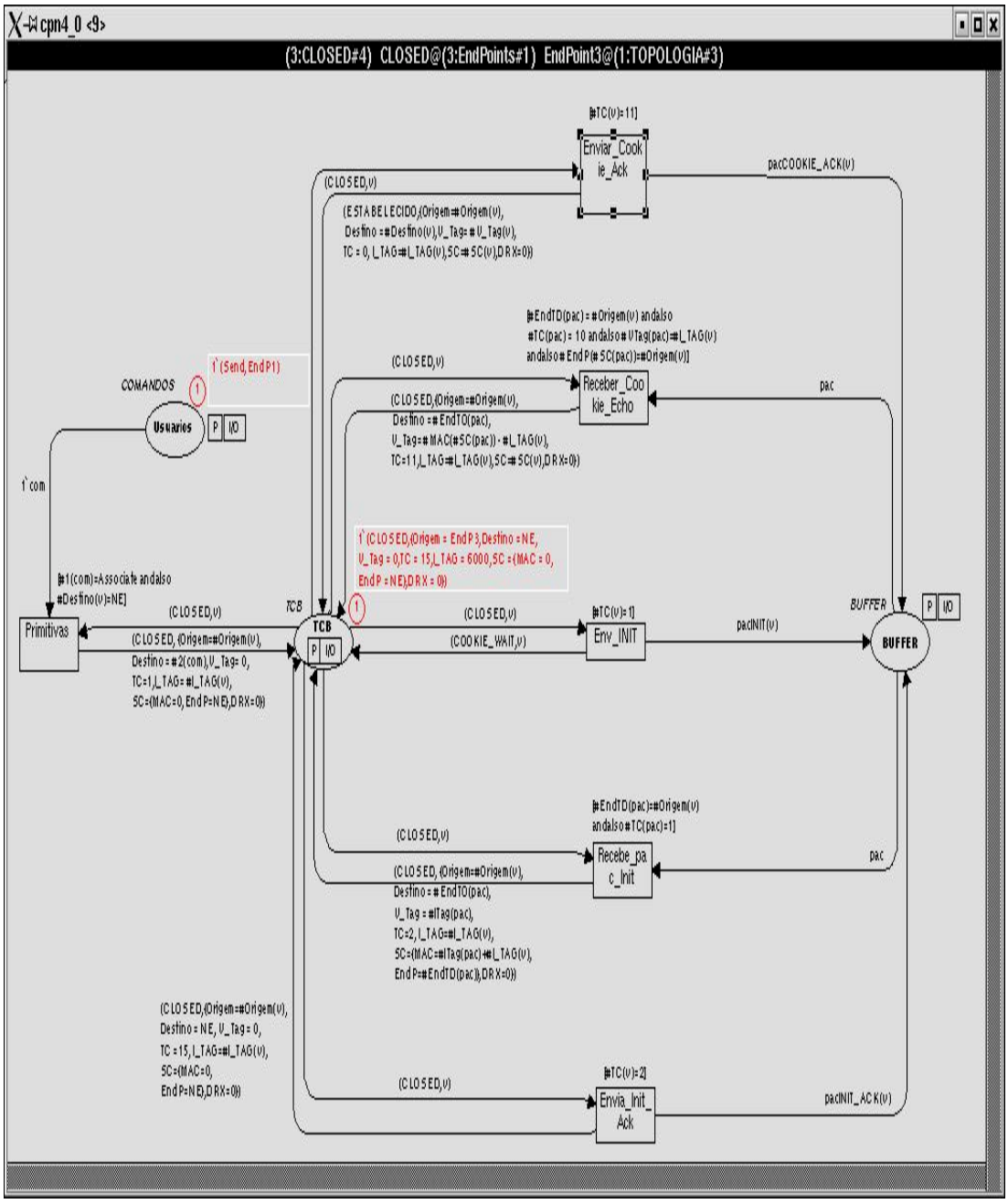


Figura 5.18 – *A página CLOSED*

A transição "**Primitivas"** modela o envio de uma primitiva do usuário para o *endpoint* SCTP. Essa primitiva contém as informações necessárias para que o *endpoint* atenda a solicitação do ULP. Como está indicado pela expressão do arco de saída, seu disparo gera as informações necessárias (TCB) para que o *endpoint* inicie uma associação.

As demais transições têm funções semelhantes às descritas nos modelos de RdP com temporização anteriores, com a diferença fundamental na manipulação das variáveis que as expressões dos arcos expressam. Todas as transições têm uma condição de guarda. Quando essas condições são satisfeitas, a transição pode ser habilitada. Esse comportamento é reproduzido em todas as páginas apresentadas.

5.9.4.2 A página COOKIE_WAIT

A página COOKIE_WAIT (Figura 5.19) modela o comportamento do SCTP para o estado COOKIE_WAIT. Ela tem 4 transições: Recebe Init_Ack, Envia_Cookie_Echo, Recebe_Init do ParEnd e enviar Init_Ack.

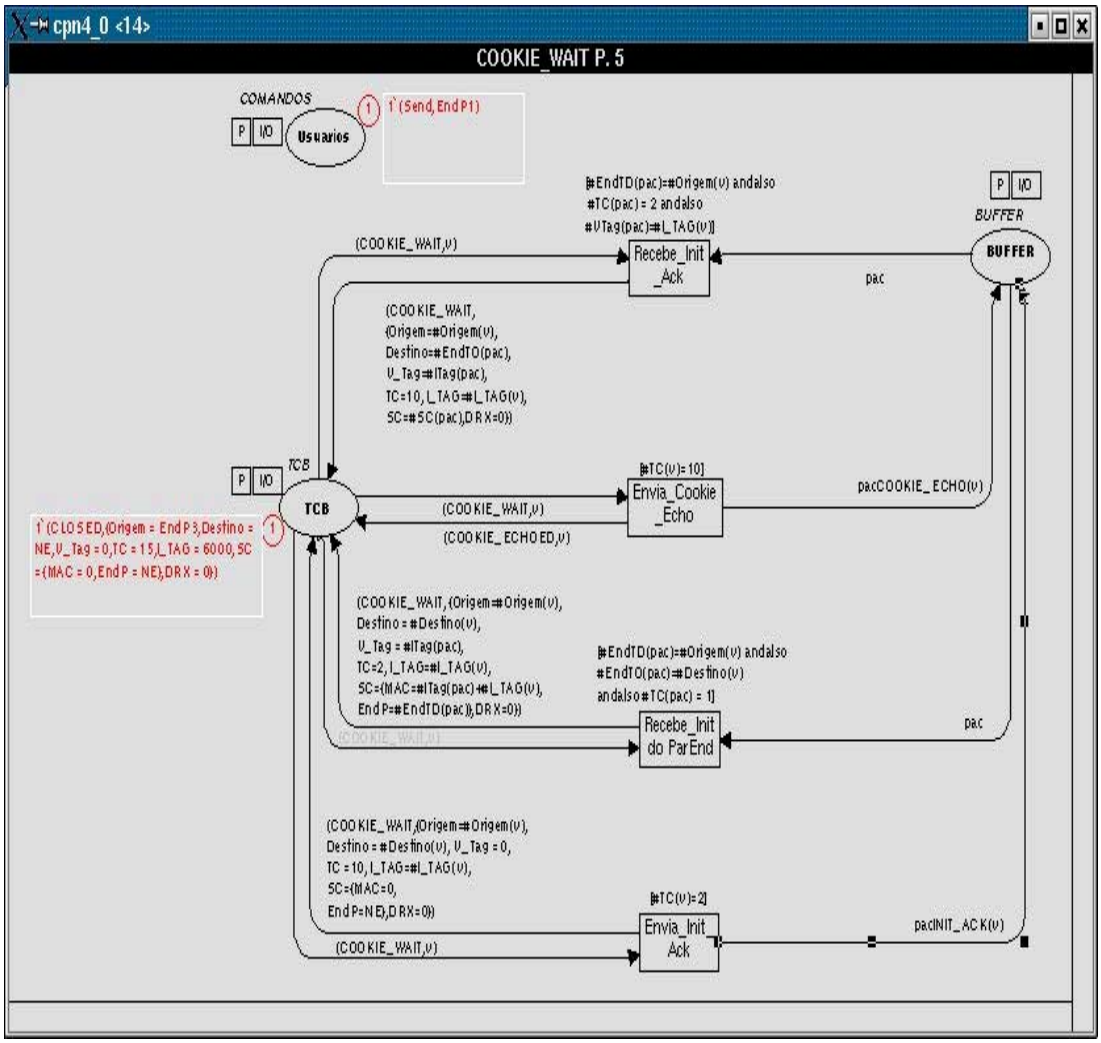


Figura 5.19 – *Página COOKIE_WAIT*

Conforme dito, a descrição semântica do modelo segue a que foi apresentada no modelo de RdP com temporização. A diferença está nas expressões dos arcos, que se pode ver na Figura 5.19 e nas atualizações das variáveis TCB.

5.9.4.3 A página COOKIE_ECHOED

A página COOKIE_ECHOED (Figura 5.20) modela o comportamento simplificado do SCTP para o estado COOKIE_ECHOED. Ela tem 1 transição: Receber Cookie_Ack e muda para o estado ESTABELECIDO.

