ANÁLISE DE UMA ARQUITETURA DE SEGURANÇA EM SISTEMAS MÓVEIS DE TERCEIRA GERAÇÃO

Jaime César Ribeiro Lopes

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____

Prof. Aloysio de Castro Pinto Pedroza, Dr.

_____

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

_____

Prof. Orlando Gomes Loques Filho, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2004

LOPES, JAIME CÉSAR RIBEIRO

Análise de uma Arquitetura de Segurança em Sistemas Móveis de Terceira Geração [Rio de Janeiro] 2004

XIII, 107 p. 29,7 cm, (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2004)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 – Arquiteturas de Segurança

2 – Sistemas Móveis de Terceira Geração

3 – Métodos Formais

I. COPPE/UFRJ II. Título (série)

Àqueles que fazem da busca do conhecimento o caminho de uma vida.

Agradecimentos

Toda lista de agradecimentos eventualmente peca pela ausência de uma ou outra pessoa que teve papel, ainda que pequeno, na conclusão de um trabalho como este. Mesmo assim, arrisco-me a relacionar os nomes de alguns que participaram dessa jornada comigo, ou antes dela forneceram-me os meios para a travessia.

À minha família, pela formação que me permitiu ter, com os sacrifícios que só ela sabe quais foram, e à minha esposa, pela paciência e pelo incentivo durante toda nossa ainda curta vida de casados.

Ao meu colega e amigo, Prof. Jorge Luiz Pestana Ferreira, da UNIABEU, que não só me convenceu a vir fazer o mestrado como me ajudou com a inscrição, bem como à UNIABEU, pelo apoio e incentivo à conclusão do curso.

Pelo saber transmitido nesses quatro anos, aos professores do GTA Jorge Lopes de Souza Leão, José Ferreira de Rezende, Otto Carlos Muniz Bandeira Duarte, e, em particular, ao meu orientador, o Prof. Aloysio de Castro Pinto Pedroza, pela orientação segura, pela paciência com meus erros e pelo entusiasmo com os meus acertos.

Ao professor Orlando Loques, que participou como examinador desta tese.

Ao Programa de Engenharia Elétrica da COPPE, pela infra-estrutura tornada disponível para o desenvolvimento deste e de outros trabalhos, e ao seu pessoal técnico e administrativo, que, de uma forma ou de outra, acabou por participar deste trabalho.

A todos os colegas do GTA, alguns cujos nomes nem sequer sei, mas todos eles muito importantes para a continuidade dos trabalhos de pesquisa de nosso grupo, em especial alguns com quem troquei idéias e de quem recebi sugestões e experiência, como Aline, Artur, Avalle, Eric, Fagundes, Gardel, Gil, Kleber, Marcelo Rubinstein, Marcial, Márcio, Paulo, Pedro, Roberta, Saulo e Sérgio Granato.

E, por fim, e de modo bem particular, agradeço ao grupo muito unido que trabalhou com a implantação de algumas das técnicas de descrição formal hoje usadas no GTA, ou seja, David Fernandes, nosso mentor intelectual, Renato Bagatelli, o pioneiro a defender tese de mestrado, e Fabrício Ribeiro, parceiro e colaborador nos trabalhos ao longo de todo o curso, e de cujo incentivo e apoio moral minha tese muito prescindiu para se concretizar.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ANÁLISE DE UMA ARQUITETURA DE SEGURANÇA EM SISTEMAS MÓVEIS DE TERCEIRA GERAÇÃO

Jaime César Ribeiro Lopes

Março/2004

Orientador: Aloysio de Castro Pinto Pedroza

Programa: Engenharia Elétrica

A 3ª Geração de Sistemas Móveis de Telecomunicações busca integrar serviços de dados, voz e multimídia numa única arquitetura. Dentre os diversos aspectos em consideração no seu desenvolvimento, está o da segurança, como deve possuir qualquer arquitetura que envolva protocolos de comunicações de dados.

Este trabalho apresenta uma proposta de arquitetura de segurança para os Sistemas Móveis de 3ª Geração que emprega o SCTP como protocolo de transporte, especificando parte dele com o uso de técnicas de descrição formal empregando a linguagem LOTOS.

Na análise do protocolo, para simular sua operação, verificar o atendimento a propriedades de equivalência e validar seu comportamento, foi empregado o pacote de ferramentas CADP. Os resultados obtidos demonstram que o protocolo SCTP, na forma como foi especificado, converge e não apresenta *deadlocks* que impeçam sua correta operação, bem como que suas características o tornam desejável integrante de uma arquitetura de segurança. Além disso, mostram a utilidade da metodologia de análise empregada, baseada na descrição formal com o uso da linguagem LOTOS, para a validação de um protocolo sem a necessidade de sua implementação prévia.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ANALYSIS OF A THIRD GENERATION MOBILE SYSTEMS SECURITY ARCHITECTURE

Jaime César Ribeiro Lopes

March/2004

Advisor: Aloysio de Castro Pinto Pedroza

Department: Electrical Engineering

The Third Generation of Mobile Telecommunication Systems aims the integration of data, voice and multimedia services in a single architecture. Amidst the various aspects being considered in its development, the security is one of them, as any data communications protocol stack must have it.

This work proposes a security architecture to be used with the Third Generation Mobile Systems, in which the SCTP acts as transport protocol. Part of this protocol was specified with the use of formal description techniques, in LOTOS language.

The Caesar Aldebáran Development Package (CADP) carried the process of protocol analysis, through the simulation of its operation, the verification of its equivalence properties and the validation of its behavior. The attained results show that SCTP, the way it was specified, is convergent and do not have deadlocks that may hinder its correct operation. Also, the results show that SCTP´s characteristics make it desirable as part of a security architecture and that the methodology of analysis, based on formal description techniques, is useful to protocol validation without prior implementation.

Sumário

Lista de Acrônimos e Siglas:

3GPP:	*Third Generation Partnership Program*
AH:	*Authentication Header*
AN:	*Access Network*
BCG:	*Binary Coded Graphic*
CADP:	*CAESAR/ALDEBARAN Development Package*
CORBA:	*Common Object Request Broker Architecture*
CSCF:	*Call Server Control Function*
DoS:	*Denial of Service*
DSS1, DSS2:	*Digital Subscriber Signaling System*
ESP:	*Encapsulating Security Payload*
ETSI:	*European Telecommunications Standards Institute*
GIOP:	*General Inter-ORB Protocol*
GPRS:	*General Packet Radio Service*
GSM:	*Global System for Mobile Communication*
HE:	*Home Environment*
HSS:	*Home Subscriber Server*
IETF:	*Internet Engineering Task Force*
IKE:	*Internet Key Exchange*
IMSI:	*International Mobile Subscriber Identity*
IP:	*Internet Protocol*
ISDN:	*Integrated Services Digital Network*
ISO:	*International Standards Organization*
ISN:	*Initial Sequence Number*
LAN:	*Local Area Network*
LOTOS:	*Language of Temporal Ordering Specification*
LTS:	*Labeled Transition System*
MAC:	*Message Authentication Code*
ME:	*Mobile Equipment*
MGC:	*Media Gateway Controller*
MIP:	*Mobile IP*
MTU:	*Maximum Transmission Unit*
NDS:	*Network Domain Security*
NE:	*Network Entity*

ORB: *Object Request Broker*
OSI: *Open Systems Interconnection*
PDU: *Protocol Data Unit*
PIN: *Personal Identification Number*
PS: *Packet Switched* ou *Packet Switching*
PSTN: *Public Telephone Switched Network*
RNC: *Radio Network Controller*
RTT: *Round-trip Time*
SAD: *Security Association Database*
SCTP: *Stream Control Transmission Protocol*
SDU: *Service Data Unit*
SEG: *Security Gateway*
SG: *Signaling Gateway*
SIGTRAN: *Signaling Transport*
SIM: *Subscriber Identity Module*
SN: *Serving Network*
SPD: *Security Policy Database*
SS7: *Signaling System No. 7*
TCB: *Transmission Control Block*
TCP: *Transport Control Protocol*
TSN: *Transmission Sequence Number*
UA: *User Agent*
UDP: *User Datagram Protocol*
EU: *User Equipment*
UMTS: *Universal Mobile Telecommunications System*
USIM: *Universal Subscriber Identity Module*
VT: *Verification Tag*
WAN: *Wide Area Network*
XTL: *Executable Temporal Language*

Lista de Figuras:

Lista de Tabelas:

Capítulo 1

Introdução

A segurança é preocupação constante em todos os ambientes em que se realizam processamento, manipulação ou transporte de dados. Com o desenvolvimento da telefonia móvel celular, que propiciou a universalização das comunicações de voz, a comunicação móvel de dados recebeu igualmente impulso no sentido de atingir a ubiqüidade. Para este fim, e considerando a demanda crescente dos usuários por aplicações com elevada qualidade de apresentação gráfica e sonora, incluindo diversos recursos multimídia, está em desenvolvimento e formulação a arquitetura universal de sistemas de telecomunicações (*UMTS – Universal Mobile Telecommunication Systems*).

De fato, espera-se que a arquitetura de 3ª Geração (3G) estenda-se além da telefonia e abranja o fornecimento de comunicação de dados em alta velocidade. Essa arquitetura vem sendo desenvolvida pelo grupo de trabalho 3GPP (*Third Generation Partnership Project*).

Em paralelo à definição dessa arquitetura, com a comunicação de dados muito presente e o foco na convergência de diversas tipos de redes, uma arquitetura de segurança abrangente deve também ser proposta. Contudo, a forma desta arquitetura de segurança ainda não foi estabelecida em definitivo pelo grupo de trabalho 3GPP. Devido a isto, a proposta deste trabalho é formalizar uma arquitetura de segurança para as redes de 3ª Geração baseada na arquitetura de segurança IP (*Internet Protocol*), denominada IPSec (KENT, ATKINSON, 1998), e usando o protocolo SCTP (*Stream Control Transmission Protocol*) como o protocolo da camada de transporte.

Além disso, como a comunicação móvel possui restrições inerentes de

vulnerabilidade e de banda, e apresenta cada vez mais uma complexidade sistêmica, o nível de abstração necessário à produção de definições e padrões eleva-se. O foco muda, passando para a definição abstrata de funcionalidades de um serviço. Assim sendo, a verificação das propriedades de segurança de um protocolo beneficia-se de uma especificação mais formal (RIBEIRO *et al.*, 2003b).

A abordagem empregada neste trabalho consiste na modelagem do protocolo de transporte SCTP, inserido numa arquitetura que visa dar garantias de segurança à comunicação móvel de 3ª Geração. Utiliza-se, para esse fim, uma técnica de descrição formal, baseada na linguagem LOTOS (*Language of Temporal Ordering Specification*), e, para os procedimentos de análise, as ferramentas contidas no pacote denominado CADP - *CAESAR/ALDEBARAN Development Package* (FERNANDEZ *et al.*, 1996).

1.1. A Segurança no Sistema Móvel de 3ª Geração

A arquitetura de 3ª Geração pretende tornar a comunicação universal e transparente ao usuário, independente de plataforma de *hardware*, sistema operacional ou tipo de equipamento, móvel ou fixo. Seu desenvolvimento, entretanto, envolverá alterações significativas na base de equipamentos sem fio instalada. O núcleo dessa rede deverá mudar da comutação de circuitos, empregada em tráfego de voz, para comutação de pacotes, além de que deverá ser construído de forma a operar de forma independente da tecnologia de acesso empregada.

A arquitetura de 3ª Geração deverá ser baseada em redes IP, já que este protocolo tornou-se o protocolo universal para comunicações de dados em rede. O uso de pacotes IP na estrutura de transmissão de dados e de sinalização apresenta-se como o caminho natural para a convergência entre as redes fixas e móveis. Sendo assim, muitos dos requisitos de segurança atualmente em definição para as redes de 3ª Geração deverão seguir os aspectos já adotados nas redes convencionais e balizarão o desenvolvimento dos seus análogos para redes móveis e sem fio.

Contudo, a arquitetura de segurança ainda não foi definida pelo grupo de trabalho 3GPP. Entretanto, já que a arquitetura a ser desenvolvida apoiar-se-á no protocolo IP, o modelo padrão para protocolos de segurança de 3ª Geração deverá se basear, naturalmente, na arquitetura IPSec (KENT, ATKINSON, 1998), para garantir às redes sem fio a interoperabilidade com os serviços já utilizados nas redes atuais.

Os aspectos de segurança das redes móveis de 3ª Geração são de grande importância devido à vulnerabilidade intrínseca dos sistemas de comunicação sem fio,

incluindo a facilidade de escuta passiva e a de ataques ativos. Apesar da diversidade de infra-estruturas empregadas, são necessários mecanismos de segurança que garantam a integridade e a privacidade da comunicação, bem como a autenticação das entidades envolvidas.

Como toda arquitetura de protocolos voltada para a comunicação de dados, a arquitetura de segurança deve ser organizada em camadas, cada uma delas provendo parte da segurança requerida. Tais camadas apresentarão protocolos de sinalização e funções diversas de monitoramento no domínio da operadora, e cada uma delas apresentará requisitos definidos com base nos padrões de segurança desejados dentro de cada um dos domínios, ou entre domínios, no caso da segurança provida fim-a-fim. Por outro lado, pelas limitações de vulnerabilidade e banda dos sistemas sem fio, a sinalização deve apresentar alta complexidade e gerar pequeno volume de tráfego adicional (RIBEIRO *et al.*, 2003a).

Muitos estudos sobre segurança em redes 3G concentram-se na tentativa de buscar uma arquitetura totalmente nova, sem vínculo com os processos de segurança aplicados nas rede convencionais. Entretanto, mesmo esses estudos admitem que os sistemas e protocolos em uso podem ser adaptados às exigências do novo sistema, ou seja, que partes de uma arquitetura já estabelecida podem ser utilizadas.

1.2. Arquitetura de Segurança Baseada no Protocolo IPSec

Redes baseadas na arquitetura IP vêm evoluindo rapidamente, e coexistem com outras infra-estruturas de telecomunicações, capazes de transportar quaisquer tipos de aplicações e prover diferentes tipos de serviços. Essa realidade resulta numa crescente demanda por de interoperabilidade e interconexão entre os diferentes tipos de plataformas e sistemas. Visando proporcionar o atendimento aos requisitos de segurança e de interoperabilidade para as redes móveis de 3ª Geração, é necessário que a arquitetura utilizada por este novo sistema use protocolos já consagrados nas redes convencionais.

Entretanto, em muitos trabalhos, diversos autores propõem-se a discutir alternativas à arquitetura IP tradicional que, muitas vezes, abandonam por completo seus principais fundamentos. Por exemplo, muito tem sido discutido no campo dos protocolos de transporte, quando se trata de adaptar as soluções existentes às

restrições e características das redes sem fio.

Em WANG (1998), uma alternativa ao TCP (*Transport Control Protocol*) é proposta, na forma do METP (*Mobile-End Transport Protocol*), onde as limitações das redes sem fio, em particular as restrições de banda e as perdas por flutuação de sinal nos canais, são analisadas. Já nesse trabalho, surge a proposta de substituir o TCP por outra solução, apontando-se as principais limitações desse protocolo, como o tamanho de seus cabeçalhos e a assunção de que perdas sempre ocorrem por congestionamento (ao invés de por flutuações nos níveis de sinal ou por *hand-offs*), provocando uma reação contrária ao desejado (o *backing off*, que causa degradação no desempenho). Entretanto, a solução proposta abre mão também do uso do protocolo IP, o que vai de encontro ao que se tem visto como tendência dominante na implantação do núcleo das novas redes sem fio e à desejada interoperabilidade. O METP prestar-se-ia, tão somente, ao uso nos trechos sem fio do sistema.

Trabalhos mais recentes e abrangentes de arquiteturas que incluem o transporte de sinalização em sistemas mistos (redes fixas e sem fio) mencionam também o MIP (*Mobile IP*). O MIP é uma versão otimizada do protocolo IP, destinada ao uso em ambientes móveis, onde mecanismos mais sofisticados de roteamento são necessários ao suporte de entidades móveis durante uma conexão. Tais técnicas de roteamento compõem o que é, em essência, definido como roteamento triangular, permitindo às entidades móveis manterem números IP intactos, mesmo estando em redes que não as suas próprias. O roteamento envolve a criação de túneis, onde pacotes IP modificados encapsulam cabeçalhos adicionais, indicando os endereços de origem e destino originais.

Como o roteamento triangular exige o periódico anúncio das informações de roteamento, que não incorporam nenhum tipo de mecanismos de segurança para assegurar a autenticação das informações passadas pelo nó móvel ou pelo agente encarregado de rotear as informações para este, as informações pode ser objeto de adulteração.

Entretanto, da idéia de encapsulamento e tunelamento, surge o IPSec como alternativa viável, inclusive em estudo pelo fórum 3GPP (3GPP TS 33.210, 2003). Na arquitetura de segurança IPSec, o encapsulamento e o tunelamento são governados pelo uso dos protocolos AH (*Authentication Header*) e ESP (*Encapsulating Security Payload*), que elevam a segurança dos pacotes transportados contra ataques maliciosos, autenticando e encapsulando o conteúdo dos pacotes IP.

Por outro lado, no que diz respeito à camada de transporte, considerando-se a arquitetura de 3ª Geração em estudo pelo fórum 3GPP e analisando-se as propostas

oferecidas pela literatura atual, percebe-se claramente não ser possível fugir de uma comparação entre o protocolo estabelecido (o TCP) e alguma alternativa que supere suas reconhecidas limitações para a aplicação em redes sem fio. Dentro deste princípio, a maior parte da literatura atual vem privilegiando a discussão entre o uso do TCP e o uso do SCTP, comparando essencialmente os pontos fortes e fracos de ambos.

De fato, embora reconhecido como um protocolo padronizado para o transporte de sinalização telefônica através de redes IP, o SCTP tem sido estudado como alternativa completa ao TCP pelas suas características avançadas de controle de congestionamento e de tolerância a falhas. O SCTP, desde seu projeto inicial, vem sendo pensado para ser eficaz para aplicações que requeiram: transporte confiável de fluxos de dados; transporte ordenado de dados pertencentes a múltiplas mensagens não relacionadas; redundância na camada de rede. Outras vantagens que leva sobre o TCP são seus aperfeiçoamentos, como: *multi-homing*, controle independente de múltiplos fluxos dentro de uma mesma associação e capacidade de detecção de falhas de sessão ou caminho pelo seu monitoramento ativo. Os resultados apresentados em ALAMGIR *et al.* (2002) mostram, ainda, a justiça na competição por recursos entre implementações SCTP e TCP concorrentes e, mesmo assim, o SCTP obtém uma taxa de transmissão (*throughput*) até 30,6% maior que o TCP. Apresenta ainda, segundo esses autores, diversas vantagens sobre o TCP quando de uma recuperação de perda de segmentos, devido, principalmente, ao reconhecimento cumulativo (SACK), não disponível no TCP (ALAMGIR *et al.*, 2002). Por outro lado, nas implementações *multi-homed*, o *throughput* também é maior pela existência de um ou mais caminhos alternativos, por onde pacotes retransmitidos poderão chegar com sucesso, mesmo com a queda do caminho principal (RAVIER *et al.*, 2001).

Outros autores, embora não estritamente ligados ao tema do transporte na 3ª Geração, têm levantado outras aplicações de transporte onde o SCTP leva vantagem sobre o TCP, como em LIANG *et al.* (2002), onde se avalia o SCTP como mecanismo de transporte para mensagens GIOP (*General Inter-ORB Protocol*) no padrão CORBA (*Common Object Request Broker Architecture*). Nesse trabalho, as limitações do TCP como protocolo de transporte são relacionadas: é inadequado numa rede de sinalização de alto desempenho; carece de suporte inerente à mobilidade; requer uma camada adicional de protocolos para prover confiabilidade e mobilidade em sistemas sem fio. Uma das características do SCTP que é mencionada como vantajosa é que a retransmissão de um pacote perdido pertencente à uma mensagem afeta menos a entrega pontual de outras mensagens não-relacionadas àquela, visto haver uma

independência entre ordenações de fluxos diferentes (LIANG *et al.*, 2002).

Em outro trabalho, MARCO *et al.* (2003) apresentam o SCTP como um protocolo adequado ao transporte de tráfego associado ao SIP (*Session Initiation Protocol*), destacando o mecanismo de controle de congestionamento do SCTP como um de seus pontos fortes sobre o UDP ou o TCP.

Já KOH e JUNG (2003) discutem o SCTP para suporte à mobilidade IP na camada de transporte, destacando a sua característica de suporte a *multi-homing*.

Especificamente com relação a redes sem fio, destaca-se uma característica inerente ao TCP: inicialmente projetado para operar em redes convencionais onde a perda aleatória de pacotes é desprezível, pode apresentar sensível queda de desempenho quando aplicado a enlaces sem fio, com perdas significativas, especialmente as aleatórias, já que presume sempre que a perda de pacotes ocorre devido a congestionamento. As duas principais fontes de perda num enlace sem fio são as provocadas pelos *handoffs* (trocas de células que ocorrem para um usuário em movimento) e as ocasionadas pelo desvanecimento (*fading*) de sinais (HU, SHARMA, 2002).