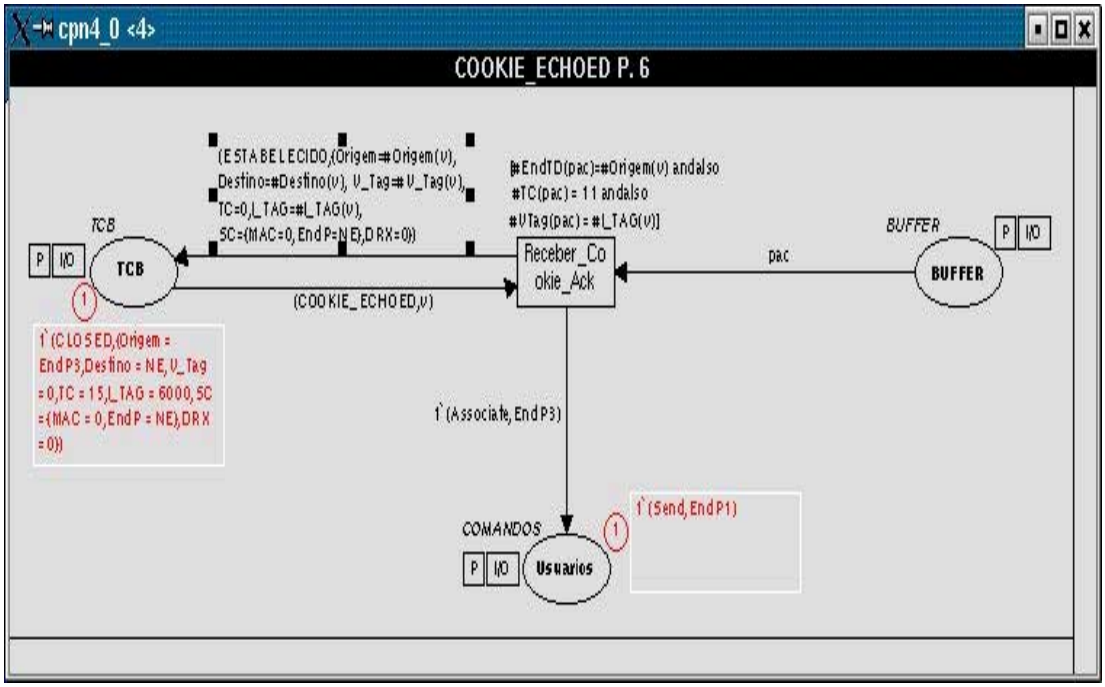


Figura 5.20 – *Página COOKIE_ECHOED*

5.9.4.4 A página ESTABELECIDO

A página ESTABELECIDO (Figura 5.21) modela o comportamento do SCTP para o estado ESTABELECIDO. Ela tem 6 transições, que também operam de acordo com as especificações da RFC2960 e utilizam a condições de guarda para

serem habilitadas. Podem, assim, enviar e receber dados de modo *fullduplex* e receber primitivas e/ou mensagens de ***shutdown***.

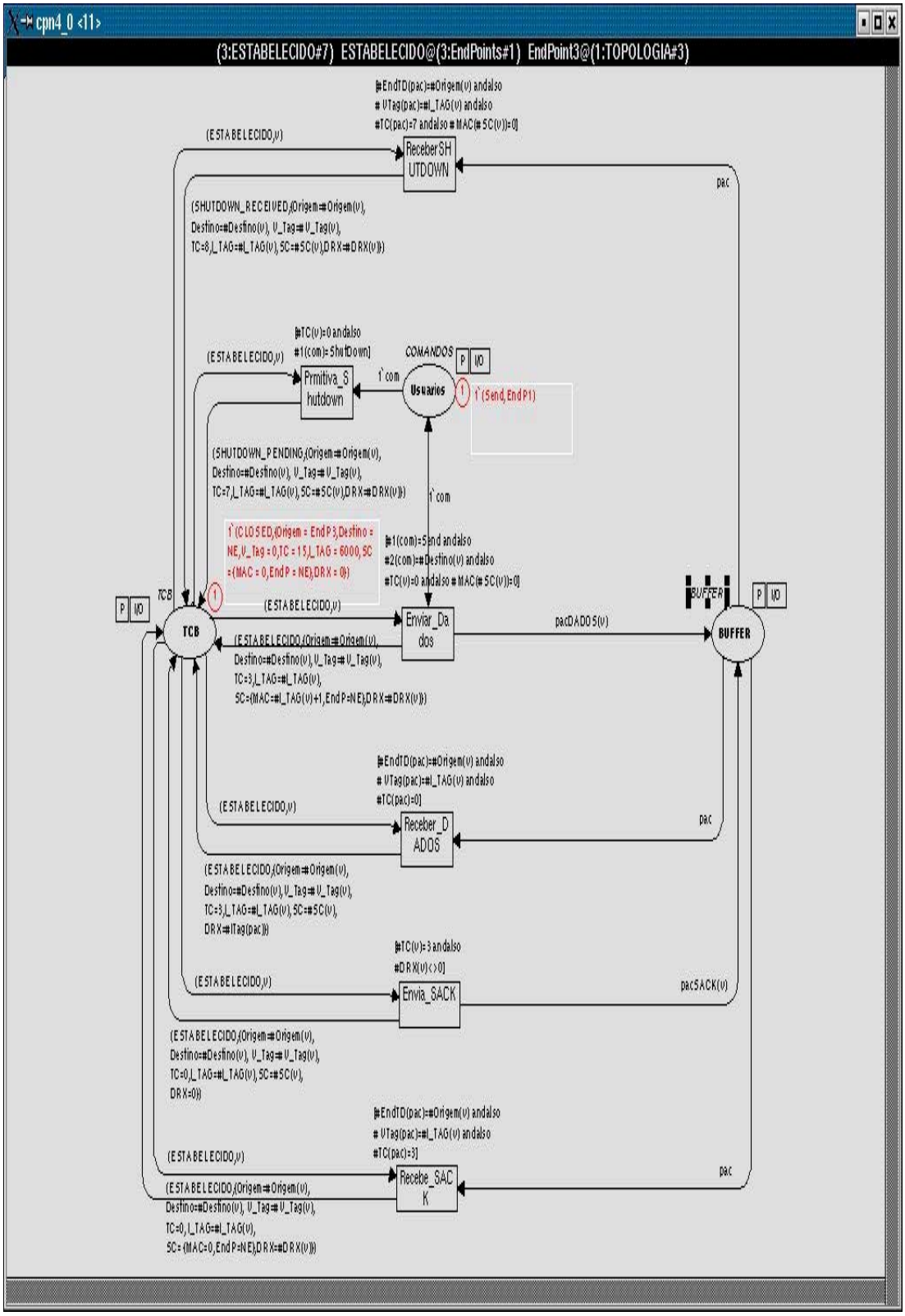


Figura 5.21 – *Página ESTABELECIDO*

5.9.4.5 A página SHUTDOWN_PENDING

Essa página (Figura 5.22) modela o comportamento do SCTP para o estado SHUTDOWN_PENDING. Ela tem 4 transições, que operam conforme especificado na RFC2960, para o caso do *endpoint* ser o originador do *shutdown* e ter ainda dados a enviar ou a confirmar.

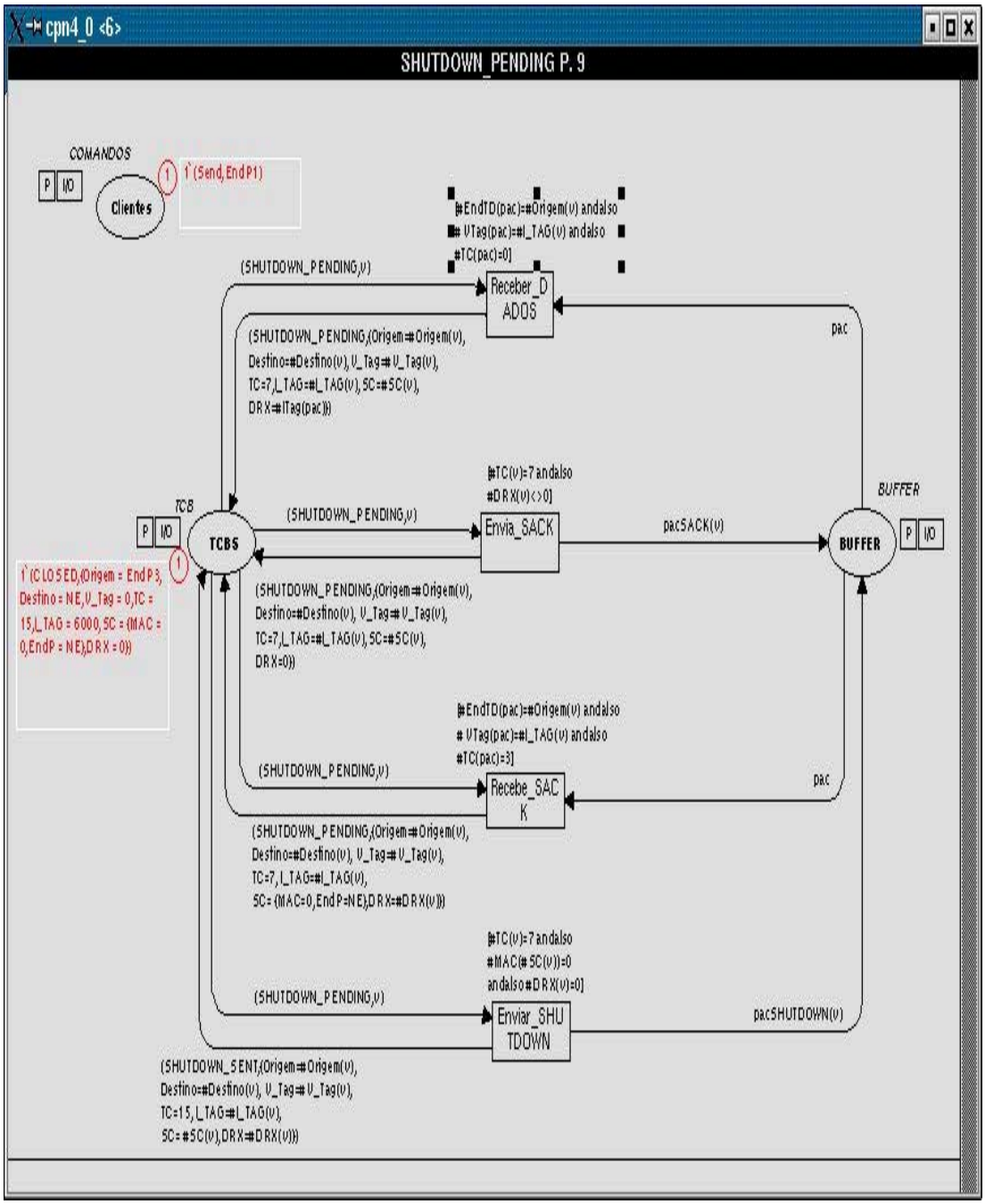


Figura 5.22 – *Página SHUTDOWN_PENDING*

5.9.4.6 A página SHUTDOWN_SENT

Essa página (Figura 5.23) modela o comportamento do SCTP para o estado SHUTDOWN_SENT. Ela tem 4 transições que operam conforme especificado na RFC2960, para o caso do *endpoint* ser o originador da mensagem *shutdown* e já a ter enviado ao destino.

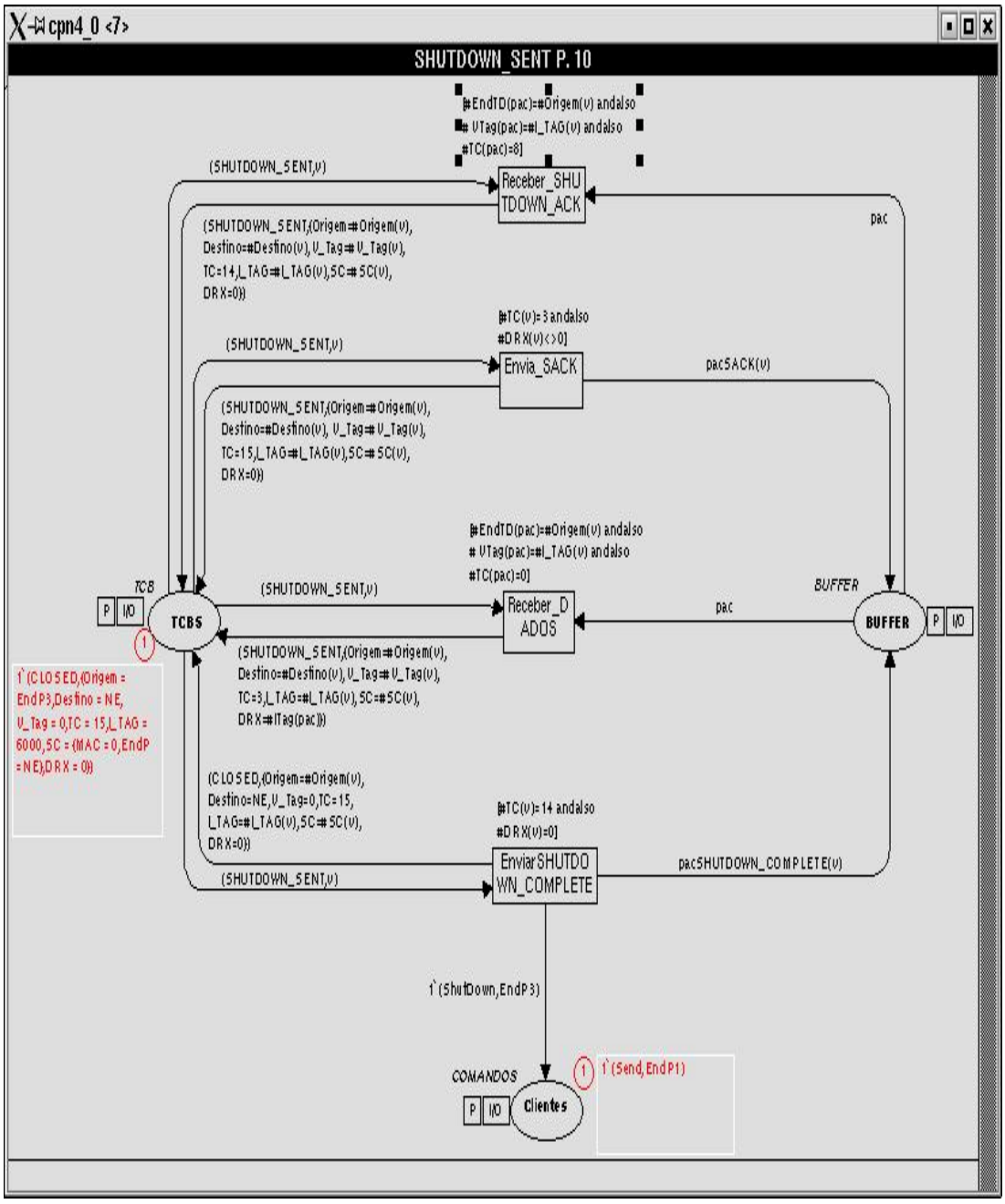


Figura 5.23 – *Página SHUTDOWN_SENT*

5.9.4.7 A página SHUTDOWN_RECEIVED

Essa página (Figura 5.24) modela o comportamento do SCTP para o estado SHUTDOWN_SENT. Ela tem 4 transições, que operam conforme especificado na RFC2960, para o caso do *endpoint* ser o recebedor da mensagem *shutdown*, confirmar todos os dados enviados/recebidos e enviar resposta à mensagem *shutdown*.

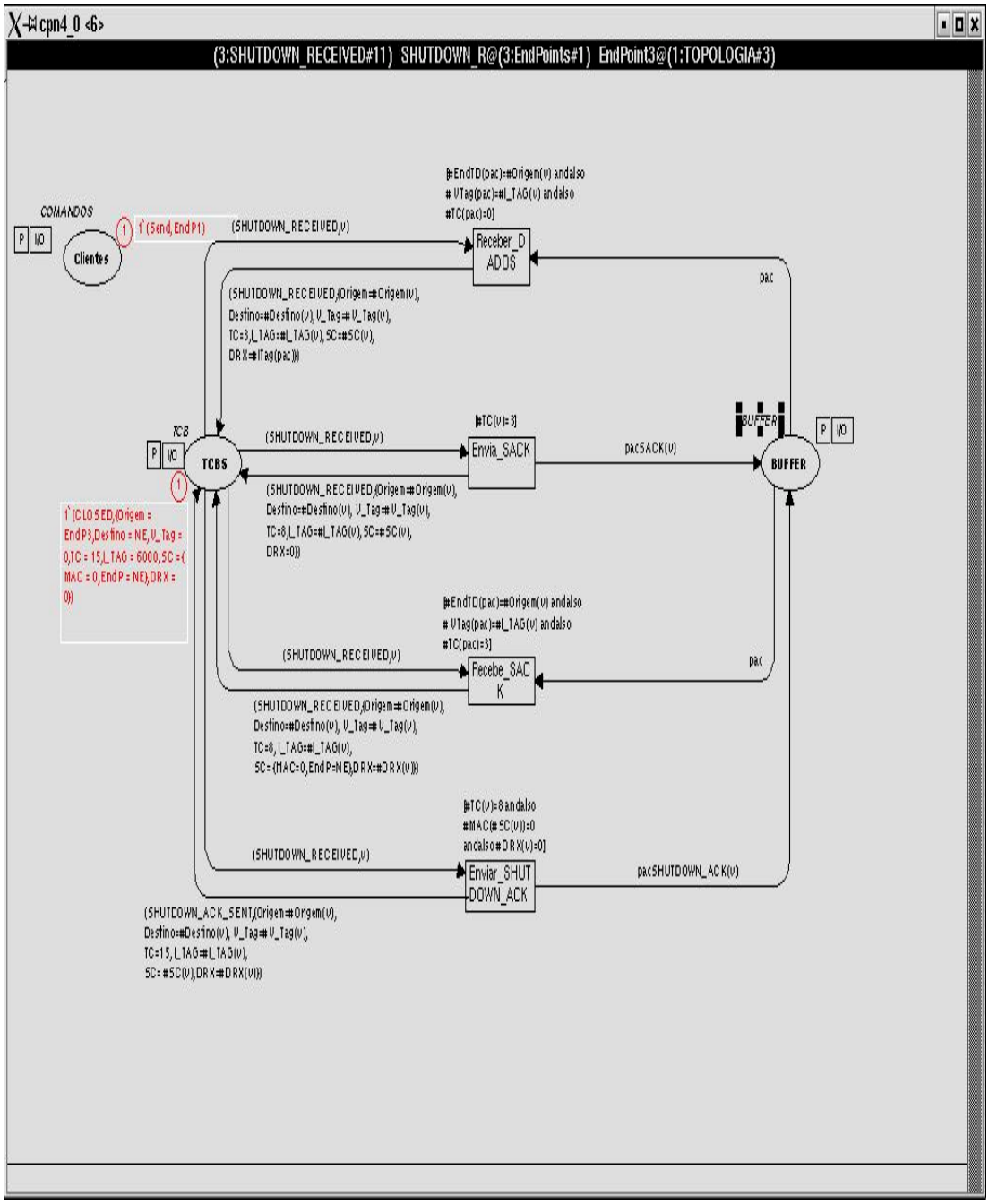


Figura 5.24 – *Página SHUTDOWN_RECEIVED*

5.9.4.8 A página SHUTDOWN_ACK_SENT

Essa página (Figura 5.25) modela o comportamento do SCTP para o estado SHUTDOWN_ACK_SENT. Ela tem 1 transição, que opera conforme especificado na RFC2960, para o caso do *endpoint* receber a mensagem *shutdown complete* e realizar a sua reiniciação definindo seu estado como CLOSED e iniciando as variáveis TCB.

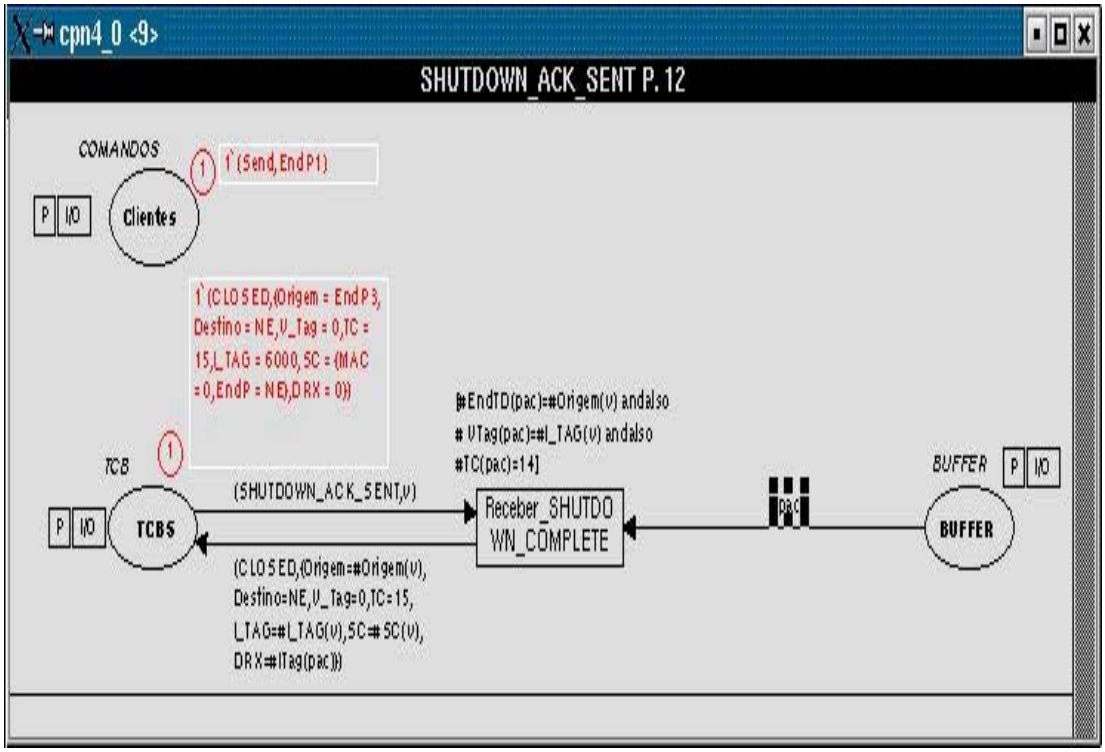


Figura 5.25 – *Página SHUTDOWN_ACK_SENT*

5.9.5 Análises da CPN para alguns Cenários

A CPN desenvolvida permite a verificação do protocolo para alguns cenários funcionais. Os cenários são estabelecidos pela marcação inicial dos *endpoints*.