Em outro trabalho, FU *et al.* (2002) avaliam e comparam o desempenho do SCTP, da implementação Eifel do TCP e do TCP Reno durante picos de atraso. Um pico de atraso, ou *delay spike*, é definido como uma situação onde o RTT (*Round-trip Time*) aumenta subitamente e depois cai acentuadamente ao seu valor habitual. Numa rede sem fio, causas de *delay spikes* são o *handoff* e a desconexão física do enlace sem fio, dentre outras, que levam a uma penalização acentuada do desempenho fim-a-fim no TCP. Na comparação entre os desempenhos do SCTP, do Eifel e do TCP Reno, os autores destacam o desempenho mais regular do SCTP em redes sujeitas a perdas, sempre superior ao Eifel e compatível com o TCP Reno, muito embora sua análise tenha se limitado a perdas por congestionamento (FU *et al.*, 2002), sabidamente aquelas para as quais o TCP foi projetado.

Essas comparações entre o TCP, como protocolo dominante na atual estrutura do núcleo da rede, e o SCTP, como uma alternativa, mostram que este tem bastante espaço a ocupar, inerentemente pelas suas características operacionais, pelas inovações sobre o TCP e, porque não dizer, também por sua compatibilidade com este tradicional protocolo, já que muitos dos processos internos do SCTP são cópias aperfeiçoadas do TCP. Por outro lado, e é o que ainda mais interessa aqui mencionar, suas características de segurança reforçada em relação ao TCP, que serão vistas mais adiante, aliadas às conclusões que tiram-se das discussões acima, fazem do SCTP um protocolo adequado à inserção em uma arquitetura de segurança.

Então, complementando a arquitetura de segurança, o SCTP aparece como um protocolo de transporte de uso geral, oferecendo serviços como *multi-homing*, permitindo à rede manter conexão (ou associação, para usar a terminologia própria do SCTP) mesmo com a falha de uma rota ou parte dela, e um certo grau de proteção contra ataques pela implementação de um mecanismo de quatro vias para o estabelecimento de conexões.

1.3. O Protocolo SCTP

Embora o foco original das redes metropolitanas fosse o de comunicações de voz, para as quais essas redes foram projetadas e otimizadas, a comunicação de dados evoluiu de forma independente na medida em que surgia a necessidade de interconexão de redes locais. Entretanto, somente com a penetração crescente dos computadores pessoais no ambiente doméstico e com a abertura da *Internet* aos usuários em geral, foi que surgiu a necessidade de uma infra-estrutura básica para a comunicação de dados. Essa necessidade levou ao surgimento das redes digitais de serviços integrados, ou ISDN (*Integrated Services Digital Network*), baseadas nos já bem estabelecidos conceitos das redes de telecomunicações clássicas (JUNGMAIER *et al.*, 2001).

Com a introdução dos sistemas digitais, outro campo que expandiu-se rapidamente foi o das comunicações sem fio. Embora as redes GSM (*Global System for Mobile Communication*) tenham sido, como as WAN (*Wide Area Networks*), originalmente projetadas para o tráfego de voz, a introdução do GPRS (*General Packet Radio Service*) e, mais recentemente, o projeto do UMTS, proverão suporte adequado ao tráfego de dados e informações multimídia. Em paralelo, evoluíram também os conceitos e a tecnologia das LAN (*Local Area Networks*), e, como o volume do tráfego de dados já ultrapassou o de voz, a tendência de adoção de redes puramente baseadas na arquitetura IP consolidou-se (JUNGMAIER *et al.*, 2001).

Como conseqüência, a coexistência entre sistemas, plataformas e infra-estruturas diferentes exige uma operação cooperativa, o que levou ao esforço de desenvolvimento de protocolos de sinalização e controle para redes de usuário, como o DSS1 e DSS2 (*Digital Subscriber Signaling System*), e para sinalização interna, como o SS7 (*Signaling System No. 7*). Entretanto, um ponto crucial a ser considerado na evolução para redes puramente baseadas no protocolo IP é que o transporte das informações de sinalização e controle seja fornecido através das redes IP de forma

confiável e com adequado desempenho. Desta forma, uma arquitetura de transporte para dados de sinalização, em desenvolvimento no grupo SIGTRAN (*Signaling Transport*) do IETF (*Internet Engineering Task Force*), baseia-se num protocolo fim-a-fim para o transporte confiável de tráfego de sinalização originado em redes diversas através de redes IP, o já mencionado SCTP (JUNGMAIER *et al.*, 2001).

Embora o SCTP tenha sido inicialmente desenvolvido para o transporte de informações de sinalização orientadas a mensagem, e seus serviços tornados disponíveis através de camadas de adaptação definidas pelo SIGTRAN, com as adaptações necessárias, pode ser aplicado sempre que mensagens tenham que ser transportadas sobre uma arquitetura IP. Assim, opera, na verdade, como um protocolo de transporte qualquer.

A integração entre SS7 e TCP/IP faz uso de um dispositivo denominado SG (*Signaling Gateway*), que intermedia as comunicações entre terminais SS-7 puros e terminais IP/SS-7. De fato, no campo de aplicações desenvolvido pelo SIGTRAN, uma atenção especial foi dada ao transporte de sinalização de telecomunicações entre um SG e um MGC (*Media Gateway Controller*).

O SG suporta interconexão no plano de controle entre redes de telefonia pública (PSTN – *Public Telephone Switched Network*), baseadas na sinalização SS-7, e redes IP. Dependendo da modalidade de adaptação, o SG atua como roteador ou *gateway*. Além do SG, é necessário haver camadas de adaptação para os terminais IP/SS7, pois diversas metáforas presentes em SCTP e TCP/IP (porta, endereço, associação) são diferentes ou inexistentes em SS-7. A camada de adaptação é inserida entre o SCTP e as camadas superiores SS-7 implementadas pelo terminal.

Mas esta é apenas uma das aplicações do protocolo SCTP, sendo que outras, dentro do escopo de trabalho do SIGTRAN, podem ser mencionadas (JUNGMAIER *et al.*, 2001). Além disso, como o SCTP é também um novo protocolo de transporte, usado diretamente sobre redes IP, pode ser encarado como alternativa a protocolos já estabelecidos no campo da *Internet*, como o TCP e UDP (*User Datagram Protocol*), para uma variedade de aplicações.

Embora o SCTP, por si só, não seja imune a ataques, em particular o denominado “homem do meio” (*man-in-the-middle*), como nota a própria documentação de especificação do protocolo (STEWART *et al.*, 2000), com a possibilidade de ter suas mensagens interceptadas e alteradas em trânsito, o uso do SCTP dentro de uma arquitetura IPSec permite evitar o sucesso de tais ataques. A adoção dos protocolos AH e ESP, dentro da arquitetura IPSec, operando em conjunto com o SCTP, eleva de forma considerável o grau de segurança das operações de

troca de mensagem, além de aproveitar as demais características desejáveis deste último protocolo, incluindo seu desempenho superior ao TCP e sua orientação a mensagens, ao invés de a conexões.

1.4. Projeto

A proposta deste trabalho é formalizar uma arquitetura de segurança para as redes de 3ª Geração, baseada na arquitetura IPSec, e analisar, em particular, a adoção do SCTP como o protocolo de transporte, usuário desta arquitetura.

A abordagem consiste na modelagem do protocolo SCTP com o uso de uma técnica de descrição formal, baseada na linguagem LOTOS, e na análise deste protocolo usando as ferramentas contidas no pacote denominado CADP (FERNANDEZ *et al.*, 1996). Esta formalização permite que o comportamento do protocolo possa ser analisado e avaliado através de critérios mensuráveis e não apenas de forma empírica, evitando a necessidade de sua implementação prévia ao teste e possibilitando uma redução do trabalho de campo.

Este trabalho está organizado da seguinte forma: o Capítulo 2 detalha a arquitetura de segurança baseada em IPSec, que propomos utilizar no sistema móvel de 3ª Geração, bem como o protocolo SCTP; o Capítulo 3 apresenta a metodologia e o procedimento de análise e verificação empregado na avaliação do protocolo SCTP; o Capítulo 4 apresenta os resultados obtidos com as simulações da operação de partes do protocolo SCTP, empregando o pacote CADP; e o Capítulo 5 finaliza o trabalho, expondo as conclusões finais obtidas.

Capítulo 2

A Arquitetura de Segurança em Sistemas Móveis de 3ª Geração

O padrão UMTS, escolhido como a arquitetura para a 3ª Geração de sistemas móveis, baseia-se em três componentes principais: um sistema de satélites, a rede de acesso aos serviços e o núcleo da rede fixa (*core network*). A arquitetura para o núcleo da rede fixa está em processo de definição pelo 3GPP e pelo ETSI (*European Telecommunications Standards Institute*).

A segurança desses sistemas funda-se em três princípios (3GPP TS 33.120, 2001):

- os sistemas de 2ª Geração, considerando-se a robustez de seus elementos de segurança (autenticação do assinante, criptografia da interface rádio, confidencialidade da identidade do assinante e o uso de cartões SIM – *Subscriber Identity Module* – como módulos de segurança);
- a correção das falhas dos sistemas de 2ª Geração (como a falta de segurança no núcleo da rede (RIBEIRO *et al.*, 2003b));
- a oferta de novas características de segurança e novos serviços seguros.

A utilização do IP como o protocolo da camada de rede do *backbone* UMTS, significa que, por este domínio, circularão o tráfego de dados e o tráfego de sinalização. O uso do IP reforça o movimento de universalização da comutação de pacotes, mas representa o emprego de protocolos totalmente abertos e facilmente acessíveis. A implicação disto é que novas ameaças e riscos devem ser encarados, como a escuta passiva e os ataques ativos (3GPP TS 33.210, 2003). Assim, são

necessários mecanismos de segurança que garantam a integridade e a privacidade da comunicação, bem como a autenticação das entidades envolvidas.

A comunicação segura nas redes de 3ª Geração deve atender a três requisitos:

- autenticação - uma entidade é realmente quem ela diz ser;
- privacidade - o destinatário é o único capaz de compreender o conteúdo da mensagem;
- integridade - a mensagem recebida é idêntica àquela enviada pelo emissor.

Novas arquiteturas de segurança vêm sendo apresentadas, mas ainda em estado embrionário, não havendo um consenso a respeito. Por outro lado, o emprego de arquiteturas já testadas e consagradas em ambientes de rede convencionais é uma tendência que conta com o endosso do 3GPP, principalmente no que se refere ao núcleo da rede UMTS (3GPP TS 33.210, 2003). Este trabalho propõe uma arquitetura de segurança modificada, baseada em outras já testadas em ambientes convencionais, para se adequar às exigências da 3ª Geração.

Este capítulo apresenta, na seção 2.1, a arquitetura de segurança proposta pelo 3GPP. Na seção 2.2, as características e protocolos da arquitetura IPSec são estudados. A seção 2.3 mostra a arquitetura de trabalho proposta para oferecer as garantias de segurança aos sistemas de 3ª Geração, destacando o SCTP como o seu protocolo de transporte. A seção 2.4, detalha o protocolo SCTP. A seção 2.5, resume o capítulo, apresentando o assunto a ser discutido no próximo capítulo.

2.1. A Arquitetura de Segurança de 3ª Geração

Para os sistemas de 3ª Geração, proteger os protocolos de sinalização do núcleo da rede (que não eram criptografados na 2ª Geração) é um objetivo essencial, exigindo soluções de segurança para protocolos baseados no SS7 e para aqueles do domínio IP. Para protocolos baseados no IP, a segurança deve ser proporcionada na própria camada de rede, por meio do uso dos elementos integrantes da arquitetura de segurança IPSec (3GPP TS 33.210, 2003).

Assim, foram definidos pontos cruciais para garantia de segurança do sistema como um todo, conforme pode ser visto na figura 1 (3GPP TS 33.102, 2003):

- Segurança no Acesso a Rede (I) – características de segurança que oferecem aos usuários o acesso seguro aos serviços de 3ª Geração e, em particular, protegem-nos de ataques no rádio-enlace de acesso. Uma dessas características é a confidencialidade do usuário, pela qual garante-

se que a sua identidade permanente (ou seu IMSI - *International Mobile Subscriber Identity*), a sua presença ou chegada a uma certa área e o tipo de serviços que lhe estão sendo oferecidos não poderão ser detectados no rádio-enlace;

- Segurança no Domínio da Rede (II) – características de segurança que habilitam os nós no domínio da rede provedora a trocar informações de sinalização de forma segura e que protegem o sistema contra ataques na parte convencional da rede;
- Segurança no Domínio do Usuário (III) - características de segurança que fornecem acesso seguro às estações móveis usuárias, entre elas a autenticação usuário-USIM (*Universal Subscriber Identity Module*), através do uso de um código de acesso (o PIN - *Personal Identification Number*);
- Segurança no Domínio das Aplicações (IV) - características de segurança que habilitam as aplicações nos domínios do usuário e da rede provedora a intercambiar mensagens de forma segura;
- Visibilidade e Configuração da Segurança (V) (não mostrada na figura 1) - características de segurança que permitem ao usuário descobrir se determinada capacidade de segurança está em operação ou não, e se o fornecimento de determinados serviços depende dessa capacidade.

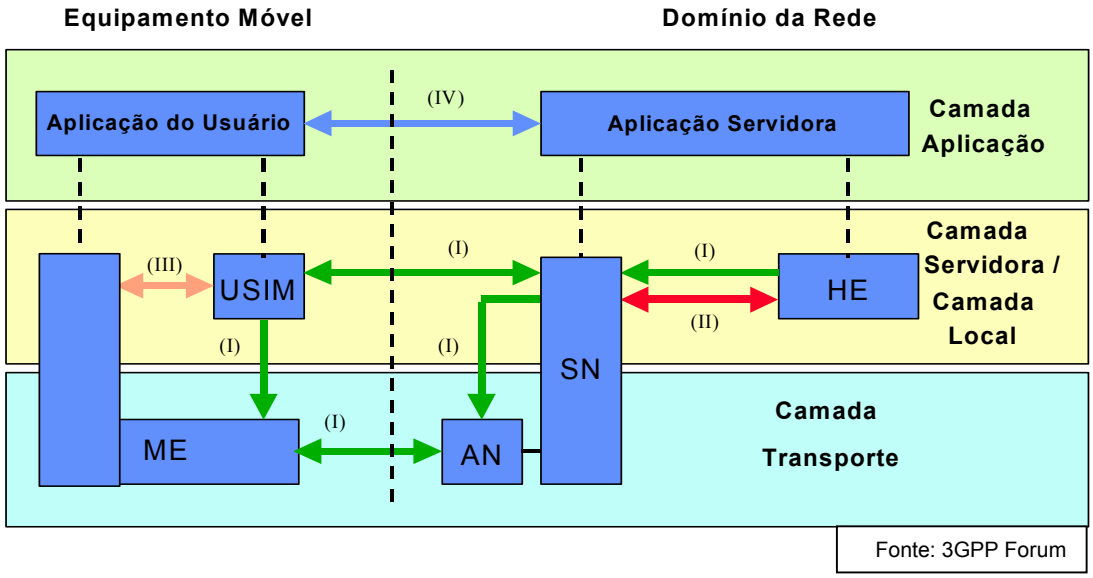

Figura 1. Visão geral da arquitetura de segurança 3G

Legenda:
- AN: Rede de Acesso (*Access Network*)
- USIM: Módulo de Identidade de Assinante Universal (*Universal Subscriber Identity Module*)
- ME: Equipamento Móvel (*Mobile Equipment*)
- SN: Rede Servidora (*Serving Network*)
- HE: Ambiente Local (*Home Environment*)

O estabelecimento de serviços seguros implica na necessidade de confiabilidade, integridade e autenticação. São procedimentos padronizados e baseados em técnicas de criptografia que podem assegurá-los.

Dentre os princípios estabelecidos na arquitetura de segurança a ser empregada no domínio das redes IP, um conceito essencial é o de domínio de segurança (ou domínio seguro (RIBEIRO *et al.*, 2003b)). Estes domínios são redes geridas por uma única autoridade administrativa, onde o nível de segurança e de utilização de serviços seguros é padronizado, como uma rede administrada por uma operadora de telefonia, mesmo que dividida em seções menores.

Como o plano de controle do domínio da rede UMTS é secionado em domínios de segurança que, tipicamente, coincidem com as bordas dos domínios das diferentes operadoras, é preciso garantir a proteção também nessas interfaces. As interfaces entre essas bordas são protegidas pelos denominados SEGs (*Security Gateways*) (RIBEIRO *et al.*, 2003a), que são responsáveis por garantir a política de segurança de um domínio em relação aos SEGs situados nos demais domínios que o confrontam, bem como a integridade e a autenticação dos dados de origem.

Um SEG pode ser definido como um roteador de borda, já que está postado na borda de um domínio seguro IP (interface Za – vide figura 2), e será utilizado para garantir a segurança de protocolos nativos IP. Interage com todos os domínios de segurança que o confrontam ou com apenas alguns deles, e gerencia todo o tráfego de comunicações através da interface Za. A esse tráfego, que requer a proteção de mecanismos de segurança, denominamos tráfego NDS/IP (onde NDS significa *Network Domain Security*). Cada SEG destina-se ao manuseio do tráfego NDS/IP de entrada ou de saída oriundo de, ou destinado a, um grupo bem definido de domínios seguros IP. Além disso, os SEGs são responsáveis por garantir as políticas de segurança na interconexão entre redes, o que inclui a filtragem de pacotes e as funcionalidades adicionais de um *firewall*, embora a especificação da arquitetura de 3ª Geração não as requeira (3GPP TS 33.210, 2003).

De qualquer modo, todo o tráfego IP dos domínios seguros de rede deve passar por estes roteadores de borda antes de entrar ou sair desses domínios, sendo o número destes dispositivos dependente do equilíbrio entre a necessidade de acessibilidade externa e o balanceamento de carga, para evitar um único ponto de falha. Na arquitetura de 3ª Geração, prevê-se que uma operadora possa ter mais de um SEG em sua rede, justamente para contornar a existência de apenas um ponto de falha, e que o domínio de rede UMTS deva ser dividido logicamente e fisicamente em domínios seguros.

No modelo de segurança proposto para os sistemas de 3ª Geração, estes dispositivos de borda devem ter capacidade de oferecer armazenamento seguro das chaves de autenticação empregadas no estabelecimento das associações seguras.

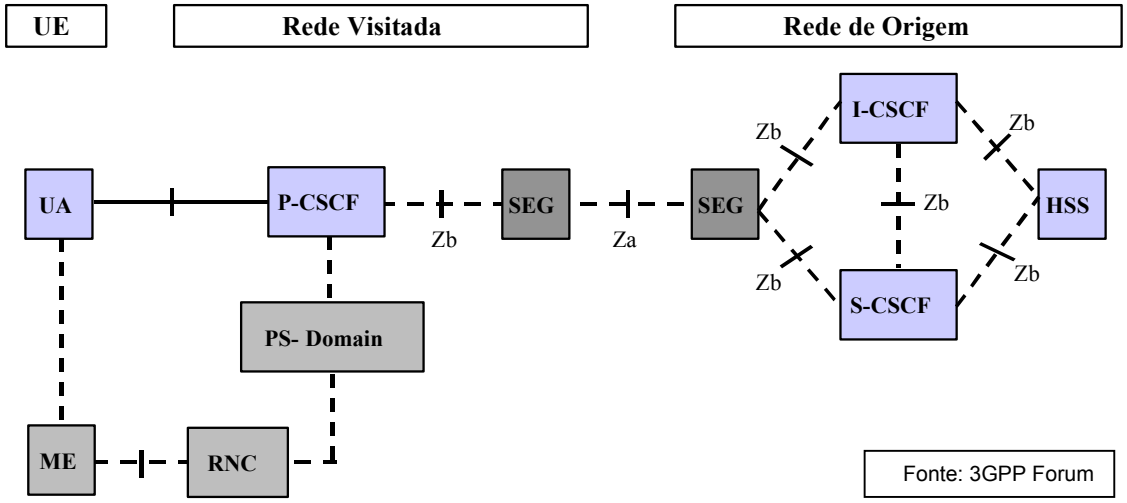

Figura 2. Arquitetura de Segurança entre Domínios Seguros

Legenda:
- UE: Equipamento do Usuário (*User Equipment*)
- UA: Agente do Usuário (*User Agent*)
- ME: Equipamento Móvel (*Mobile Equipment*)
- RNC: Controlador da Rede de Rádio (*Radio Network Controller*)
- PS-Domain: Domínio de Comutação de Pacotes (*Packet Switched*)
- CSCF: Função de Controle do Servidor de Chamada (*Call Server Control Function*)
- SEG: Roteador de Borda de Segurança (*Security Gateway*)
- HSS: Servidor do Assinante – Rede de Origem (*Home Subscriber Server*)

2.2. A Arquitetura IPSec

A arquitetura IPSec, conforme definida pelo IETF, oferece segurança aos serviços através do encapsulamento de pacotes IP e do estabelecimento de associações seguras. Estas associações são conexões unidirecionais, protegidas criptograficamente. A arquitetura NDS/IP empregada no gerenciamento das associações seguras, na troca de chaves criptográficas e na sua distribuição baseia-se no protocolo IKE (*Internet Key Exchange*) (HARKINS, CARRELL, 1998).