5.9.5.1 Análise da CPN para o Cenário 1

O primeiro cenário é definido pela marcação inicial apresentada na Figura 5.16, onde um *endpoint* cliente inicia a associação com o servidor, que por sua vez valida o cliente e estabelece a associação. A partir do estabelecimento concluído, o cliente envia dados ao servidor que são confirmados por esse e, por fim, o cliente inicia o término normal da associação.

O grafo de classes de estados para esse primeiro cenário pode ser visto na Figura 5.26, que tem 21 marcações e 24 arcos (eventos).

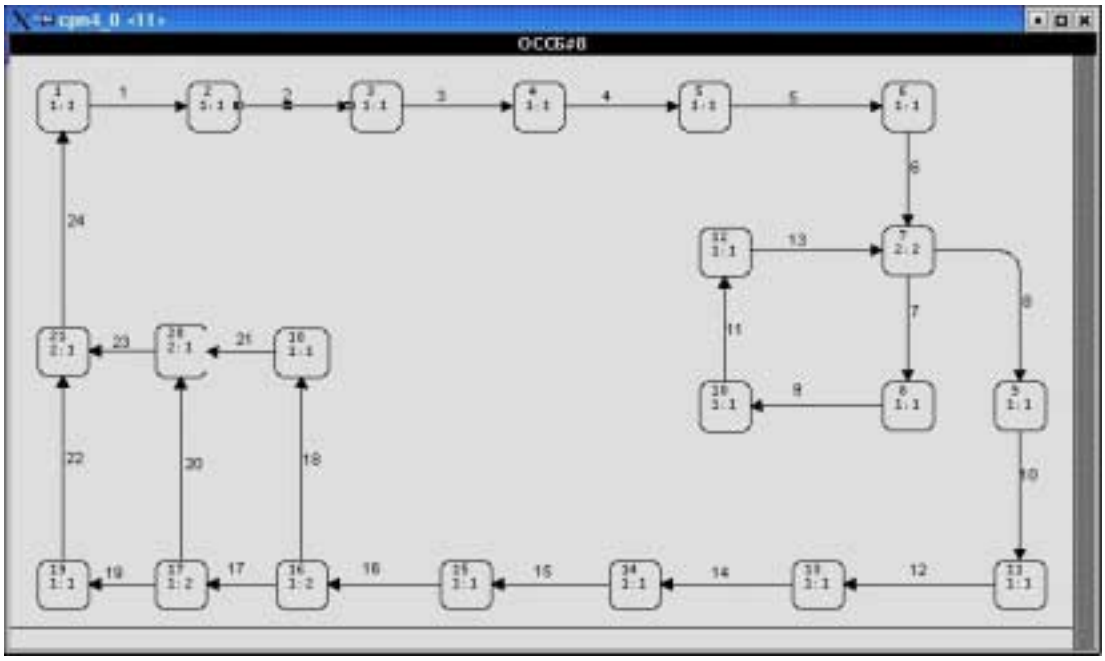


Figura 5.26 – *Grafo de Marcações do Modelo CPN para o Cenario 1*

O grafo da Figura 5.26 descreve os estados possíveis do SCTP para todas as três fases do protocolo, para o cenário 1, considerando as restrições descritas na seção 5.9.

Destaca-se que para cada mudança de valor de variáveis do sistema (TCB) tem-se uma marcação que a representa. O primeiro momento de paralelismo é evidenciado em μ_{10}, onde o sistema, no estado ESTABELECIDO, pode trocar dados e receber *shutdown*. O segundo paralelismo, μ_{19}, descreve a possibilidade de ambos

os *endpoints* retornarem ao estado inicial (CLOSED) e a possibilidade de, quando o *endpoint* servidor retornar para CLOSED, o *endpoint* cliente já estar com uma solicitação de associação (μ_2) ou já ter enviado uma mensagem INIT (μ_3). As Marcações são todas mostradas nas Figuras 5.27, 5.28 e 5.29.

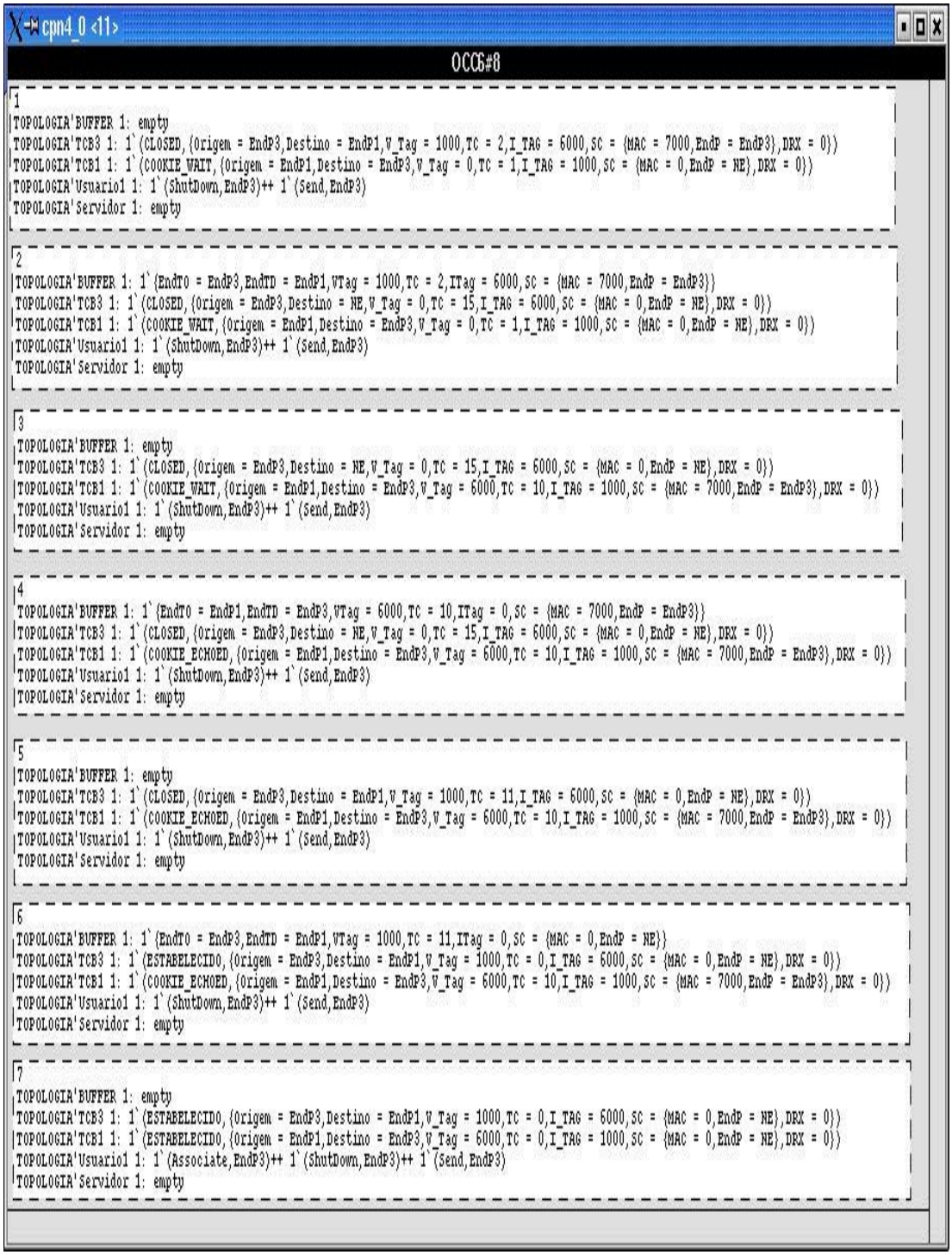


Figura 5.27 –*Marcações de 1 a 7 para o Modelo CPN no Cenario 1*

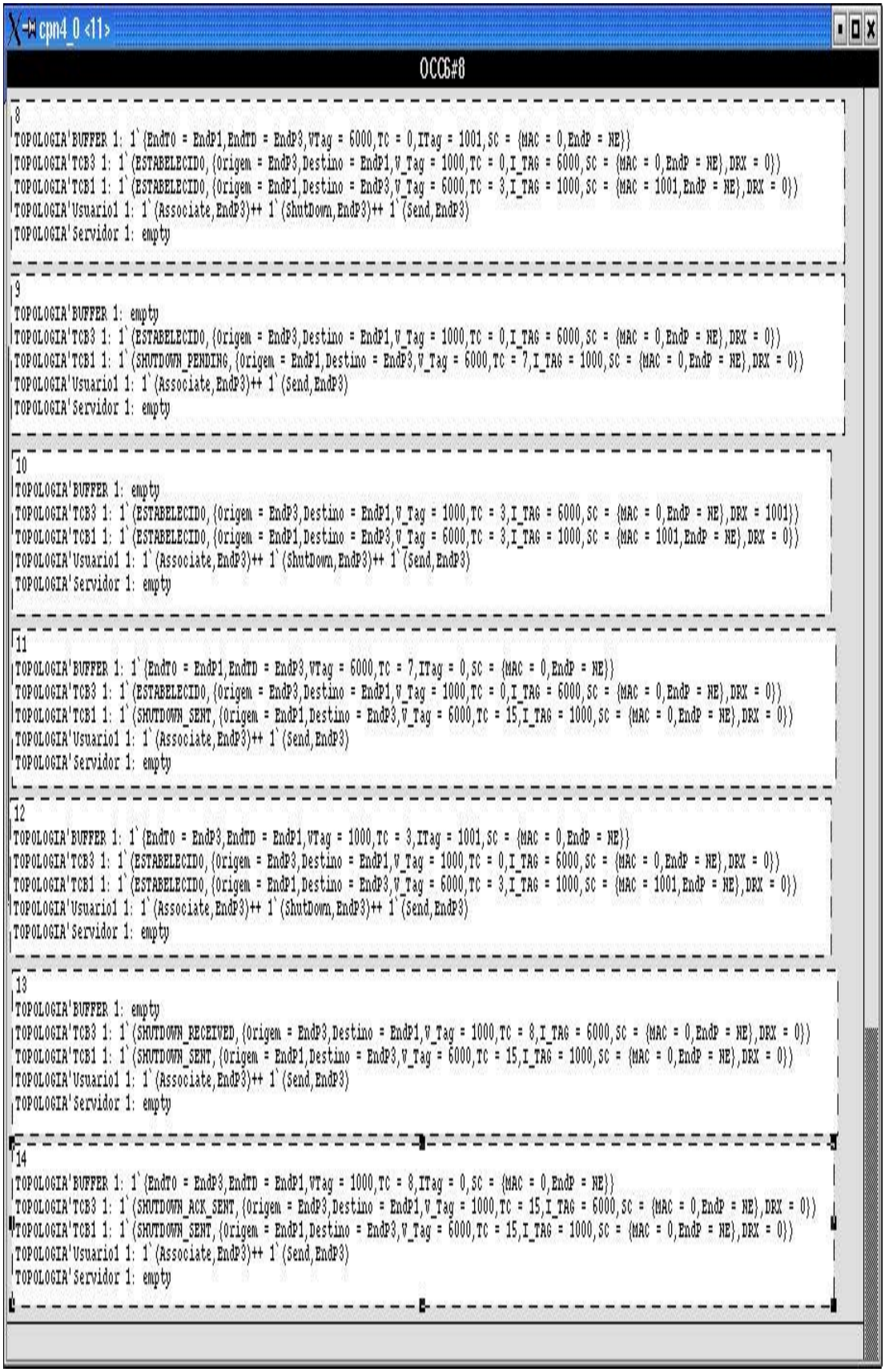


Figura 5.28 – *Marcações de 8 a 14 para o Modelo CPN no Cenario 1*

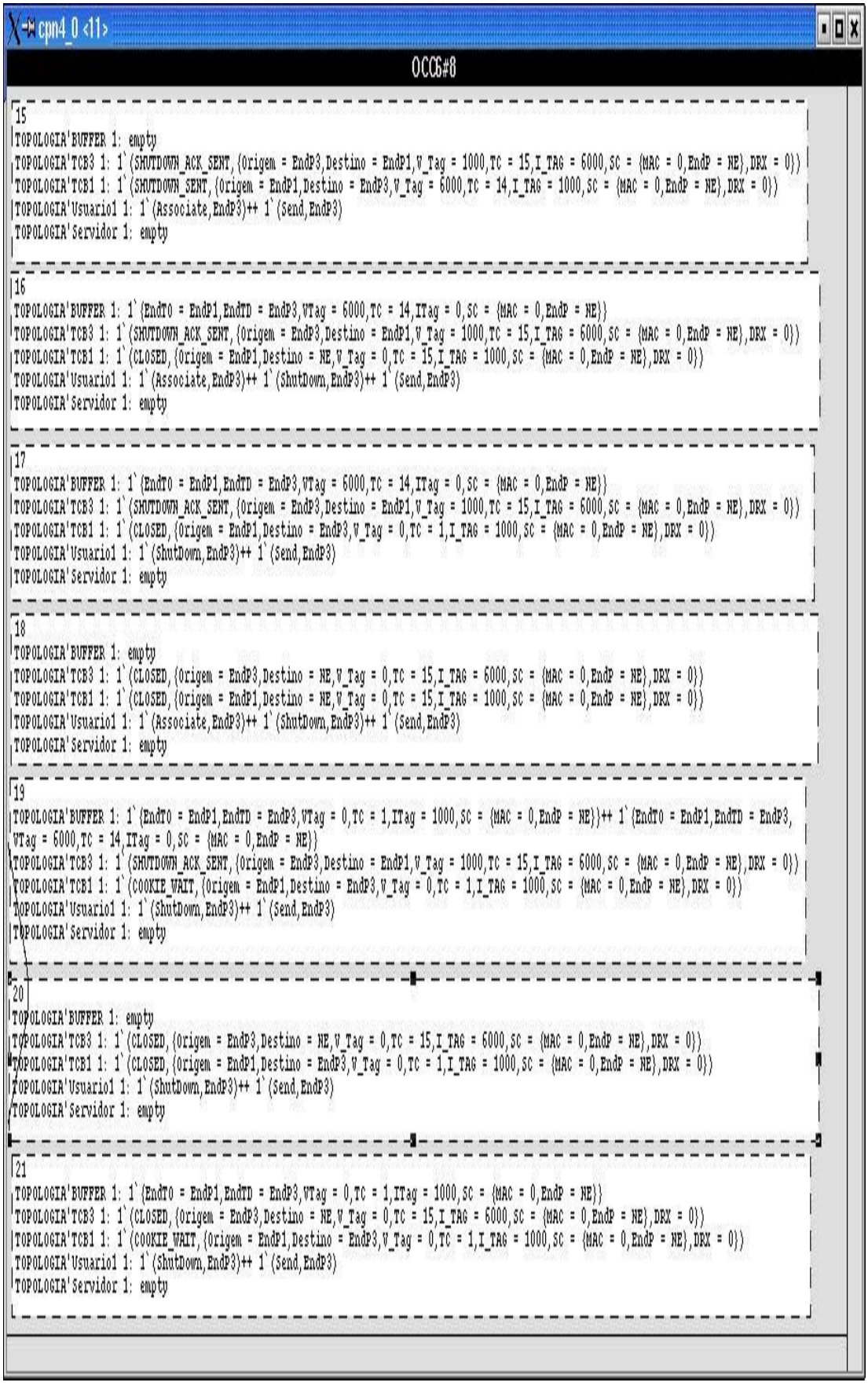


Figura 5.29 – *Marcações de 15 a 21 para o Modelo CPN no Cenario 1*

De modo geral, não há marcação estranha ou anormal. Ou seja, a quantidade de marcações é limitada e o sistema sempre pode retornar ao estado inicial. Essa inspeção confirma que, para esse cenário, o sistema é limitado, vivo e reiniciável.

As Figuras 5.30 e 5.31 apresentam as funções de arcos, conforme grafo da Figura 5.26.