Certas opções disponíveis na arquitetura IPSec não são necessárias na arquitetura NDS/IP. Por outro lado, outras características, que são opcionais naquela arquitetura, tornam-se essenciais para a NDS/IP. A idéia básica da arquitetura NDS/IP é prover segurança nó a nó (*hop-by-hop security*), de acordo com o seu modelo de operação, baseado em túneis encadeados. O uso da segurança definida nó a nó permite operar diferentes políticas de segurança internamente e através de outros

domínios seguros externos.

Na arquitetura NDS/IP, e considerando este tipo de tráfego, apenas os SEGs devem manter comunicação direta com entidades pertencentes a outros domínios de segurança. Cabe aos SEGs estabelecer e manter associações seguras IPSec utilizando o protocolo ESP, criando túneis entre os domínios seguros. Os SEGs manterão, normalmente, pelo menos um túnel IPSec disponível para cada SEG correspondente. O SEG manterá duas bases de dados logicamente separadas, a SAD (*Security Association Database*) e a SPD (*Security Policy Database*), para cada uma das interfaces que possuir com outros SEGs.

As entidades pertencentes à rede (NE, ou *Network Entity*) estabelecem e mantêm associações seguras ESP em relação a um SEG ou outra NE no mesmo domínio, conforme necessário. Todo o tráfego NDS/IP oriundo de uma NE pertencente a um domínio seguro, direcionado a uma NE pertencente a outro domínio seguro, será roteado através de um SEG e disporá de segurança nó a nó até seu destino final.

Na figura 3 abaixo, vê-se um esquema de como são estabelecidas as associações seguras entre as NEs, dentro de um mesmo domínio seguro e entre diferentes domínios seguros.

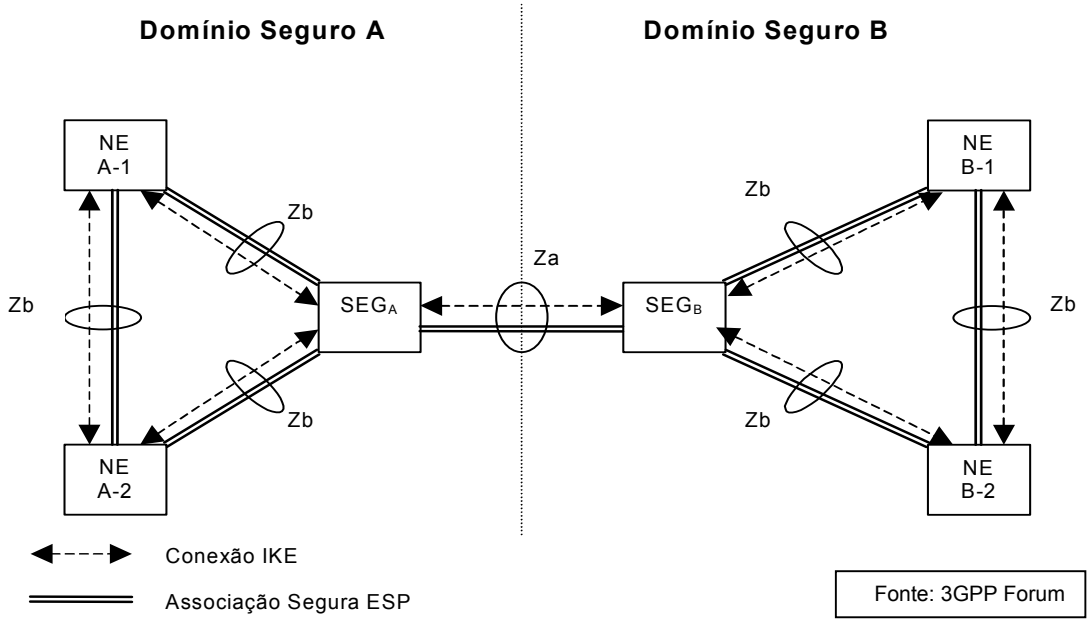

Figura 3. Arquitetura NDS para pilhas de protocolos baseadas no IP

Legenda:
- NE: Entidade de Rede (*Network Entity*)
- SEG: Roteador de Borda de Segurança (*Security Gateway*)
- IKE: Protocolo de Troca de Chaves da *Internet* (*Internet Key Exchange*)
- ESP: Protocolo de Encapsulamento de Carga Seguro (*Encapsulating Security Payload*)

Na arquitetura NDS, são definidas duas interfaces, destinadas à proteção do tráfego baseado em IP: a interface Za e a interface Zb.

Za é a interface existente entre dois SEGs, e por ela passa todo o tráfego NDS/IP entre domínios seguros (vide figura 3 acima) (RIBEIRO *et al.*, 2003a). Nessa interface, autenticação e proteção da integridade dos dados são essenciais, sendo recomendado também o uso da criptografia com esses dados. A autenticação, a proteção da integridade e a criptografia são proporcionadas pelo ESP, sendo que os SEGs empregam o IKE para negociar, estabelecer e manter um túnel seguro ESP entre si. Esse túnel é usado pelo tráfego NDS/IP entre o domínio seguro A e o domínio seguro B. Como já mencionado anteriormente, um SEG pode ser destinado a servir às comunicações com apenas um subconjunto dos domínios que o confrontam, de modo que limita-se o número de associações seguras e túneis que devem ser mantidos.

Já a interface Zb localiza-se entre NEs e SEGs, ou entre NEs pertencentes a um mesmo domínio seguro. Na arquitetura de segurança destinada a servir aos sistemas de 3ª Geração, a implementação da interface Zb é opcional. Caso seja implementada, deve operar com o ESP na autenticação e proteção à integridade dos dados, já que o uso da criptografia dentro de um mesmo domínio seguro é opcional. Esta característica deve-se à já mencionada simplificação a que a arquitetura NDS/IP nos remete, quando estudamos o conjunto da arquitetura IPSec definida pelo IETF.

2.3. A Arquitetura de Protocolos Proposta

A pilha de protocolos exibida na figura 4 representa todo o processo de troca de mensagens, tanto entre camadas de mesmo nível, através de suas SDUs (*Service Data Units*) e PDUs (*Protocol Data Units*), como entre camadas de níveis diferentes, através das respectivas primitivas de serviço. A representação da camada de aplicação destina-se a modelar a solicitação de um usuário, através das mensagens ACCEPT e DELIVER, que desencadeiam o processo de estabelecimento das associações seguras SCTP. Esse processo estende-se pelo encapsulamento das mensagens da camada de segurança de aplicação (a camada ISA) nas camadas inferiores e assim por diante, utilizando os mecanismos de segurança do IPSec.

É necessário lembrar que o diagrama da figura 4 representa apenas dois sistemas em comunicação, de forma a facilitar a visualização do mecanismo de funcionamento dos protocolos, entendendo-se que tal diagrama replica-se quando mais sistemas e entidades estão envolvidos.

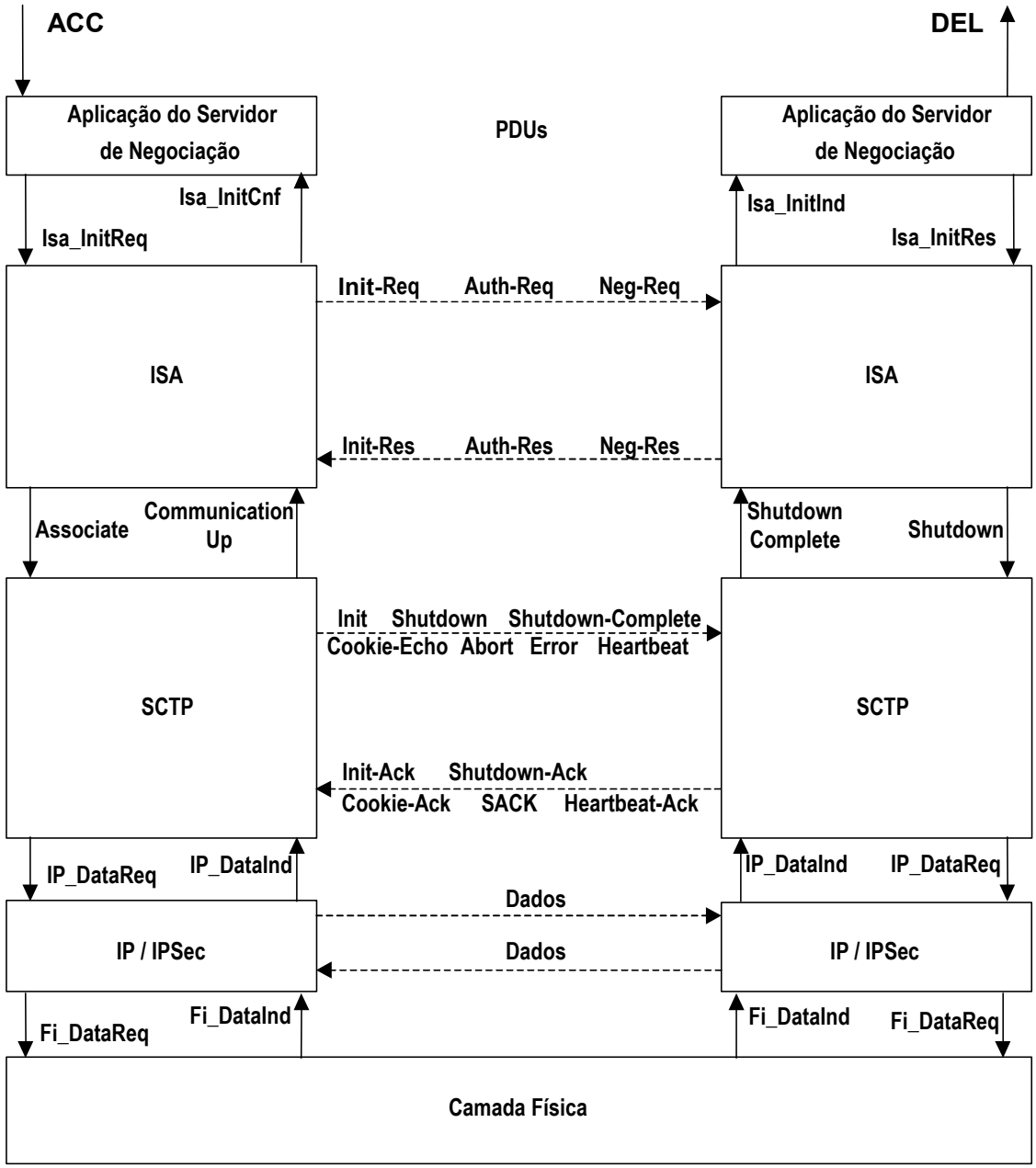

Figura 4. Pilha de protocolos utilizada pela arquitetura de segurança proposta

As camadas física e IP (esta última operando com a arquitetura IPSec), que são prestadoras de serviço a camadas superiores e, portanto, não são o objeto de nossa análise, têm suas primitivas de serviço representadas de forma simplificada. As primitivas DataReq e DataInd, respectivamente *Data Request* e *Data Indication*, representam esquematicamente todas as funcionalidades dessas camadas. No nosso modelo de trabalho de três camadas, que será apresentado no próximo capítulo, as duas camadas serão representadas como uma única entidade, denominada de Meio, com todas as funcionalidades dessas camadas.

A camada ISA é representada neste esquema como usuária dos serviços de transporte oferecidos pelo protocolo SCTP, mas não será abordada em nossa especificação. No nosso modelo de três camadas, a camada ISA e a Aplicação do Servidor de Negociação estarão englobadas numa única entidade, que recebe as mensagens ACCEPT e o DELIVER, e solicita serviços através das primitivas SCTP.

A camada que será objeto de nosso estudo será, então, a camada intermediária, a do protocolo SCTP.

2.4. O Protocolo SCTP

O SCTP é um protocolo de transporte relativamente novo, inicialmente voltado ao transporte de mensagens de sinalização telefônica, mas que, em sua forma atual, tem o potencial para suceder ao TCP e ao UDP. O SCTP atua como uma camada entre as aplicações (usuários SCTP) e um serviço de comutação de pacotes não orientado à conexão, do tipo IP. O serviço básico que presta é a transferência confiável de mensagens dos usuários entre entidades SCTP.

O protocolo SCTP é relativamente novo – a primeira definição oficial está contida no RFC 2960, de outubro de 2000 (STEWART *et al.*, 2000). Uma mudança importante no protocolo oficial, consubstanciada no RFC 3309 (STONE *et al.*, 2002), data de setembro de 2002, o que caracteriza o SCTP como um protocolo ainda em franca evolução, especialmente porque foi projetado desde o início para ser extensível.

De acordo com a sua definição básica (STEWART *et al.*, 2000), o SCTP oferece os seguintes serviços a seus usuários:

- entrega confirmada, livre de erros e não duplicada de dados de usuário;
- fragmentação de dados em conformidade com o MTU (*Maximum Transmission Unit*) descoberto do caminho;
- entrega seqüencial de mensagens de usuário em múltiplos fluxos, com opção para entrega por ordem de chegada de mensagens de usuário individuais;
- empacotamento opcional de múltiplas mensagens de usuário em um único pacote SCTP; e
- tolerância a falhas de rede através do suporte a *multi-homing* em qualquer ou ambas as extremidades de uma associação.

O SCTP tem como características distintivas principais de seus antecessores

TCP e UDP as que seguem:

- é orientado à conexão por natureza, mas a associação SCTP é um conceito mais abrangente que a conexão TCP;
- a transferência confiável envolve a associação entre dois pontos extremos SCTP;
- o SCTP provê os meios para que cada entidade SCTP (SCTP *endpoint*) forneça à outra entidade (durante o processo de início de uma associação) uma lista de endereços de transporte (múltiplos endereços IP combinados a uma porta SCTP) através dos quais essa entidade pode ser alcançada e gerar pacotes SCTP; e
- a associação espalha transferências sobre todas as combinações fonte/destino possíveis que podem ser geradas a partir das listas das entidades.

2.4.1. Associações e fluxos (*streams*)

Dois terminais estabelecem uma associação SCTP, e não uma conexão como em TCP. Isso porque cada conexão TCP apresenta apenas um fluxo *full-duplex*. Uma associação SCTP possui um número arbitrário de fluxos, acertado entre as partes na criação da associação. Cada fluxo SCTP é um canal de comunicação unidirecional, e pode haver um número desigual de fluxos em cada direção. Uma associação SCTP que pretenda simplesmente emular uma operação TCP usa dois fluxos, um em cada direção.

O SCTP não suporta um estado meio-aberto (*half-open state*), como o TCP, onde um lado continua enviando dados enquanto o outro está fechado. Quando qualquer uma das entidades executa um *shutdown*, a associação pára de aceitar novos dados do seu usuário e apenas enviará os dados enfileirados até o momento do pedido de encerramento.

2.4.2. Mensagens

A definição do SCTP prevê a transmissão de mensagens, ao invés de octetos. Isso porque cada mensagem é recebida atomicamente, como um bloco indivisível, exatamente da forma em que foi transmitida. A aplicação nunca receberá a mensagem pela metade, ou em pedaços.

Nesse aspecto, o SCTP lembra um serviço de datagrama, como o fornecido pelo UDP, só que confirmado, pois as aplicações usuárias têm a garantia da atomicidade da mensagem. Essa garantia é também oferecida pelo UDP pois nele a mensagem tem de caber dentro de um datagrama IP, que é sempre entregue integralmente ou então descartado. Já o TCP não oferece nenhuma garantia do gênero. Em TCP, os dados são transmitidos como uma simples seqüência de octetos. Cabe às aplicações interpretar essa seqüência e separar as mensagens contidas nela, baseando-se unicamente nos dados.

Entretanto, diferentemente do UDP, o SCTP é confirmado; as mensagens são entregues na ordem da transmissão e sem duplicações. As mensagens podem ter tamanho virtualmente ilimitado (até 2^{32} – 1 octetos); o protocolo encarrega-se de fragmentar a mensagem em diversos datagramas SCTP e remontá-la no terminal remoto.

Nada impede, porém, a aplicação de ignorar por completo a formatação em mensagens e tratar os dados como uma seqüência de octetos, ao estilo TCP. Mesmo assim, o desempenho bruto de transmissão do SCTP será, no mínimo, igual ao do TCP.

Quando necessário, o SCTP fragmenta as mensagens do usuário para assegurar que o pacote SCTP passado às camadas inferiores esteja em conformidade com o MTU do caminho. Na recepção, os fragmentos são remontados antes que a mensagem seja passada ao usuário.

2.4.3. Ordem das mensagens

O termo fluxo é usado em SCTP para descrever uma seqüência de mensagens de usuário que devem ser entregues ao protocolo da camada superior, ordenadamente, com respeito às outras mensagens do mesmo fluxo. O usuário SCTP pode especificar na iniciação da associação o número de fluxos a serem suportados por ela. As mensagens do usuário são associadas aos números dos fluxos (primitivas SEND e RECEIVE). Internamente, o SCTP atribui a cada mensagem recebida do usuário um número de seqüência de fluxo e, no receptor, assegura que as mensagens sejam entregues ao usuário na seqüência correta, dentro do fluxo.

No SCTP, então, ocorre o que podemos denominar de “entrega seqüencial intra-fluxo”, ou seja, as mensagens são entregues na ordem em que foram transmitidas, embora essa ordem seja respeitada apenas dentro de cada fluxo. Mensagens pertencentes a fluxos diferentes não são necessariamente entregues na

ordem da transmissão. No TCP, ao contrário, a seqüência de octetos é sempre respeitada, na ordem estrita em que foi remetida. Embora essa característica seja uma garantia importante, se várias mensagens forem transmitidas através de uma mesma conexão, a perda de um pacote IP pode atrasar a entrega de todas as demais. Por outro lado, em SCTP, pode-se usar um fluxo por requisição, ou por tipo de requisição. A perda de um datagrama atrasa apenas a entrega das mensagens de um único fluxo.

O SCTP atribui um TSN (*Transmission Sequence Number*) a cada fragmento de dados ou mensagem não-fragmentada, oriundos do usuário, independente do número de seqüência de fluxo atribuído no nível de fluxo. O receptor reconhece todos os TSN recebidos, mesmo com falhas na seqüência, de modo que a entrega confiável seja mantida funcionalmente separada da entrega seqüencial de fluxos.

A retransmissão de pacotes ocorre quando os reconhecimentos não chegam nos prazos esperados, e esta retransmissão está condicionada aos mesmos procedimentos para a prevenção de congestionamento usados no TCP.

O SCTP é um protocolo confirmado tal qual o TCP, e todos os algoritmos de reordenamento, *timeout* e retransmissão de pacotes, bem como controle de congestionamento e descoberta do MTU, estão também presentes no SCTP. Por serem de uso consagrado, a maioria dos algoritmos foi diretamente copiada do TCP, com adaptações para suporte à ordenação separada por fluxo. Muito embora o TCP dependa da perda de pacotes para detectar congestionamento, qualquer perda diminui de forma considerável o desempenho da transmissão. Por esse motivo, existe grande interesse na pesquisa de algoritmos melhores para o SCTP.

O SCTP também introduz uma variável inexistente no TCP: *multi-homing*. A existência de múltiplos caminhos, cada um com latência diferente, abre espaço para melhorias nos algoritmos de retransmissão. O SCTP permite que a aplicação altere os valores padrão de *timeout*, tornando os algoritmos mais moderados ou agressivos, recurso que pode ser utilizado para melhorar o desempenho se a aplicação conhece as características da rede em que está operando.

2.4.4. *Multi-homing*

Um recurso totalmente novo do SCTP é o suporte direto ao que se denomina como *multi-homing*. Um computador é dito *multi-homed* quando encontra-se conectado a duas ou mais redes IP, sem necessariamente ser um roteador. Particular importância tem o computador ligado por dois ou mais caminhos à *Internet* mundial, pela tolerância à falha desse tipo de enlace.

O SCTP permite que cada terminal informe uma lista de seus endereços IP na *Internet*. O IP utilizado durante a criação da associação é o IP primário, principal canal de comunicação com o terminal remoto. Os demais endereços IP informados são secundários, que podem ser utilizados caso falhe a comunicação com o IP primário. O SCTP controla o estado dos diversos endereços IP do terminal remoto por mecanismos de *heartbeat*, quando um outro tráfego de pacotes for inadequado para prover tal informação, de modo a sempre conhecer a situação de cada caminho e fazer deduções acerca do estado das conexões. O suporte direto do SCTP a *multi-homing* facilita a implantação de alta disponibilidade de rede para uma ampla gama de serviços da *Internet*.

No momento da iniciação da associação, um caminho primário é definido para cada extremidade SCTP, usado para o envio normal de pacotes SCTP. No lado receptor, o gerenciamento de caminhos é responsável pela verificação da existência de uma associação SCTP válida, a qual o pacote que entra pertence, antes de passá-lo aos processamentos adicionais.

O gerenciamento de caminhos e a validação de pacotes são realizados simultaneamente.

2.4.5. Características de segurança do SCTP

Um ponto particularmente fraco do TCP é a sensibilidade a ataques de negação de serviço (DoS, ou *Denial of Service*). O principal motivo é que o TCP não autentica as partes durante a fase de conexão, e não tem como fazê-lo pois a conexão é efetuada com apenas 3 pacotes (o que se denomina de *3-way handshake*). A única proteção do TCP contra ataques *blind spoof* é confiar que cada terminal gere números iniciais de seqüência (ISN, ou *initial sequence number*) imprevisíveis.

Outro problema do TCP é que uma conexão meio-aberta ocupa tanta memória quanto uma conexão efetiva. O lado passivo da conexão (tipicamente o servidor) precisa armazenar o ISN remetido pelo cliente, pois esse valor será utilizado quando a conexão for completada. Como em geral o TCP é implementado no *kernel* do sistema operacional, e a memória do *kernel* é limitada, o número máximo de conexões TCP simultâneas em cada sistema também terá um limite bem definido.

Assim, um invasor pode simplesmente disparar um grande número de pacotes TCP SYN com origem falseada – o que se denomina de ataque SYN *flood*. As conexões meio-abertas nunca são efetivadas e acabam descartadas por *timeout*, porém, nesse meio tempo, elas lotam a tabela de conexões e impedem a realização

de conexões legítimas. O protocolo SCTP foi, desde o início, projetado para evitar tais problemas.

Em primeiro lugar, uma associação é iniciada por um pedido do usuário SCTP (primitiva ASSOCIATE ou SEND). Um mecanismo de *cookie* (KARN, SIMPSON, 1999) é empregado durante a iniciação para prevenir ataques. Esse mecanismo usa uma autenticação de 4 vias (*four-way handshake*), permitindo às duas últimas mensagens transportar dados de usuário, para uma configuração mais rápida. O encerramento gracioso da associação também é pedido pelo usuário SCTP (primitiva SHUTDOWN), sendo possíveis outras formas, como o fechamento imediato a pedido do usuário (primitiva ABORT) ou como resultado de uma condição de erro detectada dentro da camada do SCTP.

Os pacotes envolvidos na criação da associação, listados na seqüência normal de transmissão, têm os seguintes nomes:

- INIT, do cliente para o servidor;
- INIT-ACK, do servidor para o cliente em resposta a INIT;
- COOKIE-ECHO, do cliente para o servidor, permitindo ao servidor considerar estabelecida a associação;
- COOKIE-ACK, do servidor para o cliente, confirmando o recebimento de COOKIE-ECHO, e permitindo ao cliente considerar estabelecida a associação.

Na parte fixa da estrutura do pacote SCTP, existe um parâmetro de 32 *bits* denominado etiqueta de verificação (*verification tag*). Cada associação possui duas etiquetas, uma para cada direção. Todo pacote SCTP deve apresentar a etiqueta correspondente à sua associação e direção, sob pena de ser descartado. Essa etiqueta serve primariamente para identificar instâncias diferentes de uma mesma associação (i.e. entre os mesmos terminais e os mesmos números de porta). Mas também permite discriminar facilmente pacotes forjados, eliminando assim a possibilidade de *blind spoof*, tanto na tentativa de abertura de conexões anônimas quanto na tentativa de seqüestro ou abortamento de uma conexão existente.

Nos pacotes INIT e INIT-ACK, cada parte transmite uma etiqueta de iniciação (*initiation tag*), e, dali por diante, deve repetir esse valor na etiqueta de verificação de todos os pacotes, até o fim da associação. Cada terminal calcula sua própria etiqueta, logo há duas etiquetas por associação, uma em cada sentido. O valor da etiqueta deve ser aleatório, ou pelo menos imprevisível para agentes externos, para que a proteção contra *blind spoof* seja efetiva. É o mesmo cuidado que deve ser tomado na geração do ISN do TCP, ou do TSN (*Transmission Sequence Number*) do SCTP.

O TCP possui o parâmetro TSN para indicar, em cada pacote, qual o segmento da seqüência de octetos que está sendo transmitido. Esse valor permite remontar a seqüência e detectar as lacunas e é incrementado pelo número de octetos transmitidos.

O SCTP também apresenta o TSN nos trechos de dados, porém ele é incrementado por trecho, e não por octeto, sendo a utilidade desse parâmetro a mesma que a do TCP.

Da mesma forma que no TCP, o número TSN inicial (ISN) é informado de parte a parte durante a criação da associação, e o mesmo cuidado deve ser tomado na sua geração – ele deve ser imprevisível por parte de um agente externo (tal qual a etiqueta de verificação). A diferença é que o TCP conta apenas com esse recurso para detectar ataques *blind spoof* que visem abortar ou seqüestrar uma conexão. Já o SCTP conta com o TSN e também com a etiqueta de verificação, o que o torna bem mais resistente a esse tipo de ataques.

Eliminada a ameaça representada pelo *blind spoof*, resta evitar que o lado passivo da associação reserve memória com associações meio-abertas e fique vulnerável a ataque análogo ao SYN *flood*. A solução encontrada foi transferir para o cliente a responsabilidade total pelo armazenamento dos dados de uma associação meio-aberta. Essa transferência foi conseguida no SCTP por meio do *cookie* – uma estrutura de tamanho variável, cujo formato interno só é conhecido pelo seu criador, que o servidor transmite ao cliente no pacote INIT-ACK. O cliente deve retransmitir o *cookie* ao servidor no pacote COOKIE-ECHO, como sugere o próprio nome deste pacote. Se o cliente fosse na verdade um invasor mandando pacotes forjados, o *cookie* nunca chegaria ao cliente verdadeiro, e muito menos seria devolvido ao servidor. Como o servidor não ocupa memória com associações meio-abertas, o ataque não satura a tabela de associações e não impede as associações legítimas. Ataques no estilo SYN *flood* não são possíveis contra o SCTP.

Após transmitir o pacote INIT-ACK, se o servidor não conserva nenhuma informação sobre a associação, nem sobre o potencial cliente, e a associação só é efetivada com o pacote COOKIE-ECHO, o servidor depende exclusivamente das informações contidas no *cookie* para criar a associação. Logo, embora o formato do *cookie* seja de livre escolha, o que lhe garante segurança adicional, ele tem de conter pelo menos os seguintes dados:

- os dados relevantes do pacote INIT, normalmente uma simples transcrição dos dados desse pacote;
- os dados relevantes (gerados pelo servidor) do pacote INIT-ACK,

geralmente também uma simples transcrição;

- duas etiquetas de 32 *bits* denominadas *tie tags*, que geralmente contêm o valor zero, mas podem ser eventualmente preenchidas com as etiquetas de verificação do cliente e/ou do servidor, e servem para identificar retransmissões de pacotes de criação de associação.

Os dados acima representam o mínimo necessário para a implementação funcionar, porém outros dados adicionais são necessários para garantir a segurança do SCTP:

- *timestamp*, para que *cookies* velhos possam ser descartados, bem como para proteção contra ataques de repetição (*replay attacks*);
- assinatura digital, que garanta a integridade dos dados e permita ao servidor reconhecer o *cookie* como seu, e o atrele ao cliente.

A assinatura digital é tipicamente obtida concatenando-se todos os dados do *cookie*, os endereços IP, as portas, uma chave secreta e calculando-se um *hash* de qualidade criptográfica, como MD5 ou SAH-1 (KRAWCZYK *et al.*, 1997). A RFC 2960 recomenda que a chave secreta seja trocada de forma razoavelmente freqüente (STEWART *et al.*, 2000).

A criptografia dos dados do *cookie* não é especialmente encorajada, pois não traz qualquer vantagem de segurança: os dados ali contidos poderiam ser obtidos facilmente de outras formas. O *hash* é suficiente para fins de autenticação.

Embora o *cookie* tenha tamanho variável e seja escolhido pela implementação, deve ainda assim ter o menor tamanho possível que forneça a segurança exigida, evitando problemas de interoperabilidade (prevendo que o SCTP possa ser usado em dispositivos naturalmente limitados em capacidade de memória e processamento, a RFC 2960 recomenda que a implementação crie o menor *cookie* que atenda dada necessidade de segurança). Se considerarmos 16 octetos para os dados relevantes de INIT, mais 16 para os dados de INIT-ACK, 8 para os *tie-tags*, 8 para o *timestamp* e 16 para a assinatura digital, chegamos a um *cookie* de 64 octetos.

Os pacotes SCTP contêm ainda um somatório de verificação (*checksum*), que os protege contra corrupção acidental dos dados, causada por ruído de linha ou problemas nos roteadores. Como o *checksum* do SCTP é de 32 *bits*, sua resistência à corrupção de dados é bem maior que a do TCP, com seu *checksum* de 16 *bits*. Entretanto, frise-se, o somatório não é uma assinatura digital e portanto não protege os dados contra fraude.

Por último, com relação ao tópico segurança, é bom mencionar que os mecanismos de segurança do SCTP prestam-se a evitar os ataques especificamente

previstos e mencionados acima, e não oferecem outras garantias de segurança, tais como confidencialidade ou autenticidade. Tampouco tornam o SCTP resistente ao ataque do "homem do meio", como mencionado anteriormente. Entretanto, essas garantias adicionais de segurança podem ser providas por outros protocolos utilizados em conjunto com SCTP, como, por exemplo, o IPSec e o SSL/TLS (JUNGMAIER *et al.*, 2001).

2.4.6. Operação do protocolo SCTP

Um pacote SCTP é composto de um cabeçalho comum e de pedaços ou trechos (*chunks*), que contêm informações de controle ou dados de usuário. Múltiplos trechos podem ser agrupados num pacote SCTP, até o tamanho do MTU, exceto os trechos INIT, INIT ACK e SHUTDOWN COMPLETE. No Anexo A, podem ser vistos, em detalhes, os tipos mais comuns de trechos SCTP.

O pacote SCTP possui um cabeçalho fixo de 4 parâmetros (12 octetos), mais um número variável de trechos alinhados em 32 *bits*, conforme pode ser visto na figura 5 abaixo mostrada. Determinados parâmetros do cabeçalho são invariáveis e foram mostrados em negrito na figura.

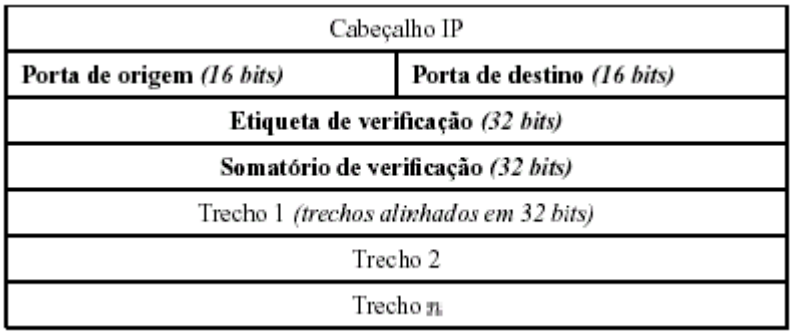

Figura 5. Representação esquemática de um pacote SCTP típico

Nos pacotes SCTP, os números de porta de origem e destino têm o mesmo fim que no TCP e UDP, ou seja, em conjunto com o número IP, permitir ao receptor identificar a aplicação destinatária do pacote e a associação SCTP à qual pertence. Como em todo protocolo da família TCP/IP, os dados que representem números inteiros devem ser representados em *network byte order* – ou seja, o primeiro octeto é o mais significativo.

Os trechos são estruturas de dados TLV (*type*, *length*, *value* – tipo, comprimento e valor), alinhadas em 32 *bits* e podem apresentar até 3 octetos de enchimento. A estrutura TLV é extremamente simples de entender, implementar e

processar, além de ser muito mais versátil que um mapa de *bits* de tamanho fixo.

Por outro lado, a grande vantagem do mapa de *bits* frente ao TLV é o seu tamanho reduzido, e conseqüente reduzido *overhead* de rede. Protocolos como IP e TCP utilizam mapas de *bits*, pois, à época de sua criação, as baixas velocidades dos enlaces justificavam qualquer esforço de diminuição do *overhead*. Entretanto, a situação atual das redes é outra - a vazão é limitada predominantemente pela latência de processamento dos terminais, e a redução dessa latência deve ser a principal preocupação dos projetistas de protocolos. Os protocolos devem ser simples, ágeis no processamento e extensíveis.

Desta forma, o SCTP utiliza principalmente estruturas TLV para transporte de dados e informações de controle. Até mesmo tarefas como abertura e fechamento de associação utilizam tais estruturas, e não *flags* (como ACK, SYN e FIN do TCP).

O valor do comprimento inclui os 4 octetos dos três primeiros parâmetros, porém não inclui o enchimento (*padding*), como se vê na figura 6 a seguir:

Tipo *(8 bits)*	*Flags (8 bits)*	Comprimento *(16 bits)*
Dados úteis do trecho, mais enchimento (32n bits)		

Figura 6. Representação esquemática de um trecho (*chunk*) SCTP típico

O parâmetro *Flags* é um mapa de *bits* interpretado de acordo com o tipo de trecho (não há nenhum *flag* com significado genérico), sendo que a maioria dos tipos de trecho não faz uso desse parâmetro. Como o SCTP foi projetado para ser extensível, pode suceder de um tipo de trecho presente em uma implementação não ser suportado por outras implementações, de modo que previu-se a reação do receptor a um trecho não suportado, dependendo do conteúdo dos dois *bits* mais significativos do tipo de trecho, conforme mostra a Tabela 1 a seguir. O valor do tipo de trecho deve, então, ser escolhido em função da reação desejada.

Tabela 1. Ação a ser adotada pelo receptor em função do tipo de trecho (*chunk*)

Tipo	*Ação tomada pelo receptor, se não suporta o trecho*
0x00 a 0x3F	descartar o pacote que contém o trecho
0x40 a 0x7F	descartar o pacote e remeter uma notifi cação de erro (vide 4.8)
0x80 a 0xBF	desprezar o trecho e continuar a processar o pacote
0xC0 a 0xFF	desprezar o trecho, continuar a processar o pacote, e remeter uma notifi - cação de erro

Os campos constituintes dos diferentes tipos de trechos empregados na definição do protocolo SCTP podem ser vistos em detalhes no Anexo A.

Dentro do espaço de dados úteis do trecho, pode haver parâmetros permanentes e opcionais. Os parâmetros permanentes, alinhados em 4 octetos, têm tamanho e ordem predefinidos para cada tipo, e sempre vêm em primeiro lugar em relação aos parâmetros opcionais. Em seguida vêm os parâmetros opcionais, em número variável, que também são estruturas TLV. Assim como nas estruturas TLV dos trechos, o comprimento do parâmetro inclui os 4 octetos do tipo e do próprio comprimento, mas não inclui os octetos de enchimento. Esse esquema pode ser visto na figura 7 a seguir.

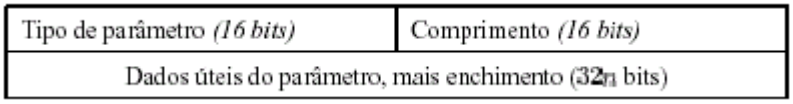

Figura 7. Representação esquemática dos dados de um trecho (*chunk*)

Analogamente ao tipo de trecho, os dois *bits* mais significativos do tipo de parâmetro também especificam a reação a um parâmetro desconhecido pelo receptor, como se vê na Tabela 2 a seguir.

Tabela 2. Ação a ser adotada pelo receptor em função do tipo de parâmetro

Tipo de parâmetro	*Ação tomada pelo receptor, se não suporta o parâmetro*
0x0000 a 0x3FFF	descartar o trecho que contém o parâmetro
0x4000 a 0x7FFF	descartar o trecho e remeter uma notifi cação de erro (vide 4.8)
0x8000 a 0xBFFF	desprezar o parâmetro e processar o trecho
0xC000 a 0xFFFF	desprezar o parâmetro, processar o trecho, e remeter uma notifi cação de erro

2.4.7. Primitivas SCTP

As requisições de serviço do protocolo da camada superior à de transporte são passadas ao SCTP através de primitivas de serviço. Da mesma forma, o protocolo SCTP pode, de forma assíncrona e eventual, notificar o protocolo da camada superior da ocorrência de diversos eventos. As primitivas de serviço e as notificações SCTP estão detalhadas em STEWART *et al.* (2000), servindo de orientação para a implementação do protocolo SCTP. Em nosso estudo, como faremos referência somente a uma parte de uma implementação viável para o protocolo SCTP,

empregaremos as seguintes primitivas:

- *Associate*, que permite ao protocolo da camada superior iniciar uma associação SCTP com uma entidade análoga de um sistema distante; e
- *Shutdown*, que solicita o encerramento da associação SCTP.

Analogamente, mencionaremos em nosso estudo apenas duas das notificações, que relacionam-se com as duas primitivas de serviço acima escolhidas, e são:

- *Communication Up*, notificação enviada à camada superior quando o SCTP encontra-se pronto para o envio ou a recepção de mensagens do usuário; e
- *Shutdown Complete*, quando os procedimentos de encerramento de uma associação SCTP são completados.

2.5. Comentários

Neste capítulo, foram apresentadas as características principais da arquitetura de 3ª Geração, respeitando-se as premissas derivadas do fórum de discussão 3G. Além disso, alguns protocolos da camada IPSec, empregados para a obtenção de uma arquitetura funcional de segurança, foram apresentados.

Quanto ao protocolo de transporte, utilizado como suporte ao serviço de troca e gerência de chave que implementa parte da arquitetura de segurança proposta, o SCTP foi o escolhido para essa nova arquitetura, pois, alinhado com as diretrizes do IPSec, já incorpora algum enriquecimento na segurança em relação ao TCP.

No próximo capítulo será apresentado o projeto do protocolo SCTP, com as simplificações que foram necessárias efetuar no modelo para a análise das formas de estabelecimento e encerramento de associações SCTP e do comportamento com o aumento de entidades envolvidas neste processo.

Capítulo 3

Projeto de uma Arquitetura de Segurança usando o Protocolo SCTP

A modelagem de certas áreas do conhecimento científico permite prever razoavelmente todas as hipóteses de funcionamento e relações de causalidade existentes entre as variáveis envolvidas, nas mais diversas condições de utilização. Um conjunto de soluções claro, conciso e completo poderá ser derivado do respectivo modelo, e será um número finito e razoável, do ponto de vista computacional. Em outros casos, como, em particular, ocorre com a modelagem do comportamento de protocolos de comunicação de dados, essas características não podem ser observadas.

De fato, mesmo sendo sistemas que se baseiam em máquinas de estados finitos, o conjunto de soluções que descreve o comportamento de um protocolo de comunicação de dados pode tender ao infinito. Em certos casos, como na análise de condições de erro ou na verificação de propriedades do protocolo, a implementação prévia com vistas à realização de testes torna-se impossível ou muito custosa de executar. No cenário ideal, a validação prévia da operação do protocolo, mesmo antes de sua implementação, deve ser realizada com o uso de ferramentas que testem todas as suas hipóteses de operação, mesmo que improváveis.

Nesse sentido, a especificação e a verificação de protocolos envolvidos nos processos de garantia de segurança deve ser orientada por técnicas de descrição

formal, empregando mecanismos e linguagens apropriados. As técnicas de descrição formal, por serem métodos de definição do comportamento de um sistema com o uso de uma sintaxe e de uma semântica, baseadas em definições matemáticas, permitem uma implementação de protocolos sem ambigüidades, precisa e completa. Além disso, oferecem uma base bem definida para a verificação e validação desses protocolos, entendidas como a avaliação de conformidade dos mesmos com relação ao comportamento esperado. De fato, a análise da quantidade dos estados e a confirmação das propriedades observacionais de um protocolo formalmente especificado permitem concluir que podemos aferir as suas propriedades de segurança e, através dos estados alcançados, verificar eventuais falhas nos procedimentos executados. As equivalências devem ser definidas para garantir que o protocolo modelado apresente, em termos observacionais, o mesmo comportamento seguro que se espera do serviço modelado (RIBEIRO *et al.*, 2003a). Além disso, deve-se verificar se possui algumas propriedades essenciais, como ser vivo e reiniciável e não conter *deadlocks*.

Neste capítulo, o item 3.1 apresenta a metodologia de análise e verificação de protocolos aplicada neste trabalho; o item 3.2 exibe a especificação do protocolo, com uma introdução à linguagem LOTOS e ao pacote de ferramentas CADP; o item 3.3 traz os comentários finais do capítulo, fazendo alusão aos resultados obtidos com as simulações a serem objeto do capítulo seguinte.

3.1. Metodologia de Trabalho

A escolha de um protocolo que atendesse às necessidades da arquitetura de segurança proposta foi o primeiro passo deste estudo. Em seguida, tendo em vista as considerações mencionadas nos parágrafos anteriores, em que se apontou a importância da utilização de um formalismo bem estabelecido na análise e verificação do protocolo em estudo, optou-se pela linguagem de descrição formal julgada apropriada e por uma ferramenta de implementação que lhe desse adequado suporte. Nesse caso, a linguagem LOTOS e o pacote CADP, como mencionado anteriormente.

O passo seguinte foi a modelagem da rede que serve de ambiente de teste, bem como do protocolo, utilizando os instrumentos da linguagem de simulação.

Uma premissa adotada neste estudo, de fundamental importância na formulação do modelo da arquitetura de segurança e do respectivo formalismo, é a de que o protocolo da camada imediatamente inferior àquela em estudo lhe prestava os

serviços necessários e corretos, sem problemas ou erros, para o seu bom funcionamento. Da mesma forma, o protocolo da camada imediatamente superior àquela em estudo recebia desta os serviços que se deseja modelar. Assim, cada um dos protocolos pertencentes a essas camadas recebeu um tratamento em separado, dividindo-se o problema em várias partes. As interações entre os protocolos das diferentes camadas ocorreriam somente através das primitivas de serviço e pela troca de PDUs e SDUs entre si, conforme já mencionado.