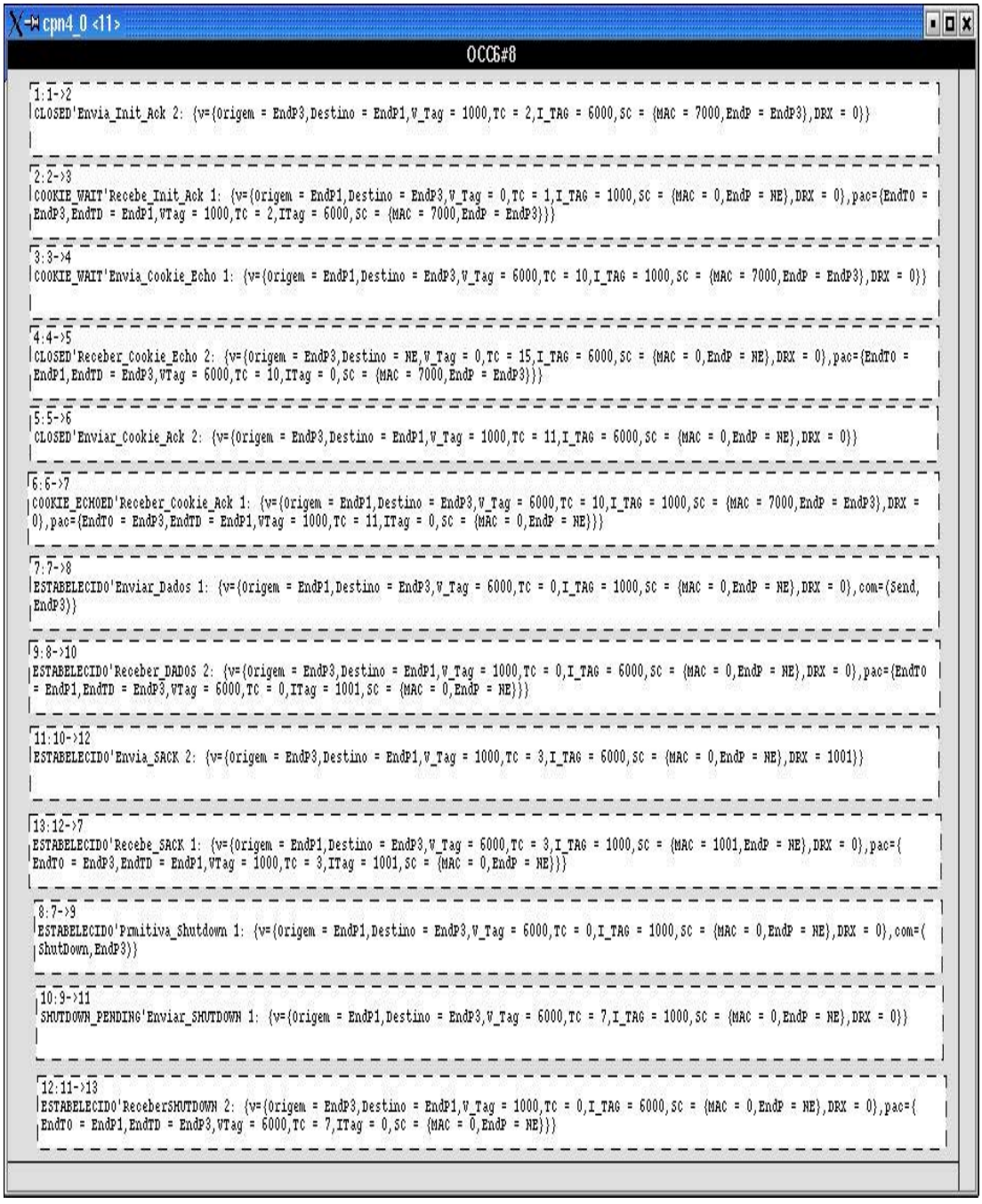


Figura 5.30 – *Funções dos Arcos 1 a 13 do Modelo CPN no Cenario 1*

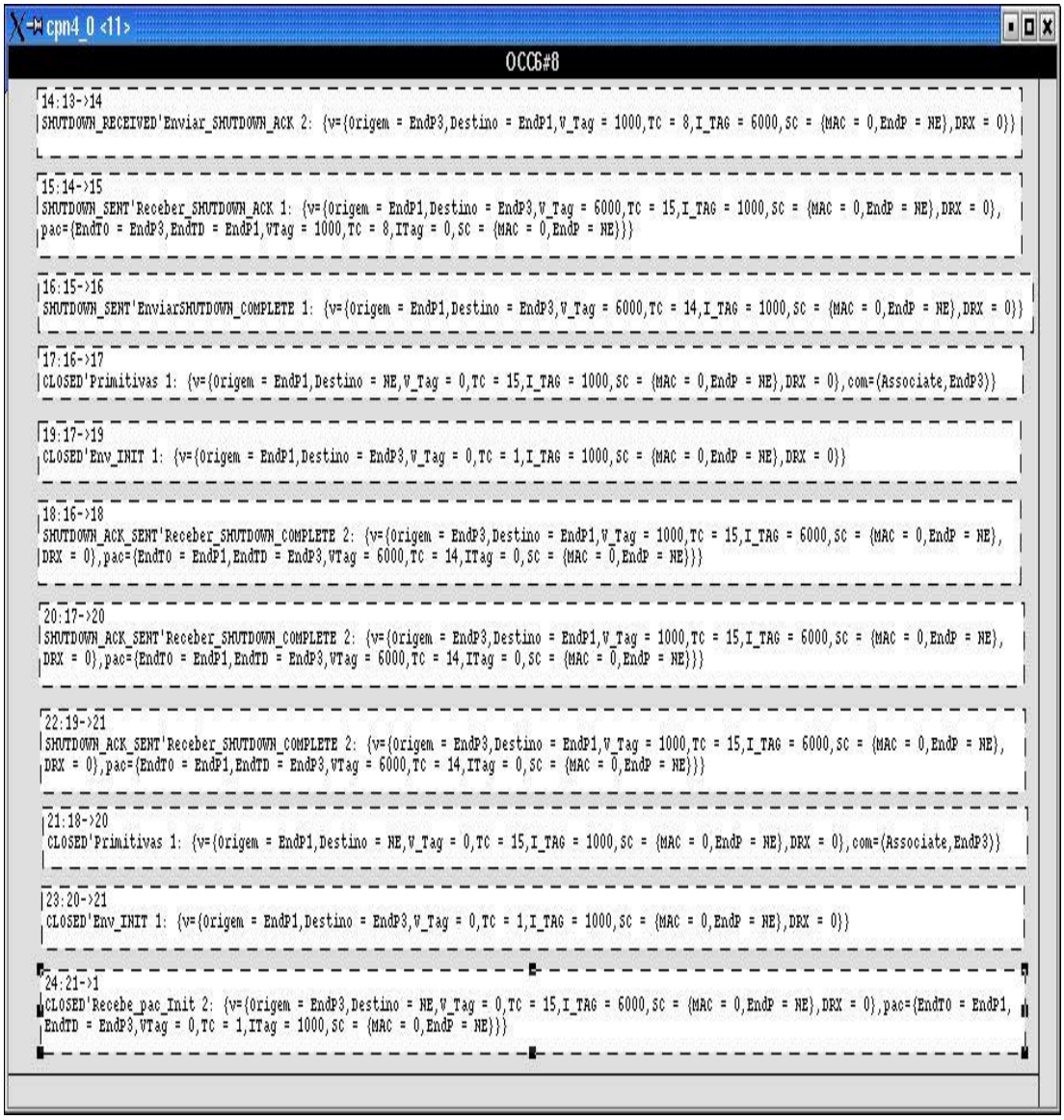


Figura 5.31 – *Funções dos Arcos 14 a 24 do Modelo CPN no Cenario 1*

5.9.5.2 Análise da CPN para o Cenário 2

O segundo cenário ao qual o modelo é investigado também pode ser visto na Figura 5.32. Nesse caso, além dos eventos do cenário anterior, após o estabelecimento da associação, ambos os endpoints realizam a troca de dados e a qualquer instante o cliente pode solicitar o término normal da associção.

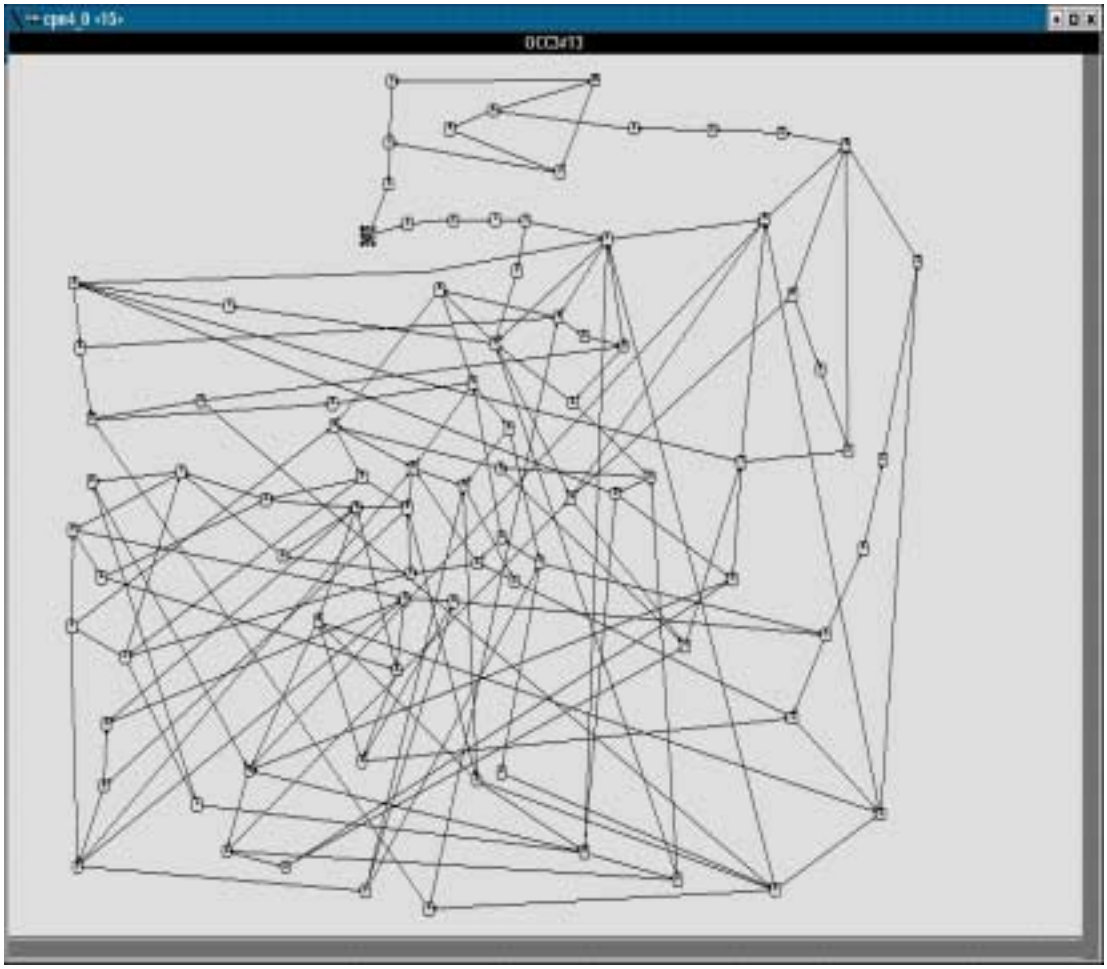


Figura 5.32 – *Grafo de marcações alcançáveis para o Cenário 2*

Conforme pode ser visto no grafo de marcações alcançáveis para esse cenário, que tem 87 nós e 163 arcos, também não há marcações com bloqueio e o sistema sempre pode retornar ao estado inicial. As propriedades encontradas indicam que o sistema é limitado, vivo e reiniciável, confirmando que o protocolo procede corretamente aos eventos desse segundo cenário.

5.9.5.3 Análise da CPN para o Cenário 3

O terceiro cenário ao qual o modelo é verificado é o de uma situação onde ambos os *endpoints* solicitam o estabelecimento da associação (igualmente à seção

5.4), realizam a troca de dados e, a qualquer instante, o Usuário1 solicita o término normal da associação.

O grafo de marcações alcançáveis, gerado pelo Design/CPN, é mostrado na Figura 5.33. Ele tem 234 nós e 343 arcos. Nesse caso, há marcações indesejáveis, mortas. Isto pode significar que, caso o sistema realize a seqüência de eventos que levam a essas marcações (em vermelho), o sistema entrará num estado de bloqueio (deadlock).