Com base nas premissas básicas mencionadas no parágrafo anterior, supomos que a camada inferior à do protocolo SCTP (camada IP) presta-lhe os serviços necessários de forma correta e não nos deteremos na sua análise. De forma análoga, a camada superior (Aplicação) recebe os serviços oferecidos pelo SCTP, de transporte seguro de dados e sinalização, não sendo também objeto de análise.

3.2. A Técnica de Descrição Formal nas Comunicações Seguras

Antes de uma especificação formal de qualquer dos protocolos, faz-se necessário apresentar os conceitos básicos da linguagem de especificação formal LOTOS e das ferramentas que a implementam.

As técnicas de descrição formal são métodos para a definição do comportamento de um sistema de processamento de informações qualquer, empregando uma linguagem com sintaxe formal e semântica, ao invés de faze-lo utilizando uma linguagem natural, como o Português, por exemplo. Assim, tornam possível a captura do comportamento funcional e das propriedades de sistemas, e, em particular, de protocolos de segurança, sendo essenciais à produção de documentação sem ambigüidades (RIBEIRO *et al.*, 2003b). A implementação de protocolos sem ambigüidades, precisa e completa, provê uma base bem definida para a verificação e a validação desses protocolos, entendidas como a avaliação de conformidade dos mesmos com relação ao comportamento esperado (BOLOGNESI, BRINKSMA, 1987), (FERNANDEZ *et al.*, 1996), (LOGRIPPO *et al.*, 1992).

3.2.1. A Linguagem LOTOS

LOTOS é uma técnica de descrição formal surgida como um produto do

esforço de padronização do modelo OSI (*Open Systems Interconnection*) pela ISO (*International Standards Organization*). É empregada na descrição formal de sistemas abertos, e sua fundamentação teórica, conceitos, regras de inferência e equivalências de comportamento encontram-se bem estabelecidos em BOLOGNESI e BRINKSMA (1987). Seu projeto foi motivado pela necessidade de uma linguagem que oferecesse alto nível de abstração sobre uma forte base matemática, alinhada com o esforço de obter descrição precisa para os padrões integrantes do modelo OSI (LOGRIPPO *et al.*, 1992). Os modelos em LOTOS permitem o uso de várias técnicas de validação e verificação, e diversas ferramentas foram desenvolvidas para a automatização destes processos.

Como os projetistas da linguagem LOTOS perceberam, na época de sua concepção, nenhum formalismo até então existente era suficientemente genérico para expressar de forma conveniente tanto a componente de controle como a componente de dados de uma especificação (LOGRIPPO *et al.*, 1992). Desta forma, foi LOTOS concebida pela união de dois elementos formais: a parte de dados da linguagem é baseada na teoria de tipos de dados algébricos abstratos, mais especificamente na linguagem de especificação ACTONE (MEER *et al.*, 1992); e a parte de comportamento é baseada na abordagem que a álgebra de processos faz da concorrência, combinando características das linguagens CCS (MILNER, 1989) e CSP (HOARE, 1985).

A premissa fundamental de LOTOS é a de que sistemas nada mais são do que um conjunto de processos, os quais interagem e trocam dados entre si e com o ambiente - cada especificação deve representar a relação temporal entre estas interações, caracterizando o comportamento externamente observável do sistema (BAGATELLI, 2003).

Os tipos de dados abstratos empregados em LOTOS descrevem os valores que os dados podem assumir e as operações que sobre eles atuam, sem especificar como são representados e manipulados na memória, o que contribui para a abstração inerente à linguagem (FERNANDEZ *et al.*, 1996).

Os tipos básicos de dados em LOTOS, tais como *Boolean* e *NaturalNumber*, estão especificados em uma biblioteca padrão, sendo que, para toda especificação de protocolo em LOTOS, com passagem de parâmetros, é necessário que se construa a biblioteca contendo os parâmetros que farão parte do intercâmbio de mensagens pelo protocolo. Esta biblioteca é a responsável por todas as PDUs e SDUs trocadas pelo protocolo. Além disso, é necessário avaliar o tipo de dados que se quer enviar e inserir nesta biblioteca (BAGATELLI, 2003).

No aspecto do comportamento, um sistema concorrente é descrito como uma coleção de processos que evoluem em paralelo e interagem por meio de *rendezvous* (ou pontos de encontro), conceito denominado de “sincronização”. O comportamento de cada processo é especificado com o uso de uma álgebra de operadores e os processos podem manipular e trocar dados em pontos de interação específicos, denominados *gates* (ou portas) (BOLOGNESI, BRINKSMA, 1987), o único meio de interação entre sistemas concorrentes, visto que seus espaços de memória são disjuntos. A Tabela 3, mostrada a seguir (RIBEIRO *et al.*, 2003b), apresenta alguns dos operadores LOTOS mais relevantes.

Uma expressão de comportamento em LOTOS consiste em um comportamento básico, tal como *stop* ou *exit*, bem como em ações e expressões combinadas por meio de operadores (prefixo, escolha, habilitação, composição paralela, etc.). A componente de processos permite a representação de conceitos primitivos de sistemas concorrentes (composição em paralelo e em seqüência, escolha não determinística, interrupção, dentre outros), descrevendo assim as mais diversas interações, com base nos fundamentos de álgebra de processos. Tais interações podem também permitir a troca de valores de dados entre os diferentes processos e o ambiente externo. A semântica adotada combina características das linguagens CCS e CSP.

Tabela 3. Operadores LOTOS

OPERADOR	INTERPRETAÇÃO
P !V ?Y:T; A	Interação pela porta P, enviando um valor V e recebendo uma variável Y de valor T, com execução da ação A
A [ ] B	Executa A ou B
A |[h,i,j]| B	Executa A e B em paralelo, com sincronização nas portas h,i,j
A ||| B	Executa A ou B em paralelo, sem sincronização
Exit	Termina com sucesso
Stop	Parada do processo
P [h,i,j] (H,I,J)	Chamada do processo com parâmetros das portas h,i,j e parâmetros de valores H,I,J

3.2.2. O Pacote de Ferramentas CADP

Para automatizar a análise do comportamento de um protocolo, descrito através duma especificação LOTOS, utilizamos o pacote CADP (FERNANDEZ *et al.*, 1996). O CADP oferece um conjunto integrado de funcionalidades, que vão da simulação interativa até a verificação exaustiva baseada em modelos. Este processo pode ser visualizado na figura 8 (RIBEIRO *et al.*, 2003b).

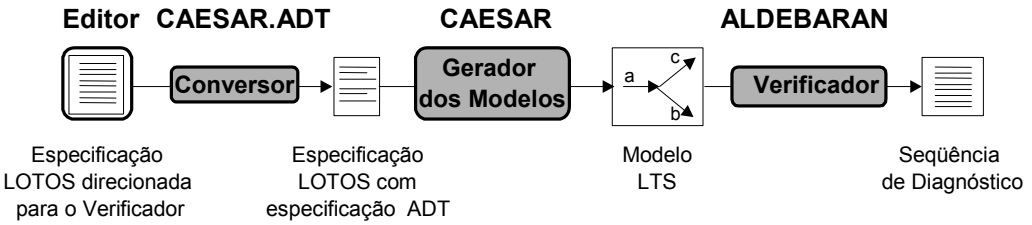

Figura 8. Processo de análise empregando as ferramentas do pacote CADP

Legenda: - LTS: Sistema de Transições Rotuladas (*Labeled Transition System*)

As funcionalidades apresentadas pelo CADP podem ser reunidas sob três grandes grupos:

- compilação de especificações descritas em LOTOS;
- verificação de sistemas comunicantes;
- validação e teste de protocolos.

Quando da compilação das especificações descritas em LOTOS, o primeiro passo consiste na tradução da especificação em LOTOS, seja para um programa escrito na linguagem C (tanto para execução como simulação), seja para um Sistema de Transições Rotuladas (LTS). Tal tradução se dá tanto na parte do comportamento da especificação em LOTOS (utilizando-se a ferramenta CAESAR) como na parte de descrição de tipos abstratos de dados (com o uso da ferramenta CAESAR.ADT). Diversas propriedades dos protocolos, como número de estados e transições, podem ser observadas, e a exibição e edição dos grafos de acessibilidade e redução de estados pode ser realizada com o conjunto de ferramentas BCG (*Binary Coded Graphic*, ou Gráfico Codificado em Binário): BCG_DRAW, BCG_EDIT, BCG_IO, BCG_MIN, BCG_OPEN. O Gráfico Codificado em Binário é um formato para representação do Sistema de Transições Rotuladas que utiliza a representação binária com técnicas de compressão que reduz o LTS em 20 vezes, se comparado a uma versão baseada em ASCII.

Já na verificação de sistemas comunicantes, a partir da obtenção de sistemas de transições rotuladas, pode-se promover a verificação de várias relações de

equivalência (bissimulação forte, equivalência observacional, equivalência de segurança, dentre outras), de acordo com algoritmos de verificação fornecidos pela ferramenta ALDEBARAN. Esta ferramenta possui avaliadores de lógica temporal e de equivalências, que permitem comparar se o LTS de um protocolo está compatível com o do seu respectivo serviço, além de especificar as propriedades e classificar cada protocolo.

Na definição da linguagem LOTOS, BOLOGNESI e BRINKSMA (1987) diferenciam a especificação da implementação. Na primeira, inserem-se as descrições de alto nível do comportamento desejado do sistema, como seriam observadas externamente. A segunda é uma descrição mais detalhada de como o sistema opera ou de como é construído, a partir de componentes mais simples. Como LOTOS é uma linguagem que permite a especificação de sistemas em diferentes níveis descritivos, dadas duas especificações sintaticamente homogêneas S1 e S2, S2 será uma implementação da especificação S1 quando S2 fornecer uma descrição mais detalhada e estruturada do sistema especificado por S1. Assim, a equivalência observacional, que é baseada na idéia de que o comportamento de um sistema é determinado pela forma como interage com os observadores externos, permite não só provar que uma implementação é correta em relação à uma dada especificação, mas também substituir subsistemas complexos por outros, mais simples e, entretanto, equivalentes.

Um exemplo dessa idéia, apresentado em BOLOGNESI e BRINKSMA (1987), é a equivalência entre a especificação dum serviço prestado pela camada N da arquitetura OSI e a implementação obtida pela composição das entidades de um protocolo da camada N com o serviço prestado pela camada N-1.

Na validação e teste de protocolos, através da ferramenta XTL (*Executable Temporal Language*, ou Linguagem Temporal Executável), permite-se a inserção de operadores lógicos temporais. XTL é uma linguagem de programação funcional, desenhada para permitir a implementação de vários operadores de lógica temporal, que são avaliados sobre um LTS codificado em BCG. A linguagem XTL define tipos especiais para um conjunto de estados, transições e rótulos do LTS e os operadores são aplicados sobre sistemas de transições rotuladas, possibilitando a verificação de propriedades nos protocolos especificados.

Além destas, o pacote CADP contempla ainda outras ferramentas, dedicadas primordialmente à promoção de ambiente gráfico de interface com o usuário (EUCALYPTUS, XSIMULATOR, CAESAR.INDENT).

Na análise de sistemas comunicantes, a partir da obtenção de sistemas de

transições rotuladas, pode-se promover a verificação de várias relações de equivalência (como a equivalência forte e equivalência observacional, dentre outras).

A equivalência observacional, como visto acima, prevê que todo comportamento externamente observado de determinado processo pode ser igualmente realizado por uma ou mais ações de outro processo. De fato, a relação entre diferentes descrições em LOTOS de um dado sistema e, em particular, entre especificações (serviço) e implementações (protocolo), pode ser estudada usando esta noção de equivalência, oriunda da linguagem CCS. Essa equivalência é baseada na idéia de que o comportamento de um sistema é determinado pelo modo pelo qual ele interage com os observadores externos.

Para a equivalência forte, toda ação interna de um processo deve ser igualmente equiparada a uma ação interna de outro processo. A equivalência forte, entretanto, é muito forte do ponto de vista de verificação de programas. Não leva em consideração critérios de abstração, especialmente o conceito de ações internas ou não observáveis (MILNER, 1989) (FERNANDEZ *et al.*, 1996). Como a especificação do serviço é uma modelagem do comportamento externamente observável do protocolo, não contém necessariamente as mesmas transições nem passa pelos mesmos estados internos que o protocolo modelado. Esta equivalência pode ser utilizada para verificar a relação entre implementações incrementais dos diferentes protocolos, e não é observável quando se comparam a implementação do protocolo com a de seu serviço, visto que diferentes ações serão executadas em cada caso.

Uma outra propriedade empregada é a que permite a redução de grafos de sistemas de transições rotuladas, de acordo com relações de equivalência forte. Este método foi utilizado para representar a essência do comportamento do protocolo, retirando redundâncias e permitindo uma análise mais simples de convergência.

3.2.3. Análise de Comportamento

Uma análise do comportamento dos protocolos que garantem a segurança de um sistema deve basear-se nos seus requisitos. Para isso, estes devem ser adequadamente especificados, sem ambigüidades, e a análise de protocolos usando técnicas de descrição formal contribui com esse esforço de se atestar a segurança de um sistema e compará-lo à sua especificação. A especificação de um protocolo empregando entidades confiáveis e não confiáveis torna-se viável devido à flexibilidade, em LOTOS, dos tipos de dados abstratos (GERMEAU, LEDUC, 1997).

A estrutura da especificação é composta por vários processos que interagem

entre si através das portas de comunicação existentes no protocolo. Cada entidade envolvida é modelada pelo processo que descreve o seu exato comportamento. No entanto, este processo pode acarretar modelos infinitos, sendo assim, necessário efetuar alguma simplificação. Esta simplificação é viável pela limitação do número das entidades envolvidas (RIBEIRO *et al.*, 2003b).

A especificação formal permite, de um modo abstrato, a obtenção de todos os detalhes dos mecanismos de segurança. Assim, pode-se focalizar a atenção somente nos serviços realmente seguros. A verificação de que todos os eventos que ocorrem no protocolo são seguros atesta a segurança deste protocolo.

3.3. Estudo do Protocolo SCTP

O protocolo SCTP é a base do serviço de transporte confiável, necessário ao atendimento dos requisitos de troca de chaves criptográficas da arquitetura em estudo. Durante o processo de troca de chaves, mediado pelo SCTP, os SEGs realizam o intercâmbio de informações para o estabelecimento de associações seguras através das interfaces Za.

A complexidade das mensagens de cada estabelecimento das associações SCTP (*four-way handshake*) influencia positivamente a segurança do sistema, se comparada àquela obtida com o uso do TCP. O grande problema encontrado é que, quanto maior a robustez de um sistema, mais complexa é sua implementação. Achar este equilíbrio é cada vez mais uma necessidade (RIBEIRO *et al.*, 2003a).

Para a análise a que se propõe este trabalho, foi preciso introduzir algumas simplificações no protocolo estudado, modelando-se apenas uma parte de seus mecanismos. Além disso, neste estudo também desprezou-se a atuação dos *timers* T1-init e T2-shutdown, por simplicidade de implementação. Detalhes desses dois elementos do protocolo SCTP podem ser vistos em STEWART *et al.* (2000).

Da mesma forma, outras simplificações foram feitas neste estudo, sem prejuízo da qualidade da análise realizada. Tais simplificações têm a ver com procedimentos internos do protocolo, não observáveis quando se busca analisar as interações entre as entidades. Por exemplo, a geração aleatória dos valores das etiquetas de verificação (*Verification Tags*), a criação do bloco de controle de transmissão, ou TCB (*Transmission Control Block*) e as ações de criptografia que levam à criação do código de autenticação da mensagem, ou MAC (*Message Authentication Code*), são alguns desses procedimentos internos que foram objeto de simplificação ou não chegaram a

ser abordados na modelagem.

Um modelo de três camadas, visto na figura 9, foi adotado para a análise do SCTP.

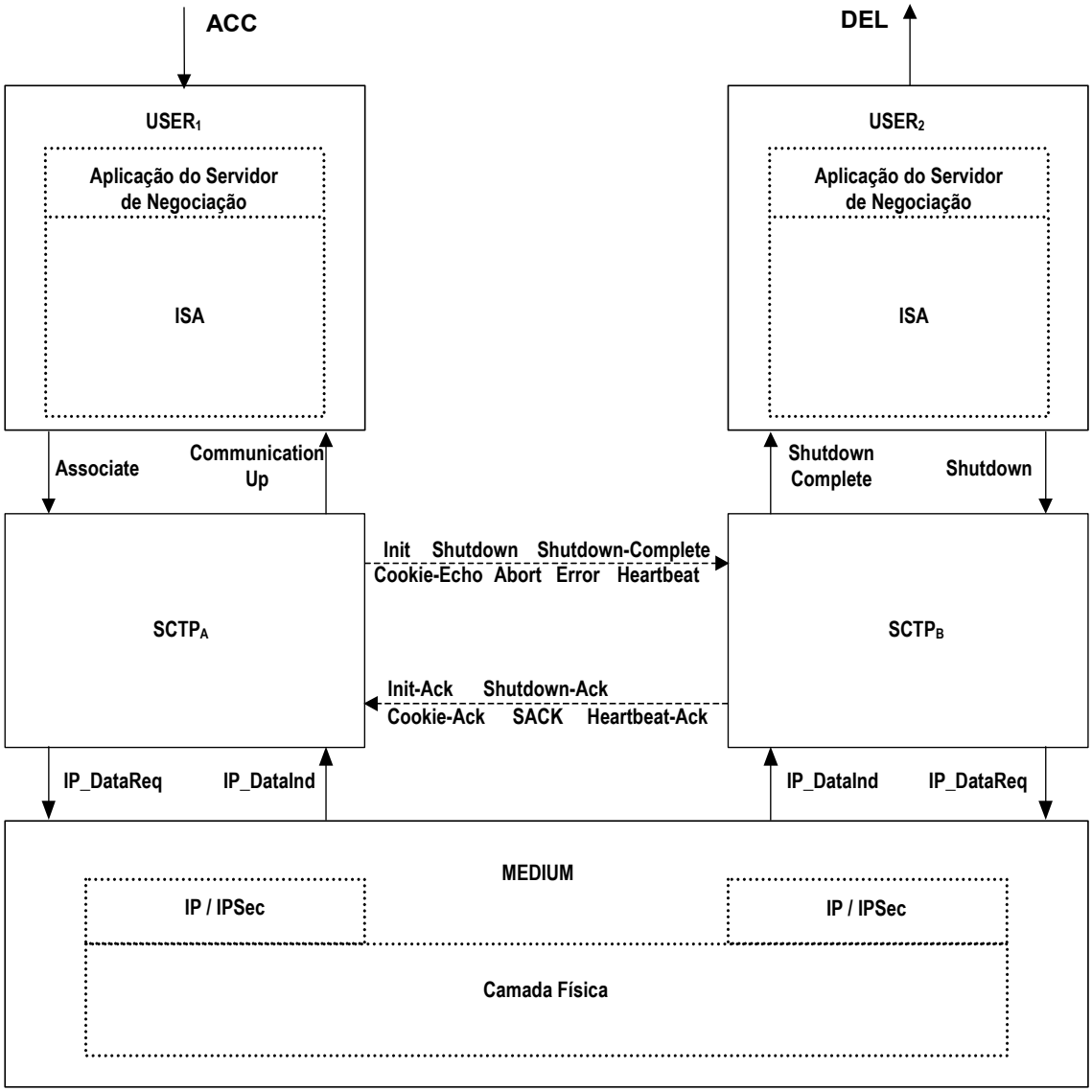

Figura 9. Modelo de três camadas da arquitetura de segurança proposta

Esse modelo é uma versão simplificada da pilha de protocolos mostrada na figura 4 e reduz o número de entidades estudadas ao necessário para analisar corretamente as interações. As entidades usuárias, a Aplicação do Servidor de Negociação e a camada ISA, foram englobadas numa só entidade, denominada Usuária (*User*). Já as camadas IP e Física, prestadoras de serviço à camada SCTP, foram englobadas na entidade Meio (Medium). Essa camada opera com primitivas simplificadas *Data Request* e *Data Indication*, respectivamente representando a solicitação de envio de uma PDU à entidade distante e a informação da chegada

dessa PDU.

3.3.1. Modelagem em LOTOS do Estabelecimento de uma Associação SCTP

Nas especificações elaboradas, foram definidos processos que implementam partes do protocolo SCTP, interligados por meio de comunicações simples e executados por dois roteadores de borda seguros (SEGs). Estes roteadores trocam as mensagens apresentadas.

O processo de estabelecimento de uma associação SCTP está mostrado esquematicamente na figura 10 a seguir, sem previsão de erros ou colisões. Em primeiro lugar, cada entidade SCTP está no estado Fechado (*Closed*). O cliente solicita uma associação SCTP pelo envio de um trecho de controle INIT, contendo um valor de etiqueta de verificação (Tag_c) gerado aleatoriamente. O pacote SCTP em si contém uma etiqueta de verificação com valor 0.

O servidor responde à solicitação e cria um TCB contendo todos os dados relativos à associação. Usando uma função *hash* e uma chave secreta, é criado um MAC para esse TCB. O TCB e o MAC são combinados e enviados ao cliente num trecho INIT-ACK na forma de um *cookie*, e o pacote é marcado com Tag_c. Um valor aleatoriamente gerado de etiqueta de verificação (Tag_s) é enviado como parâmetro do trecho INIT-ACK. O servidor permanece no estado Fechado, sem, ainda, manter qualquer dado relativo à associação ou reservar recursos para esta.

Após a recepção do INIT-ACK, o cliente retorna o *cookie* ao servidor no trecho COOKIE-ECHO, usando Tag_s no pacote SCTP. O TCB validado com o MAC é utilizado para verificar que foi o servidor que criou o *cookie* inicialmente. Sendo assim, o servidor então atribuirá recursos à associação e mudará o seu estado para Associação Estabelecida (*Established*). O cliente recebe esta informação através do envio de um trecho COOKIE-ACK, mudando igualmente seu estado para Associação Estabelecida e iniciando a transferência de dados.

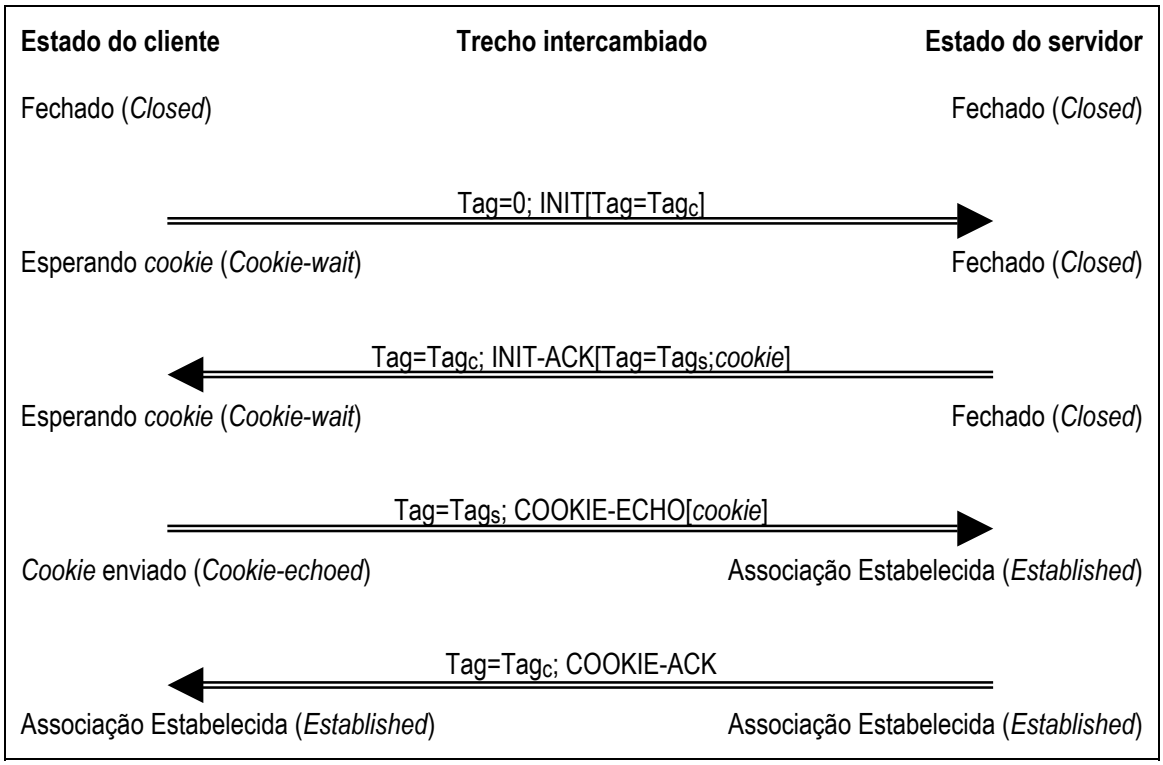

Figura 10. Esquema de estabelecimento de uma associação SCTP

Pela forma como é estabelecida uma associação, pode-se perceber o incremento à segurança oferecido pelo protocolo SCTP. O uso de um *cookie* gerado pelo servidor, enviado ao cliente e por este retornado, aliado ao fato de que recursos só são alocados após a validação desse *cookie* retornado, tornam o SCTP robusto contra ataques *DoS*. Da mesma forma, o uso das etiquetas de verificação (*Verification Tags*), escolhidas no início do processo e incluídas em todos os pacotes SCTP, tornam extremamente difícil a inserção de conteúdos maliciosos numa associação já estabelecida sem a correta interceptação de pacotes SCTP. Além disso, a quebra da codificação de 32 *bits* é impraticável se bons geradores de números aleatórios forem empregados, e pacotes cujos *Tags* sejam incorretos são logo descartados (JUNGMAIER *et al.*, 2001).

Essa troca de mensagens foi especificada, respeitando a arquitetura de modelagem em LOTOS do protocolo SCTP, pela descrição do comportamento esperado neste modo de operação. No trecho de especificação LOTOS visto abaixo, pode-se ver como fica a modelagem do comportamento da atividade de estabelecimento de uma associação SCTP. Esta especificação, em sua forma completa, encontra-se no Anexo B.

specification SCTP_Associate [ACC, IP_DataReq1, IP_DataInd1, IP_DataReq2,

```
IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit

library
   SCTP_LIB
endlib

behaviour

   hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2, SCTP_AssocReq,
SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
   (
    (
     (((
     USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
     |||
     USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
     |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
     (
     SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1] (INIT
of pdu_type)
     |||
     SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
     )))
    )
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )

where

process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA:
data_type) : noexit :=

(...)
```

Os processos SCTP_A e SCTP_B são aqueles que emulam o comportamento do protocolo SCTP, quando do estabelecimento de uma associação SCTP, e controlam as trocas de PDUs e dados ocorridas no processo, conforme se pode ver nos trechos abaixo:

```
process SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(PDU: pdu_type) : exit :=

   SCTP_AssocReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(TAG, INIT)
```

```
  where

    process SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

      IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
      IP_DataReq1 !DATA !PDU;
      SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (COOKIE, COOKIE_ECHO)

      where

        process SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

          IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
          IP_DataReq1 !DATA !PDU;
          SCTP_CommunUp !DATA;
          SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(INIT)

        endproc

    endproc

endproc

process SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(PDU: pdu_type) : exit :=

  IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
  IP_DataReq2 !DATA !PDU;
  SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2](COOKIE, INIT_ACK)

  where

    process SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

      IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
      SCTP_AssocInd !DATA;
      SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (TAG, COOKIE_ACK)

      where

        process SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

          SCTP_AssocAck ?DATA:data_type;
          IP_DataReq2 !DATA !PDU;
          IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
          SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
```

(INIT_ACK)

endproc

endproc

endproc

(...)

Foram empregadas as ferramentas do pacote CADP para a modelagem, sendo os compiladores CAESAR.ADT e CAESAR utilizados para a geração dos LTS, o BCG_INFO para a exibição do número de estados, transições e rótulos obtidos e o BCG_DRAW para a visualização do grafo de acessibilidade que exibe a evolução do comportamento do protocolo. A ferramenta BCG_MIN permitiu a redução dos estados, pela observação da existência de equivalências internas, e a ALDEBARAN efetuou a comparação do protocolo com o seu serviço modelado, quanto às equivalências observacional e forte. O serviço modelado para a criação da associação SCTP pode ser visto no item 3.3.3.

3.3.2. Modelagem em LOTOS do Encerramento Normal de uma Associação SCTP

O encerramento de uma associação SCTP pode ocorrer por iniciativa de qualquer uma das entidades comunicantes. Esse encerramento poderá ser abrupto, ou seja, abortivo, quando quaisquer dados pendentes serão descartados, ou normal, por definição gracioso, quando todos os dados pendentes serão transmitidos por ambas as partes antes da desconexão.

O encerramento abrupto dá-se pelo uso da primitiva de serviço ABORT, enquanto o normal emprega a primitiva SHUTDOWN. Em particular, abordamos neste trabalho o encerramento via primitiva SHUTDOWN, mais complexo.

Na figura 11, podemos verificar as etapas seguidas nessa operação de encerramento. Quando do recebimento da primitiva SHUTDOWN, oriunda da camada superior, a entidade SCTP entra num estado denominado de Encerramento Pendente (*Shutdown-pending*), nele permanecendo até que todos os dados pendentes tenham sido enviados e confirmados pela entidade correspondente do sistema distante. Nesse estado, a entidade SCTP não aceitará nenhum novo dado vindo da camada superior.

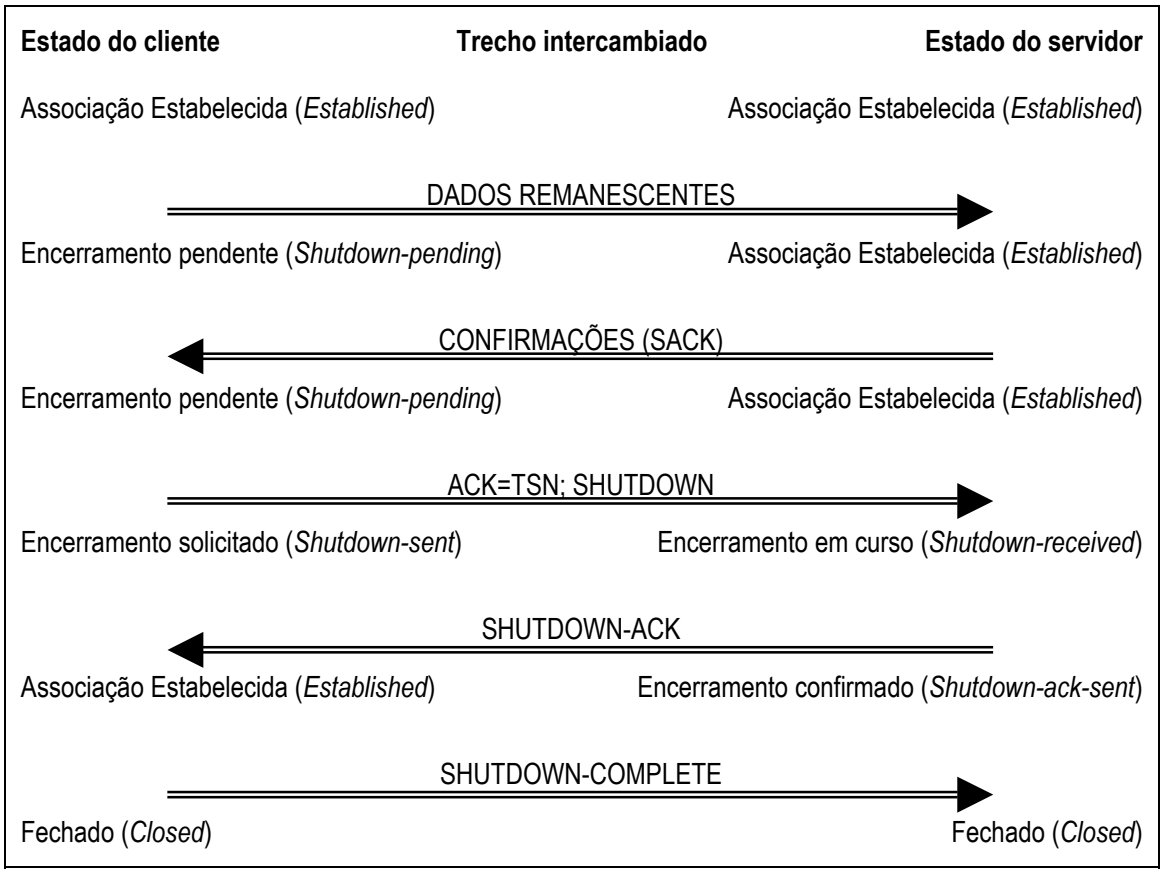

Figura 11. Esquema de encerramento de uma associação SCTP

Como dito anteriormente, as mensagens foram sincronizadas respeitando a arquitetura de modelagem em LOTOS do protocolo SCTP, na busca de descrever a expressão de comportamento esperado para este modo de operação. A especificação do encerramento de uma associação SCTP pode ser vista no trecho a seguir, sendo o restante encontrado no Anexo B.

```
specification SCTP_Shutdown [ACC, IP_DataReq1, IP_DataInd1, IP_DataReq2,
IP_DataInd2, SCTP_ShutdReq, SCTP_ShutdCompl, SCTP_ShutdInd,
SCTP_ShutdAck, DEL] : noexit

library

  SCTP_LIB

endlib

behaviour

 hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2, SCTP_ShutdReq,
SCTP_ShutdCompl, SCTP_ShutdInd, SCTP_ShutdAck in
  (
   (
```

```
    (((
    USER_1 [ACC, SCTP_ShutdReq, SCTP_ShutdCompl, DEL] ({} of data_type)
    |||
    USER_2 [SCTP_ShutdInd, DEL, ACC, SCTP_ShutdAck] ({} of data_type)
    )
    |[SCTP_ShutdReq, SCTP_ShutdCompl, SCTP_ShutdInd, SCTP_ShutdAck]|
    (
    SCTP_A [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1, IP_DataInd1]
(SHUTDOWN of pdu_type)
    |||
    SCTP_B [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2, IP_DataInd2]
(SHUTDOWN_ACK of pdu_type)
    )))
   )
   |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
  )

where

process USER_1 [ACC, SCTP_ShutdReq, SCTP_ShutdCompl, DEL] (DATA:
data_type) : noexit :=

(...)
```

A descrição da troca de mensagens é feita nos processos SCTP_A e SCTP_B, onde podemos observar a especificação de cada mensagem, com seu respectivo tipo de dados transmitido, no trecho abaixo:

```
process SCTP_A [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1, IP_DataInd1]
(PDU: pdu_type) : exit :=

  SCTP_ShutdReq ?DATA: data_type;
  IP_DataReq1 !DATA !PDU;
  SCTP_SendShut [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1,
IP_DataInd1] (TAG, SHUTDOWN)

  where

    process SCTP_SendShut [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1,
IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

      IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
      IP_DataReq1 !DATA !PDU;
      SCTP_SendShutCompl [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1,
IP_DataInd1] (TAG, SHUTDOWN_COMPLETE)

      where

        process SCTP_SendShutCompl [SCTP_ShutdReq, SCTP_ShutdCompl,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=
```

```
            IP_DataInd1 ?DATA:data_type ?PDU: pdu_type;
            IP_DataReq1 !DATA !PDU;
            SCTP_ShutdCompl !DATA;
            SCTP_A [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1, IP_DataInd1]
(SHUTDOWN)

         endproc

    endproc

endproc

process SCTP_B [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2, IP_DataInd2]
(PDU: pdu_type) : exit :=

  IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
  IP_DataReq2 !DATA !PDU;
  SCTP_SendShutAck [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2,
IP_DataInd2](TAG, SHUTDOWN_ACK)

  where

    process SCTP_SendShutAck [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2,
IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

       IP_DataInd2 ?DATA:data_type ?PDU: pdu_type;
       SCTP_ShutdInd !DATA;
       SCTP_ShutdAck ?DATA:data_type;
       IP_DataReq2 !DATA !PDU;
       SCTP_B [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2, IP_DataInd2]
(SHUTDOWN_ACK)

    endproc

endproc

(...)
```

3.3.3. Especificação do Serviço

A expressão do comportamento do serviço é idêntica nos dois modos de operação, e serve como base para a verificação das propriedades de equivalência observacional.

```
specification SCTP_AssociateService [ACC, DEL] : noexit

library

  SCTP_LIB
```

```
endlib

behaviour

    SCTP_AssocServ [ACC, DEL]

where

process SCTP_AssocServ [ACC, DEL] : noexit :=

  ACC;
  DEL;
  SCTP_AssocServ [ACC, DEL]

endproc

endspec
```

3.4. Estudo do Protocolo SCTP com Aumento no Número de Entidades Envolvidas

Após a análise do estabelecimento e do encerramento das associações SCTP, foi necessário avaliar o comportamento do protocolo com o aumento do número de entidades envolvidas. Para isto, foi utilizado o mesmo processo de estabelecimento de associações, só que envolvendo mais que duas entidades. Em particular, uma delas (o usuário 1) foi escolhida como sendo a iniciadora dos processos de estabelecimento das associações, que foram estabelecidas com 2, 3, 4 e 5 outras entidades análogas simultaneamente. O processo utilizado nesta etapa visa verificar a variação do comportamento do protocolo modelado com o aumento do número de associações e entidades envolvidas.

Na especificação elaborada, foram definidos processos que implementam o sincronismo entre as entidades modeladas. Todos os processos foram interligados por meio de comunicações simples e executados com 2, 3, 4 ou 5 roteadores de borda seguros (SEGs).

A arquitetura empregada para o estabelecimento de comunicações é descrita em LOTOS pela mesma expressão de comportamento do protocolo já apresentada no item 3.3.1.

A modelagem em LOTOS que representa o comportamento geral do protocolo na interação de vários SEGs (um para cinco) pode ser vista no trecho abaixo:

```
specification SCTP_Associate6Seg [ACC, IP_DataReq1, IP_DataInd1, IP_DataReq2,
IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit

library

  SCTP_LIB

endlib

behaviour

  hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2, SCTP_AssocReq,
SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
  (
   (
    (
     (
      USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
      |||
      USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
     |||
     (
      USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
      |||
      USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
     |||
     (
      USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
      |||
      USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
     |||
     (
      USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
      |||
      USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
     |||
     (
      USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
      |||
      USER_6 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
    )
    |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
    (
     SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1] (INIT
of pdu_type)
     |||
     SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
```

```
    )
   )
   |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
  )
```

where

process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA: data_type) : noexit :=

(...)

3.5. Comentários

Este capítulo explicou como as especificações formais dos protocolos e de seus respectivos serviços foram submetidas a testes das propriedades como a equivalência observacional e a equivalência forte, se eram vivos e reiniciáveis, se possuíam *deadlocks*, etc. Todas essas características são condições essenciais para o correto funcionamento do protocolo, e habilitaram o prosseguimento dos experimentos.

Como mencionamos anteriormente, concluiu-se que os modelos do protocolo SCTP apresentados e suas respectivas descrições de serviço apresentam equivalência observacional, demonstrando que o protocolo especificado atende aos requisitos do serviço. Os modelos foram submetidos à verificação de equivalência forte, mas o fato de não terem satisfeito este último teste já era esperado, pois naturalmente não foram descritas as ações internas na especificação do serviço. Além disso, o protocolo foi sempre testado quanto à possibilidade de haver *deadlocks*, se era vivo e reiniciável, condições essenciais para o prosseguimento do projeto.

Estes testes e procedimentos foram feitos em todas as fases do projeto do protocolo SCTP e, em cada experimento, além do incremento em complexidade, foram examinadas todas as propriedades e condições anteriormente mencionadas. Após a análise do estabelecimento e do encerramento das associações SCTP, foi necessário avaliar o comportamento do protocolo com o aumento do número de entidades envolvidas. Para isto, foi utilizado o mesmo processo de estabelecimento de associações, só que envolvendo mais que duas entidades. Uma delas, denominada de usuário 1, foi escolhida como sendo a iniciadora dos processos de estabelecimento das associações, que foram estabelecidas com 2, 3, 4 e 5 outras entidades análogas simultaneamente. O processo utilizado nesta etapa visa verificar a variação do

comportamento do protocolo modelado com o aumento do número de associações e entidades envolvidas.

Diversas simulações foram feitas para melhor visualizar e acompanhar cada protocolo, bem como o comportamento de seus gráficos de estados e transições. Dada a complexidade da especificação dos protocolos e o conseqüente consumo de recursos computacionais, desenvolveu-se uma metodologia incremental para a descrição dos mesmos, até o ponto onde não foi mais possível obter redução do número de estados e transições observadas, o que ocorreu no exame das interações de uma entidade com cinco outras.

Estas ressalvas e conclusões estão melhor detalhadas no próximo capítulo, que apresenta os resultados obtidos, enquanto as especificações completas encontram-se no Anexo B.

Capítulo 4

Verificação e Simulação do Protocolo SCTP

A verificação e simulação do protocolo SCTP serão apresentados neste capítulo. Descreveremos os experimentos realizados com o protocolo proposto e suas especificações mostradas no capítulo anterior.

A busca de uma especificação que reflita da forma mais realista possível o funcionamento de um protocolo obriga à utilização de um processo gradual de descrição destas funcionalidades. Num primeiro momento, prepara-se uma descrição simples e genérica, contando somente com as características básicas de funcionamento do protocolo, possibilitando posteriormente a inclusão de funcionalidades mais específicas e complexas.

A obtenção dos resultados de cada simulação proporciona mecanismos mais eficazes para implementações mais complexas. Esta complexidade refere-se à qualidade da descrição dos detalhes internos do protocolo, das trocas de mensagens executadas, das especificações dos nós (SEGs) e do aumento do número destes na rede. A agregação de complexidade através do aumento do detalhamento funcional torna as especificações cada vez mais completas. Esta abordagem tem o objetivo de obter uma validação eficaz do protocolo com relação ao sistema 3G.