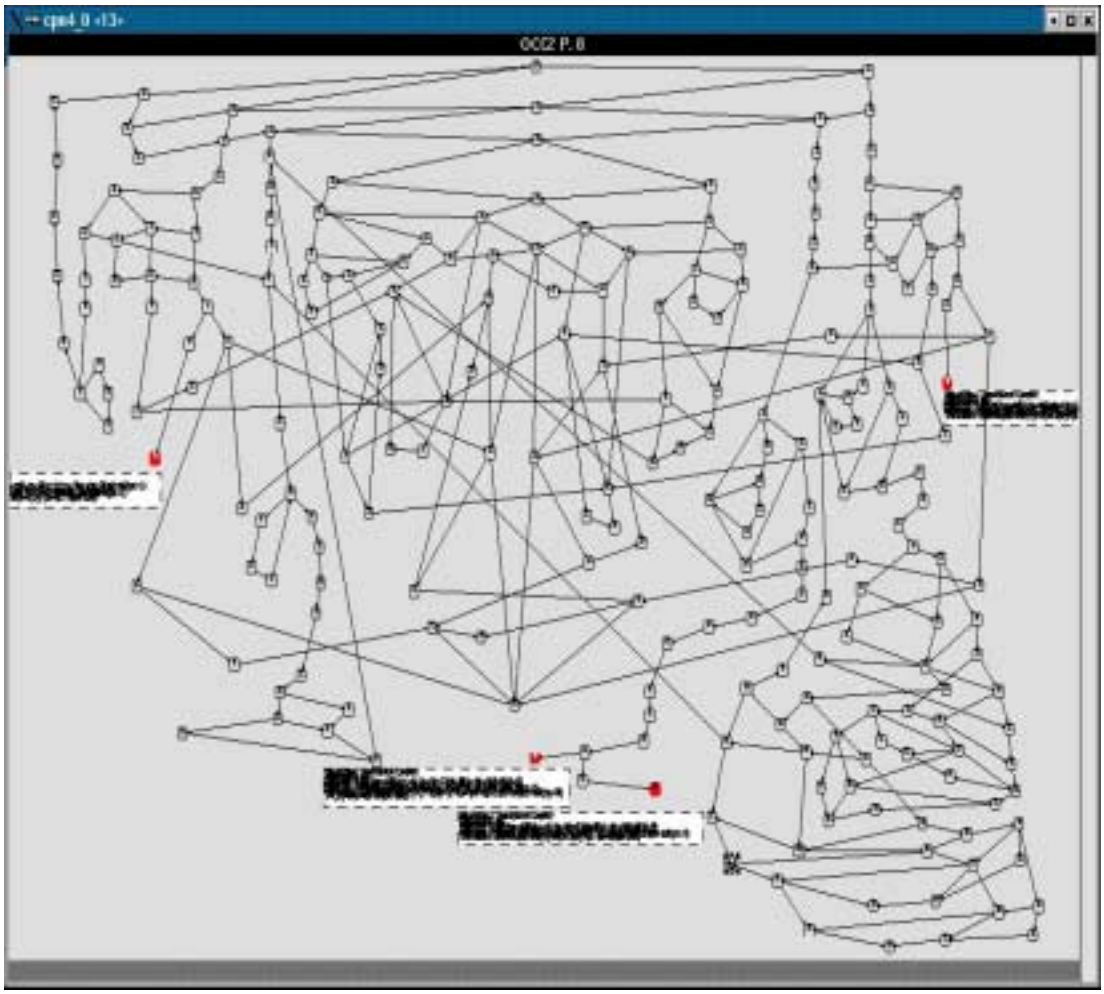


Figura 5.33 – *Grafo de marcações alcançáveis para o Cenário 3*

As marcações destacadas na figura acima são descritas a seguir.

No caso do cenário 3, apesar do sistema ser limitado, ele é não-vivo, nas marcações 92, 99, 202 e 210. A Figura 5.34 mostra os detalhes dessas marcações.

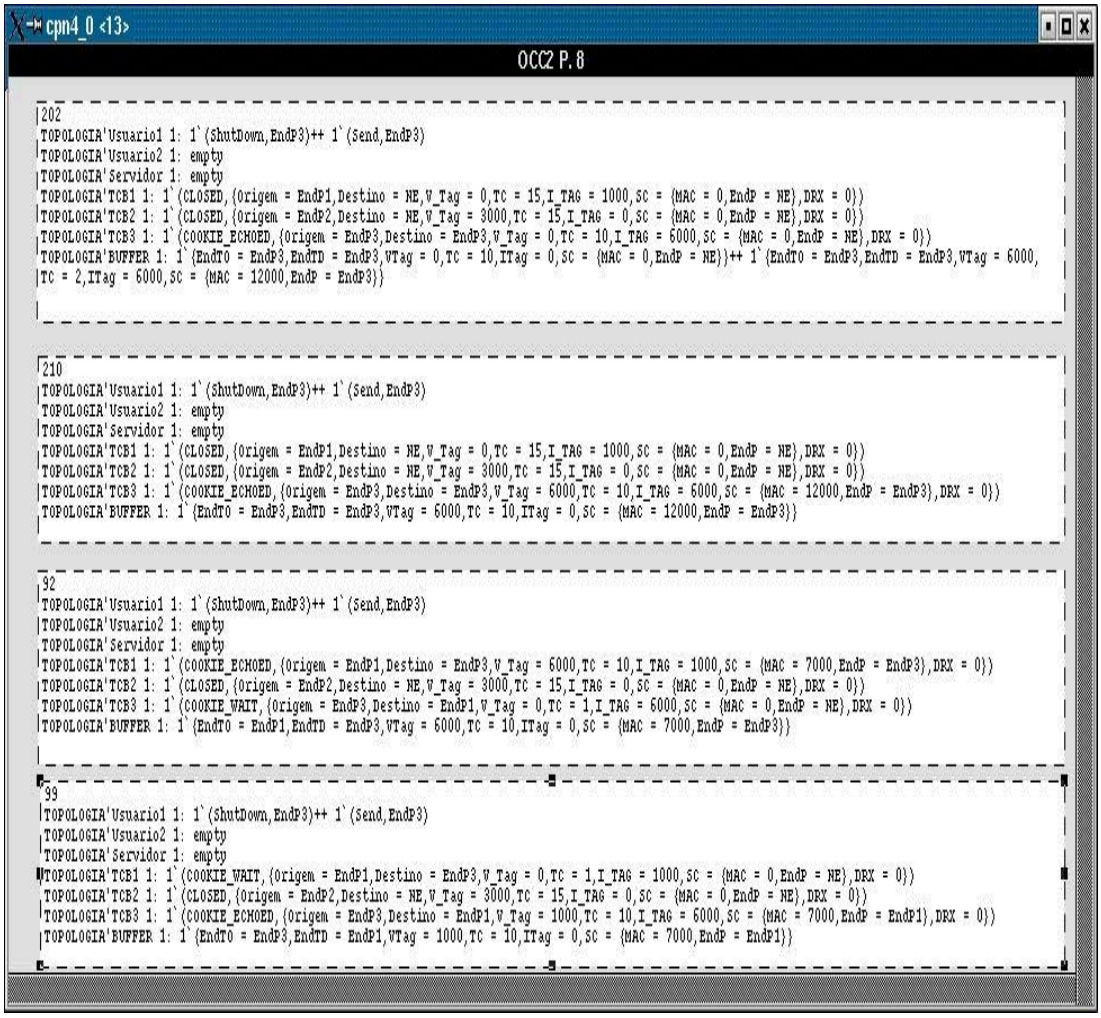


Figura 5.34 –M*arcações mortas (Deadlock) do modelo para o Cenário 3*

A partir da análise dessas marcações, foi possível constatar que os *deadlocks* encontrados no modelo não são oriundos da especificação do protocolo e, sim, do fato de o modelo não estar preparado para processar todas as seqüências de eventos de mensagens duplicadas ou inesperadas. Para que esses eventos e muitos outros (cenários) pudessem ser verificados seria necessário uma melhoria do modelo, o que não foi realizado no escopo desta dissertação.

Contudo, o objetivo da dissertação foi alcançado pelo fato de que a ferramenta rede de Petri foi, de fato, empregada para o estudo e verificação formal de algumas funcionalidades do protocolo e de se ter constatado sua importância no estudo e desenvolvimento de protocolos de comunicação. A continuação do desenvolvimento da CPN do SCTP fica proposta para trabalhos futuros.

Capítulo VI – Conclusões

Foi descrito, nessa dissertação, a importância do protocolo SCTP no cenário de convergência das redes SS7 e IP e foi realizado o estudo e verificação de consistência de especificações funcionais do SCTP através da ferramenta Redes de Petri. As experimentações das redes de Petri no modelamento formal das especificações do protocolo SCTP tiveram objetivo de analisá-las qualitativamente e não quantitativamente. Ou seja, o modelamento permitiu a verificação das propriedades sistêmicas estruturais e comportamentais e não de desempenho do protocolo.

A primeira vantagem de utilizar as RdPs foi evidenciada no processo de especificação, visto que exige do modelador bons conhecimentos sobre o sistema a ser desenvolvido. O processo de abstração do sistema fez com que fossem feitos questionamentos sobre o sistema, funcionando como uma poderosa ferramenta de *feedback,* tanto no estudo quanto no desenvolvimento do modelo do protocolo. De modo mais consistente que o apresentado pelas RFCs, a descrição dos protocolos, em RdPs, apresentam as especificações de maneira formal, direta e sem ambigüidade, o que é fundamental para o entendimento das especificações e para o correto desenvolvimento do projeto.

Apesar do tempo demandado na fase de modelagem do sistema ser considerável, as etapas seguintes são mais rápidas e confiáveis, e o tempo de modelamento com RdP pode ser reduzido proporcionalmente à experiência do modelador, que tende a crescer com a prática na criação das redes.

Os modelamentos baseados em redes de Petri com temporização, desenvolvidos nessa dissertação, mostram ser especificações visuais convenientes para expressar não só as estruturas de controle, mas, principalmente, as restrições de tempo intrínsecas ao sistema, onde os mecanismos que tratam eventos temporais de perda e retransmissão de mensagens por *time-out* e o *Heartbeat* têm implicação fundamental na correção funcional do SCTP, evitando ou levando o sistema a uma explosão de estados ou mesmo a um estado de bloqueio.

O aspecto do *four-way handshake*, que proporciona maior segurança ao protocolo SCTP, evitanto o ataque do tipo DOS (*Denial of Service*), foi especificado facilmente pela RdP, tornando a visualização dos eventos e procedimentos muito mais perceptíveis. Além disso, a RdP permitiu várias análises sobre os efeitos dos eventos temporais nessa fase do sistema e seus impactos na sincronização, onde se constatou a necessidade se definir as janelas de tempo corretas nesta fase do SCTP.

Situações que envolvem paralelismo, sincronismo, priorização e decisão, foram visualmente apresentadas pelas RdPs e, através do PAREDE, verificou-se seqüências de eventos aleatórios possíveis e a necessidade de se prever tais eventos e preparar o sistema para processar tais eventos aleatórios. Gerando-se a Árvore de Classes de Estados pode-se chegar às propriedades dinâmicas do sistema, como Limitação, Vivacidade, Reiniciação e, dessa forma, validar algumas funcionalidades do protocolo nesse nível de abordagem.