Assim, neste capítulo, mostramos os resultados obtidos com a simulação da operação do protocolo SCTP no estabelecimento de uma associação, no encerramento normal desta e no estabelecimento simultâneo e paralelo de diversas associações de um SEG com outros SEGs (de dois até cinco).

Um breve histórico, resumindo os experimentos realizados nesta etapa, pode ser descrito como a seguir:

- O primeiro experimento consistiu na simulação de duas entidades de borda (dois SEGs), envolvidas no estabelecimento de uma associação SCTP, trocando as quatro mensagens componentes do *four-way handshake*;
- O segundo experimento consistiu na modelagem do respectivo serviço de estabelecimento de uma associação SCTP, de modo a verificar a equivalência observacional entre o protocolo modelado e o serviço proposto;
- No terceiro experimento, o encerramento normal (*Shutdown*) de uma associação SCTP foi modelado, envolvendo novamente dois roteadores de borda (SEGs);
- O quarto, o quinto, o sexto e o sétimo experimentos envolveram, respectivamente, três, quatro, cinco e seis entidades na modelagem de processos de estabelecimento de associações SCTP, atuando um desses SEGs sempre como o iniciador (cliente) e os demais como receptores (servidores).

Em cada um desses experimentos, foi elaborada uma especificação em LOTOS, submetida à compilação com as ferramentas CAESAR.ADT (parte de dados) e CAESAR (parte de comportamento). Os resultados obtidos foram visualizados na forma gráfica com o uso da ferramenta BCG_DRAW, para avaliar a correta convergência do protocolo, a ausência de *deadlocks*, e as propriedades de ser vivo e reiniciável. Em seguida, empregando a ferramenta BCG_MIN, e através da bissimulação forte, o LTS foi reduzido e visualizado novamente com o BCG_DRAW. A ferramenta BCG_INFO permitiu obter informações, antes e depois da redução, acerca do número de estados, transições e rótulos obtidos.

No segundo experimento, propriedades adicionais foram verificadas, como a equivalência observacional e a forte, através da comparação entre o protocolo (implementação) e o seu respectivo serviço (especificação).

Neste capítulo, o item 4.1. apresenta a verificação e simulação do protocolo SCTP no estabelecimento das associações e no encerramento normal de uma associação SCTP; no item 4.2, apresentamos a verificação e simulação do comportamento do protocolo no estabelecimento de associações com o aumento do número de SEGs envolvidos, tomando um deles como o iniciador das associações e até cinco outros como seus pares; o item 4.3 apresenta os comentários finais sobre os aspectos de interesse das simulações.

4.1. Simulação do Estabelecimento e do Encerramento de uma Associação SCTP

As simulações com o protocolo SCTP seguem lógica incremental de complexidade, apresentando primeiramente uma descrição básica do protocolo, levando em consideração os mecanismos originais (STEWART *et al.*, 2000), e, a partir destes, realizando modificações no modelo que culminassem em uma especificação coerente com o sistema de 3ª Geração.

A modelagem na linguagem LOTOS permitiu que as trocas de mensagens entre os dispositivos em um meio de comunicação fossem alteradas de forma eficaz, graças ao grau de abstração conseguido com o seu uso.

A ferramenta CADP foi de grande importância para a produção dos resultados que possibilitaram a definir a forma de funcionamento do protocolo modelado. As mensagens e campos necessários para uma comunicação eficiente foram simuladas. Estas simulações se detiveram em determinar a melhor forma de estabelecimento da associação SCTP sem se preocupar, num primeiro momento, com variações de comportamento derivadas da mudança do número de dispositivos envolvidos, tópico que será abordado no item 4.2.

4.1.1. Simulação do Estabelecimento de uma Associação SCTP

A simulação da operação do protocolo SCTP no ato de estabelecimento de uma associação tomou como base o diagrama mostrado na figura 10. Além disso, o diagrama de estados mostrado na figura 12 a seguir também representa a operação mencionada. As conclusões serão apresentadas no decorrer do texto com auxílio de tabelas e gráficos.

O estabelecimento de uma associação SCTP foi descrito no item 3.3.1. As simulações representam uma troca de mensagens básicas, utilizando partes da estrutura original do protocolo SCTP, conforme descrito em STEWART *et al.* (2000). A primeira simulação foi feita com dois SEGs, um denominado USER_1 (que atua como cliente) e o outro, USER_2 (o servidor). Os papéis de cliente e servidor são intercambiáveis em LOTOS, bem como a operação é bidirecional, sendo a escolha desses papéis de mero valor didático e útil para simplificar a especificação. Os processos USER_1 e USER_2, então, representam os usuários das camadas superiores à de transporte, que recebem os serviços da camada SCTP.

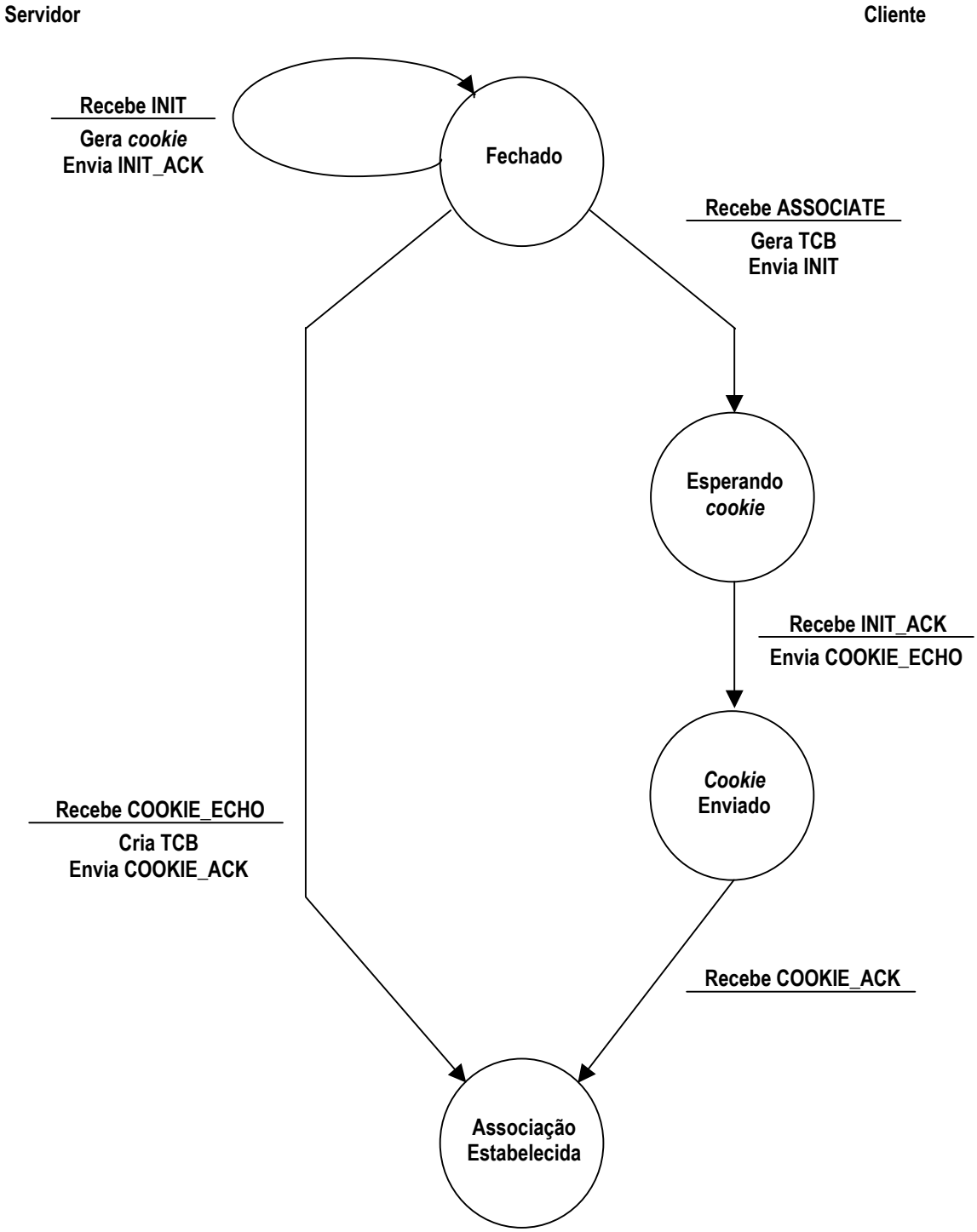

Figura 12. Diagrama de estados do estabelecimento de uma associação SCTP

Os resultados das simulações do protocolo levaram à criação de um gráfico de estados e transições. Este gráfico foi obtido pelas ferramentas do CADP, em particular o BCG_DRAW. Este gráfico, mostrado na figura 13 a seguir, representa o número de

estados e transições obtido para o protocolo, 58 estados e 66 transições, neste caso.

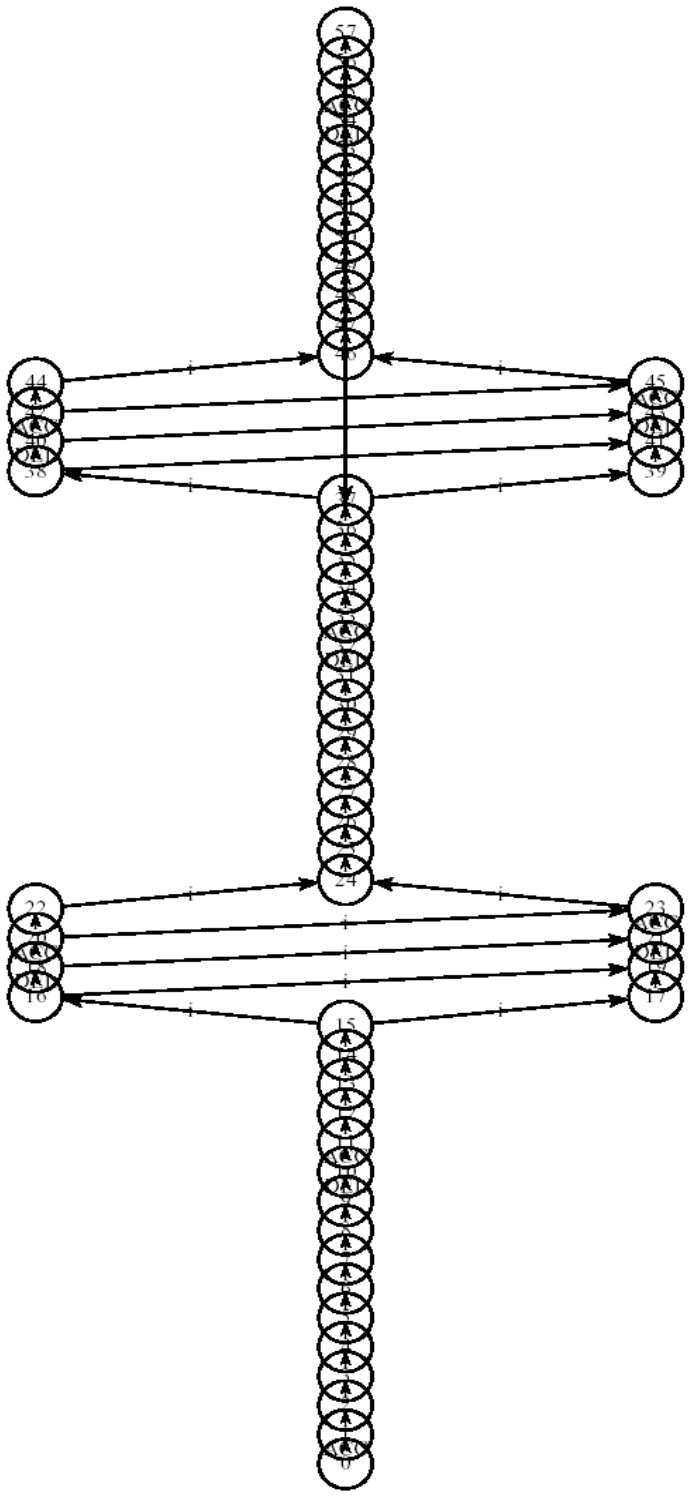

Figura 13. Gráfico de estados e transições do protocolo SCTP no estabelecimento de uma associação

A propriedade de redução, oferecida pela ferramenta BCG_MIN do pacote CADP, possibilitou uma análise mais qualitativa dos resultados, e seu gráfico

correspondente pode ser visto na figura 14.

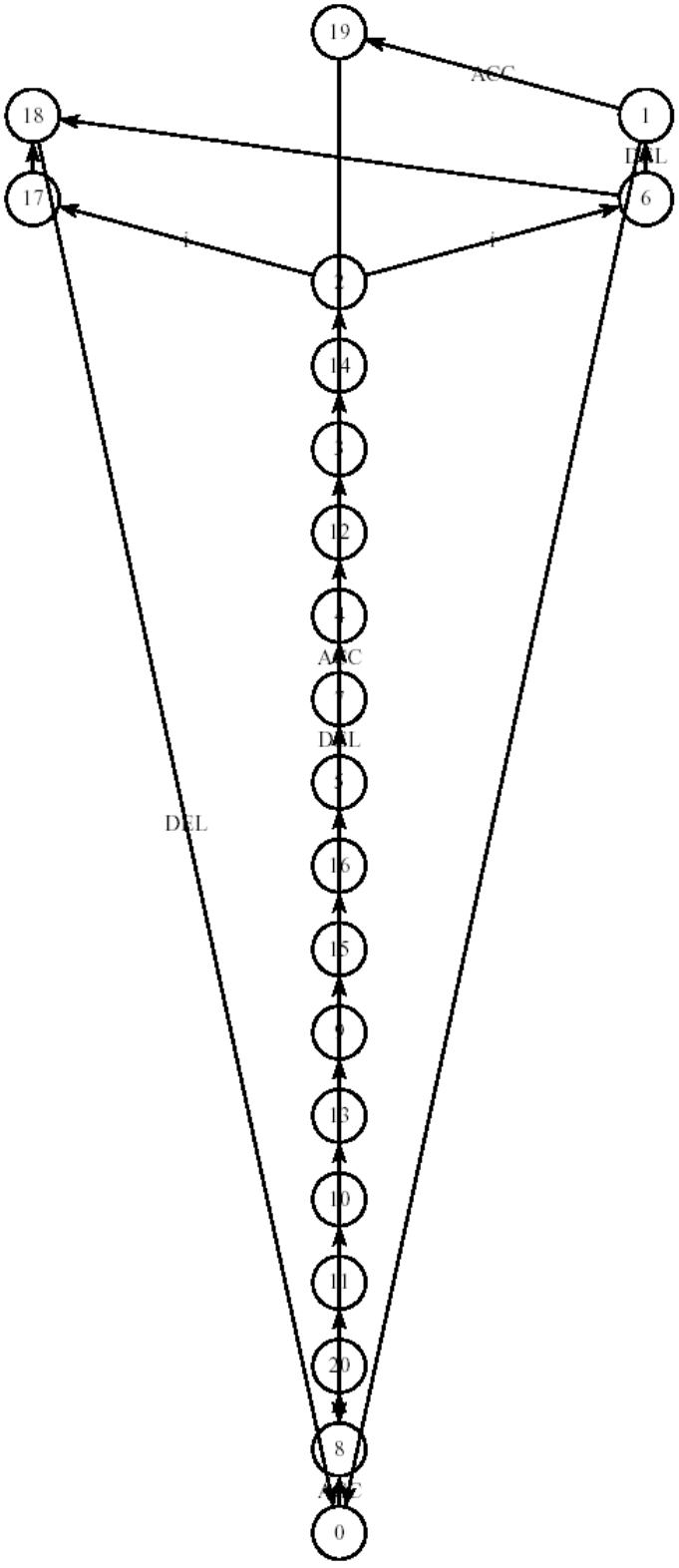

Figura 14. Gráfico de estados e transições reduzidos do protocolo SCTP no estabelecimento de uma associação

Através da operação de redução, descartam-se as informações redundantes que possuem mesmo comportamento, mesma origem e mesmo destino, através de equivalência forte (*strong bisimulation*), e, com isso, o número de estados e transições foi reduzido para 21 e 24, respectivamente, como se vê na Tabela 4.

Tabela 4. Número de estados, transições e rótulos no estabelecimento de uma associação SCTP

Tipo de especificação	Número de Estados	Número de Transições	Quantidade de Rótulos
Normal	58	66	3
Reduzida	21	24	3

Os gráficos demonstram a coerência da descrição formal e a confirmação das propriedades observacionais, bem como a ausência de *deadlocks*. Algumas simulações realizadas não geraram resultados que contribuíssem de forma significativa para o projeto e por isso não foram citadas.

4.1.2. Simulação do Serviço

A simulação de um protocolo, para a verificação de suas propriedades e de seu comportamento, deve ser comparada com a simulação de um serviço que lhe é equivalente. Na verdade, o serviço é a expressão externa do comportamento (esperado) do protocolo. A especificação do serviço equivalente ao estabelecimento de uma associação SCTP foi mostrada no item 3.3.3, e os resultados de sua simulação podem ser vistos no gráfico da figura 15 e na Tabela 5.

Tabela 5. Número de estados, transições e rótulos no serviço de estabelecimento de uma associação SCTP

Tipo de especificação	Número de Estados	Número de Transições	Quantidade de Rótulos
Normal	2	2	3
Reduzida	N/A	N/A	N/A

Legenda: N/A – não aplicável; os resultados da redução do serviço são idênticos aos do serviço não reduzido

Figura 15. Gráfico de estados e transições do serviço SCTP no estabelecimento de uma associação

De maior relevância que os resultados acima são as comparações que podem ser feitas entre o comportamento do protocolo e o do respectivo serviço. Tais comparações, conforme mencionado no capítulo anterior, são denominadas de "equivalências" em LOTOS, sendo elas a observacional e a forte. A equivalência observacional, esta a mais relevante para os fins de nossa análise, verificou-se, como pôde ser comprovado empregando a ferramenta Aldebáran do pacote CADP. A equivalência observacional indica que todo comportamento externamente observado de determinado processo (o serviço) pode ser igualmente realizado por uma ou mais ações de outro processo (o protocolo, neste caso). Desta forma, pode-se afirmar que o protocolo modelado apresenta, para um observador externo, exatamente o mesmo comportamento do serviço respectivo.

Já a equivalência forte indica que toda ação interna de um processo deve ser igualmente realizada por uma ação interna de outro processo, de modo que, como seria de se esperar, na comparação entre serviço e protocolo, não se verifica essa equivalência.

4.1.3. Simulação do Encerramento Normal de uma Associação SCTP

Na simulação do protocolo SCTP quando do recebimento de um pedido de encerramento de associação gracioso, mostramos somente as etapas posteriores ao envio de todos os dados pendentes. Em outras palavras, em nossa especificação, consideramos que não há dados pendentes a enviar (todos já foram enviados, antes mesmo do pedido de encerramento da associação). Essa é uma simplificação razoável, visto não termos como objetivo deste item analisar o envio e a recepção de dados.

Adicionalmente à consideração feita no parágrafo anterior acerca da existência de dados pendentes, as premissas que foram adotadas no item 4.1.1 aplicam-se ao disposto neste item. A figura 11, mostrada no item 3.3.2, indica claramente os estados pelos quais passam cliente e servidor, bem como que PDUs trocam entre si as entidades SCTP_A e SCTP_B que simulam o protocolo em operação.

O comportamento do encerramento de uma associação SCTP pode ser verificado através do gráfico de estados e transições da figura 16. Pelo mesmo motivo apresentado no item 4.1.1, o de facilitar a visualização, é necessário reduzir o número de estados e transições. Para esta modalidade de funcionamento, foram obtidos os estados e transições apresentados na Tabela 6, antes e depois da redução. É interessante observar que, dado que o encerramento de uma associação, conforme aqui especificado, é um processo mais simples que seu estabelecimento, a redução daquele processo não resultou em significativa redução de estados. O gráfico da figura 17, onde se vêem estados e transições após a redução, demonstra a coerência funcional da descrição e a confirmação das propriedades observacionais.