Já a especificação baseada em CPN mostrou-se fundamental para a modelagem detalhada das mensagens de controle do sistema, a ponto de nos aproximar do código do protocolo (Estrutura do código fonte do protocolo). Além disso, o Design/CPN proveu recursos apropriados para abstrações hierárquicas de modelagem do SCTP, o que possibilitou a criação de um modelo bem estruturado para visualização e simulação. E, mais uma vez, o método de geração das marcações alcançáveis mostrou-se fundamental, tanto na fase de elaboração do modelo quanto na sua verificação. A validação das especificações do SCTP foi possível apenas para

os dois primeiros cenários. Para validar a correção funcional de outras especificações do protocolo seria necessária uma melhoria do modelo.

Não obstante, o Design/CPN não faz análise de redes com temporização. Isso limita muito o desenvolvimento do modelo do sistema e, sobretudo, os resultados de análise. Portanto, há necessidade de uma ferramenta que tenha os recursos de modelagem do Design/CPN e a capacidade de análise do PAREDE.

O fato das especificações desenvolvidas em RdPs demonstrarem correção de algumas funcionalidades verificadas, ainda não é suficiente para validar a correção do protocolo. Seria necessário desenvolver um modelo mais completo para garantir a correção funcional do SCTP. No entanto, dentro dos objetivos desta dissertação, o desenvolvimento dos modelos e suas análises puderam evidenciar que o emprego das redes de Petri no estudo, desenvolvimento e especificação de protocolos de comunicação pode trazer maior garantia de funcionamento livre de erro de projeto.

6.1 - Trabalhos Futuros

- Desenvolver um modelo completo, em RdPs, do SCTP; Desenvolver ferramentas software que tenham o poder de modelagem do Design/CPN e os procedimentos de análise do PAREDE;

- Estudo e emprego de RdPs Estocástica, Orientada a Objetos e de Lógica Fuzzy; Pesquisa e desenvolvimento de métodos formais de análise das redes de Petri complexas;

- Pesquisa e desenvolvimento de ambientes de desenvolvimento de protocolos; Pesquisa e desenvolvimento com o *JAIN Protocol APIs*, para telefonia, INs, *wireless networks* e Internet.

APÊNDICE - Conteúdo do CD-ROM

Faz parte desta dissertação de mestrado um CD-ROM que inclui alguns dos documentos citados na seção da Bibliografia, disponíveis em formato eletrônico, e muitos outros arquivos relacionados com redes de Petri e SCTP. Todos os documentos públicos encontrados na Internet ao longo do período de pesquisa estão disponibilizados no CD-ROM, exceto aqueles que são livros ou revistas ou os publicados pelo **ITU-T, IEEE** ou **ACM**, que não são liberados para cópia. Para esses documentos é necessário ser assinante das organizações.

Foi criada uma pasta **/Bibliografia**. Dentro dessa pasta existem as subpastas **/PetriNet e /SCTP.** Dentro de **/PetriNet** existem as subpastas **/Documentos** e **/Ferramentas**, e dentro dessas subpastas uma coleção de documentos e ferramentas encontradas. Dentro da subpasta **/SCTP** estão vários documentos do IETF (RFCs e Drafts Internet) relacionados ao SCTP e outros artigos públicos encontrados. Todos os documentos estão no formado **.txt, .pdf, .ps** ou **.htm.**

Finalmente, dentro da pasta /bibliografia existe ainda a subpasta **/Dissertação**, onde se encontra este documento em formato eletrônico, em PDF.

REFERËNCIAS BIBLIOGRAFIA

[Berg82] Berg, H.K. & Boebert, W.E & Franta, W.R. & Moher, T.G. [1982]. *Formal methods of program verification and specification*. New Jersey: Prentice-Hall, Inc., 1982.

[Bova99] Bova, T., e Krivoruchka, T.: *Reliable UDP Protocol*, Internet-Draft, expirado. Agosto de 1999. http://www.watersprings.org/pub/id/draft-ietf-sigtran-reliable-udp-00.txt

[Coll01] Collins, Daniel, *Carrier Grade Voice Over IP*, Professional Telecom, McGraw-Hill, USA, 2001.

[Gira98] Claude, Girault e Rüdiger, Valk, *Petri Nets for Systems Engineering. A Guide to Modeling, Verification, and Applications* - Springer, 1998.

[Gold00] Goldbarg, Marcos C. e L. Luna P. H., *Modelos e Algoritmos*, UFRJ, Ed. Campos, 2000.

[ITUB96] ITU-T: B-ISDN ATM adaptation layer – Service specific connection oriented protocol (SSCOP), Recommendation V.42, outubro de 1996.

[ITUH99] ITU-T: Packet – based multimedia communication systems. Annex E: Framework and wire-protocol for multiplexed call signaling transport, Recommendation H.323, Annex E, maio de 1999.

[Jens92] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Us*e. Volume 1. Springer-Verlag. 1992.

[Jens97] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Volume 3. Springer-Verlag. 1997.

[Loug03] Loghney, J. (editor), "*Signalling Connection Control Part User Adaptation Layer (SUA)*", draft-ietf-sigtran-sua-15.txt, expira em dezembro 2003.

[MaG99] Ma, G.: *T/UDP: UDP for TCAP*, Internet-Draft, expirado. Maio de 1999. http://www.watersprings.org/pub/id/draft-ma-tudp-00.txt

[MCPN93] *Design/CPN Reference Manual for X-Windows*. Version 2.0. Meta Software Corpora-tion. 1993.

[Mena01] Menasche, M., *Apostila de Redes de Petri*, Mestrado do Inatel, Santa Rita do Sapucaí, 2001.

[Mena85] MENASCHE, MIGUEL, *PAREDE: An Automated Tools For The Analysis of Time(d) Petri Nets*, International Workshop on Timed Petri Nets, IEEE, Torino, Italy, julho de 1985.

[Merl74] P.M. Merlin. *A Study of recoverability of computer Systems*. PhD. Thesis, Computer Science University of California IRVINE, 1974.

[Morn01] Morneault, K., Sidebottom, G., Kalla, M. e Rengasami, S, "*ISDN Q.921-User Adaptation Layer*", RFC 3057, fevereiro 2001. http://www.ietf.org/rfc/rfc3057.txt

[Morn02] Morneault, K., Dantu, R., Sidebottom, G., Bidulock, B. e J. Heitz, "*Signaling System 7 (SS7) Message Transfer Part 2 (MTP2) - User Adaptation Layer*", RFC 3331, agosto 2002. http://www.ietf.org/rfc/rfc3331.txt

[Ong99] Ong, L., Rytina, I., Holdrege, M., Lode, C., García, M. A., Sharp, C., Juhasz, I., Lin, H. P., e Schwarzbauer, H. J.: *Framework Architecture for Signaling Transport*, RFC 2719, outubro 1999. http://www.ietf.org/rfc/rfc2719.txt

[Post80] Postel, J. (editor): *User Datagram Protocol*, RFC 768, agosto 1980. http://www.ietf.org/rfc/rfc768.txt

[Posl81] Postel, J. (editor): *Internet Protocol*, RFC 791, setembro 1981. http://www.ietf.org/rfc/rfc791.txt

[Post81] Postel, T., (editor): *Transmission Control Protocol*, RFC 793, setembro 1981. http://www.ietf.org/rfc/rfc793.txt

[Ramc74] Ramchandani C., "*Analysis of asynchronous concurrent systems by timed Petri nets*", Massachusetts Institute of Technology, Cambridge MA, 1974.

[Reis98] REISIG, Wolfgang, *Elements of Distributed Algorithms. Modeling and Anslysis with Petri Nets*. Humboldt -Universität zu Berlin - Springer Verlag, 1998.

[RFC1750] D. Eastlake, S. Crocker. *Randomness Recommendations for Security*, IETF RFC 1750, dezembro de 1994, www.ietf.org.

[RFC2104] H. Krawczyk, M. Bellare, *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104, fevereiro de 1997, www.ietf.org.

[RFC2522] P. Karn, W. Simpson, *Session-Key Management Protocol*, IETF RFC 2522, março de 1999, www.ietf.org.

[Russ00] Russell Travis, *Signalling System #7*, Third Edition, Telecomunicações, McGraw-Hill, New York, USA, 2000.

[Sánc99] Sánchez, D., A simple SCCP Tunneling Protocol (SSTP), Internet-Draft, expirado. Julho de 1999. http://www.watersprings.org/pub/id/draft-sanchez-garcia-SSTP-v1r0-00.txt

[Sánd99] Sánchez, D., Connectionless SCCP over IP Adaptation Layer (CSIP), Internet-Draft, expirado. Maio de 1999. http://www.watersprings.org/pub/id/draft-sanchez-CSIP-v0r0-00.txt

[Side02] Sidebottom, G., Morneault, K., e Pastor-Balbas, J., Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA), RFC 3332, setembro 2002. http://www.ietf.org/rfc/rfc3332.txt

[Site1] http://www.daimi.au.dk/cgi-desgncpn/showlicenselist

[Site2] http://www.daimi.au.dk/CPnets/intro/example_indu.htm

[Site3] http://www.daimi.au.dk/designCPN/

[Site6] http://www.daimi.au.dk/PetriNets/tools/db.html

[Site9] http://www.sctp.org

[SteI00] Stewart, R. R, Ong, L., Arias-Rodríguez, I., e Poon, K.: *SCTP Implementors Guide*, Internet-Draft, expirado. Janeiro de 2000. http://www.watersprings.org/pub/id/draft-ietf-tsvwg-sctpimpguide- 03.txt

[Stew00] R. Stewart, Q. Xie, ett, *Stream Control Transmission Protocol*, IETF RFC 2960, outubro de 2000, www.ietf.org.

[Tane96] Tanenbaum, A. S.: *Computer Networks. Third Edition*, Prentice-Hall International, 1996.

[Tone99] TONEY, K.: *PURDET. Reliable Transport Extensions on UDP*, Internet-Draft, expirado. Setembro de 1999. http://www.watersprings.org/pub/id/draft-toney-purdet-00.txt