Tabela 6. Número de estados, transições e rótulos no encerramento de uma associação SCTP

Tipo de especificação	Número de Estados	Número de Transições	Quantidade de Rótulos
Normal	20	29	3
Reduzida	19	28	3

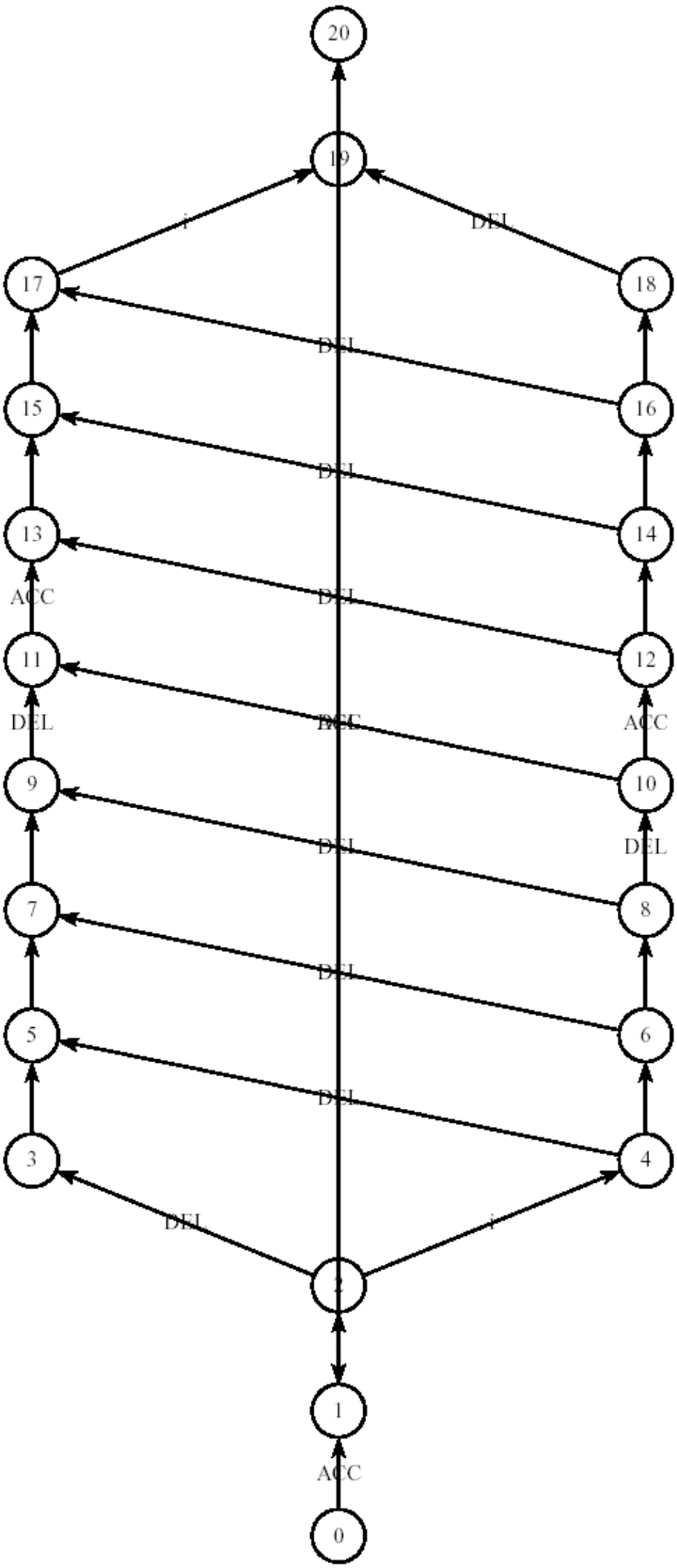

Figura 16. Gráfico de estados e transições do protocolo SCTP no encerramento de uma associação

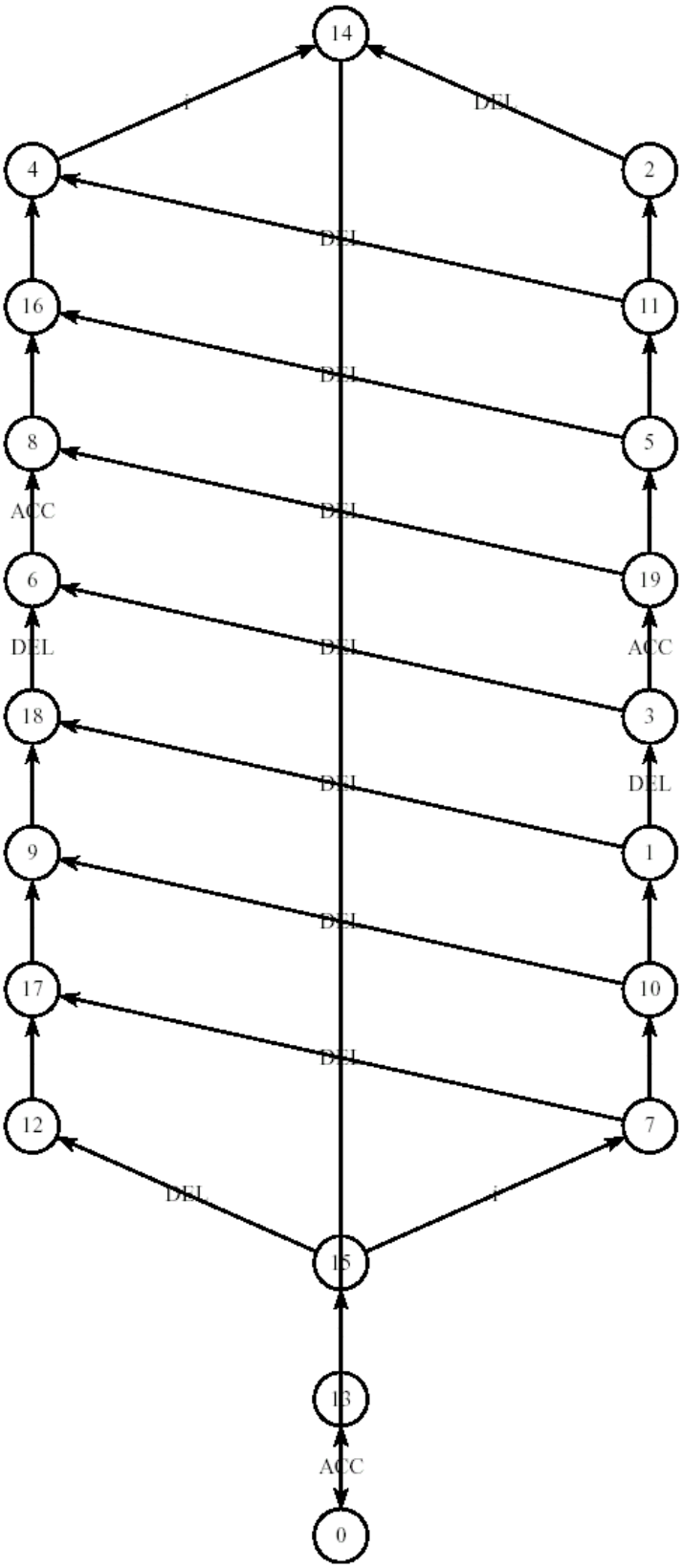

Figura 17. Gráfico de estados e transições reduzidos do protocolo SCTP no encerramento de uma associação

Nas simulações que resultaram desta última especificação, foram feitos testes de lógica interprocessos, entre as mensagens. Cada ramo da árvore de comportamento que é apresentada pelo gráfico apresenta uma alternativa de rota para as mensagens entre os SEGs, todas com origem no SEG emissor (USER_1). Estas estruturas de decisão de envio e recebimento de mensagens foram testadas tendo como objetivo verificar a possibilidade de haver falhas como, por exemplo, *deadlocks*. Algumas simulações não geraram resultados que contribuíssem de forma significativa para o projeto e por isso foram descartadas.

Observando o comportamento apresentado pelo protocolo SCTP neste ponto do projeto, podemos afirmar que já está bem estabelecido e consolidado. Os mecanismos de troca de mensagens estão bastante robustos para garantir que ele convergirá, mesmo para uma quantidade maior de SEGs interagindo. Para verificar isto, seria necessária apenas a repetição de alguns dos processos já testados nestas simulações, aumentando-se o número de entidades envolvidas, o que será feito no item 4.2 a seguir.

4.1.4. Resumo dos Resultados Obtidos

Os resultados obtidos com a verificação do comportamento do protocolo SCTP, no estabelecimento de uma associação, em relação ao modelo do serviço, e quanto ao encerramento de uma associação, foram sintetizados na Tabela 7.

Por se tratar de modos diferentes de execução de um mesmo protocolo, a grande diferença encontrada entre o estabelecimento de uma associação SCTP e o seu encerramento é a expansão do comportamento dos estados e rótulos que determinam o grau de vulnerabilidade dos modos em relação ao comportamento das mensagens. Podemos perceber que o estabelecimento de uma associação apresenta um número maior de estados e transições, realizando o serviço proposto ao protocolo de forma mais complexa em relação ao encerramento de uma associação, o que é razoável de se supor, visto ser o término de uma associação momento menos que apropriado para uma tentativa de invasão ou interceptação. O processo de estabelecimento de uma associação SCTP é por si só mais complexo, por ser um *four-way handshake*, enquanto no encerramento normal apenas três pacotes são trocados, após enviados todos os dados pendentes.

A Tabela 7 apresenta o resumo das simulações, contendo as seguintes informações de cada simulação:

- Característica da simulação;
- Número total de SEGs interagindo, quer sejam emissores, ou receptores ou ambos;
- Número de estados gerado pela ferramenta responsável pela obtenção do grafo LTS;
- Número de transições gerado pela ferramenta responsável pela obtenção do grafo LTS;
- Número de rótulos gerado pela ferramenta responsável pela obtenção do grafo LTS;
- Grafo reduzido, que indica se o grafo LTS apresenta ou não estados e transições redundantes.

Tabela 7. Resumo das simulações relevantes executadas

Característica da simulação	Número total de SEGs	Número de Estados	Número de Transições	Número de Rótulos	Grafo Reduzido
Serviço SCTP	2	2	2	3	Não
Estabelecimento de associação	2	58	66	3	Não
Estabelecimento de associação	2	21	24	3	Sim
Encerramento de associação	2	20	29	3	Não
Encerramento de associação	2	19	28	3	Sim

A equivalência observacional entre o protocolo executando o estabelecimento de uma associação e seu respectivo serviço modelado foi verificada. A equivalência forte, pelo motivo já exposto, não foi verificada integralmente, pois existem diferenças significativas entre a evolução do diagrama de estados do protocolo e o do serviço modelado. Diversos estados internos surgem no modelo mais completo (protocolo), que não têm correspondentes no modelo mais simples (serviço).

Para todas as especificações, foi testada a existência de *deadlocks*, se o protocolo era vivo e reiniciável e se respeitava as propriedades observacionais. Todas as simulações convergiram para este tipo de abordagem, cabendo, então, passar-se a outra fase do projeto, que seria a verificação do comportamento do protocolo com o aumento do número de dispositivos que trocam informações entre si para o estabelecimento das associações.

4.2. Simulação com o Aumento de Entidades

Para completar a verificação da especificação do protocolo de associações seguras foi necessário a verificação do comportamento em relação à variação do número de dispositivos estabelecendo associações. Como mencionado anteriormente, todas as simulações foram feitas com um dos SEGs interagindo com os demais, de forma simultânea. Os SEGs apresentam todas as funcionalidades necessárias, tanto para a transmissão quanto para a recepção das mensagens.

Após a alteração da especificação do protocolo, com a introdução de mais um SEG, o número de estados e transições tende progressivamente ao infinito. Começando-se as simulações com três SEGs, chegou-se ao máximo alcançado neste estudo, que foram seis SEGs estabelecendo associações simultâneas. Neste ponto, dado o número enorme de transições e estados obtido, a operação de redução não mais foi capaz de simplificar o LTS, de modo que tornou-se improdutivo adicionar mais SEGs à análise.

4.2.1. Simulação da Interação envolvendo três ou mais SEGs

Para a simulação com três SEGs, foram modelados três usuários (USER_1, USER_2 e USER_3), utilizando as mesmas entidades SCTP_A e SCTP_B. Um dos usuários (USER_1), que atua como o cliente e iniciador das associações SCTP, troca mensagens com os outros dois (USER_2 e USER_3), que agem como seus servidores. O gráfico de estados e transições obtido, embora mostre a evolução da implementação descrita anteriormente, confirmando a convergência da simulação, e que o protocolo é vivo, reiniciável e não possui *deadlocks*, apresenta-se visualmente muito complexo para que sua análise seja de algum valor. Por esse motivo, não foi sequer apresentado neste trabalho.

Enquanto a análise de um gráfico com tamanha quantidade de estados e transições torna-se visualmente inviável, o gráfico de estados e transições da figura 18, por outro lado, mostra a evolução da implementação após a redução de estados, de modo que torna-se mais útil como ferramenta de análise.

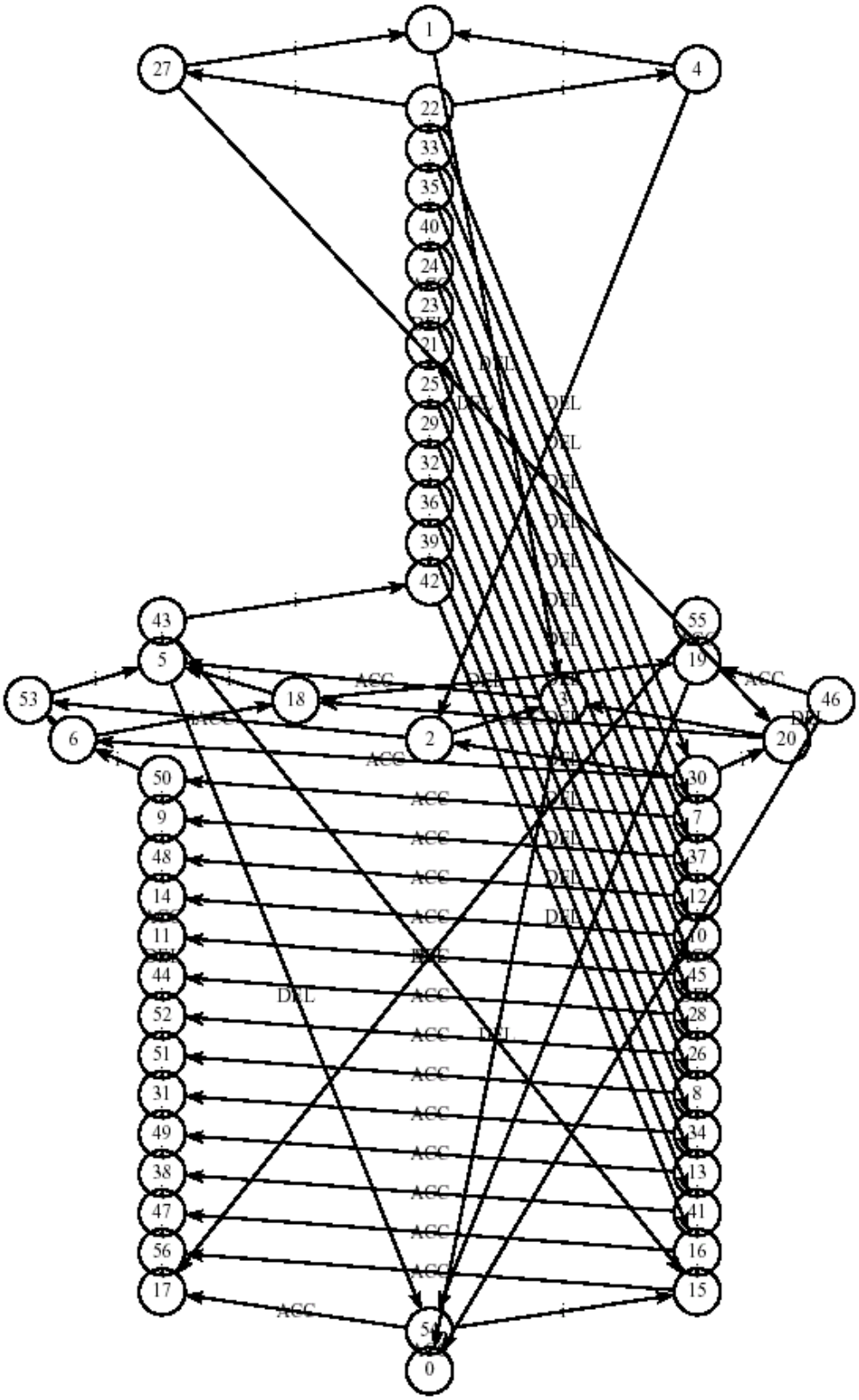

Figura 18. Gráfico de estados e transições reduzidos do protocolo SCTP no estabelecimento de duas associações simultâneas

Devido a isto, a partir da interação envolvendo três SEGs, abandonamos a exibição do gráfico de estados e transições antes da operação de redução. A comparação do número de estados e transições no processo de estabelecimento de associações SCTP envolvendo três SEGS, antes e depois da redução, pode ser vista na Tabela 8.

Tabela 8. Número de estados, transições e rótulos no estabelecimento de duas associações SCTP simultâneas

Tipo de especificação	Número de Estados	Número de Transições	Quantidade de Rótulos
Normal	2008	3764	3
Reduzida	57	99	3

As figuras seguintes, 19 e 20, mostram, respectivamente, os gráficos de estados e transições obtidos com a interação envolvendo quatro e cinco SEGs. A interação entre seis SEGs, como mencionado, embora convergente, não pôde ser reduzida pela ferramenta BCG_MIN, de modo que não se pôde igualmente exibir um gráfico de estados e transições reduzidos. Como última etapa de interesse deste item, entretanto, seus dados foram incluídos na Tabela 9, que resume todas as simulações realizadas com mais de dois SEGs envolvidos.

Tabela 9. Resumo das simulações relevantes executadas, envolvendo mais de dois SEGs

Característica da simulação	Número total de SEGs	Número de Estados	Número de Transições	Número de Rótulos	Grafo Reduzido
Estabelecimento de associação	3	2.008	3.764	3	Não
Estabelecimento de associação	3	57	99	3	Sim
Estabelecimento de associação	4	44.504	115.866	3	Não
Estabelecimento de associação	4	110	226	3	Sim
Estabelecimento de associação	5	815.492	2.706.158	3	Não
Estabelecimento de associação	5	180	405	3	Sim
Estabelecimento de associação	6	13.542.444	54.486.056	3	Não

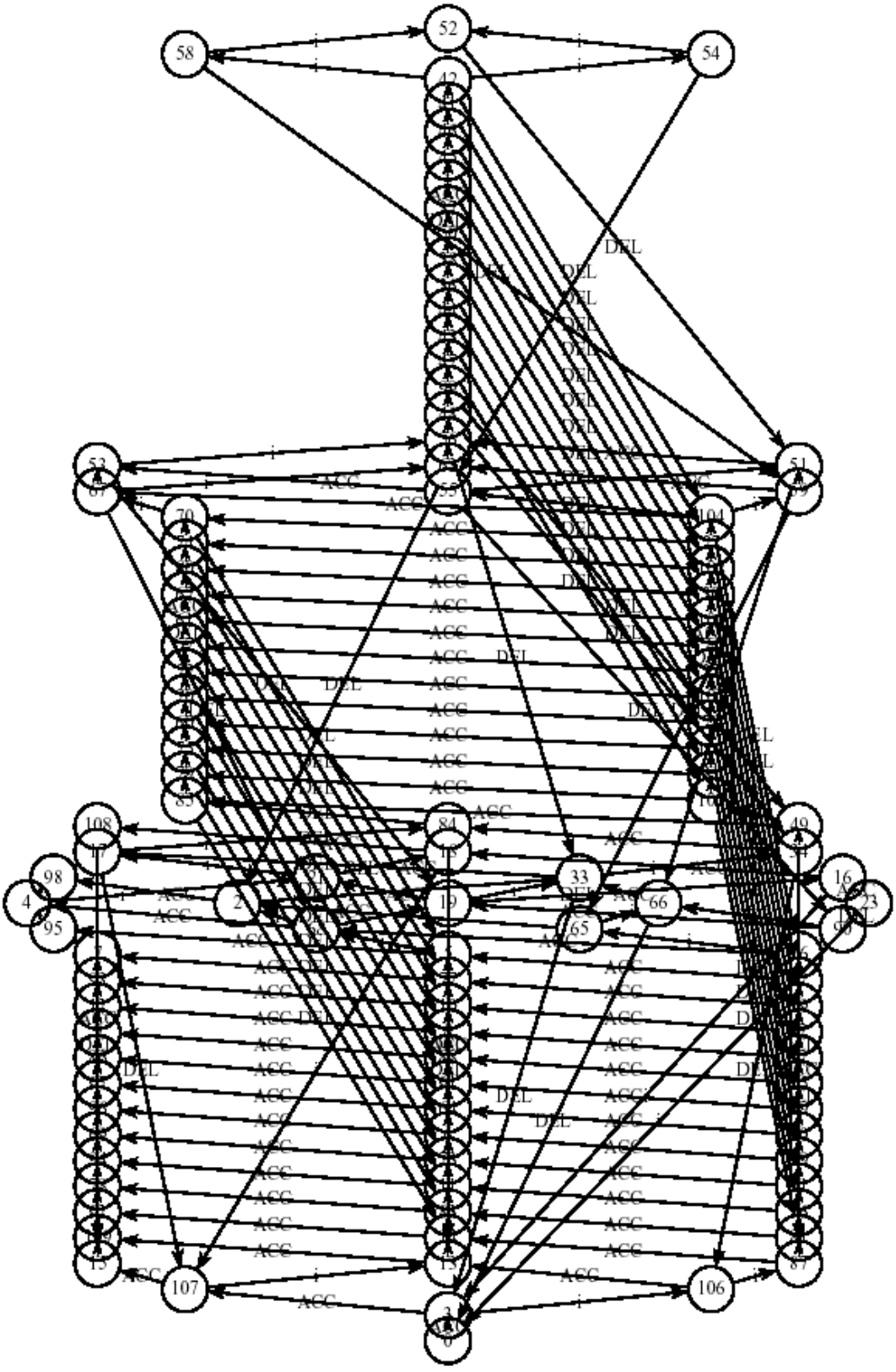

Figura 19. Gráfico de estados e transições reduzidos do protocolo SCTP no estabelecimento de três associações simultâneas

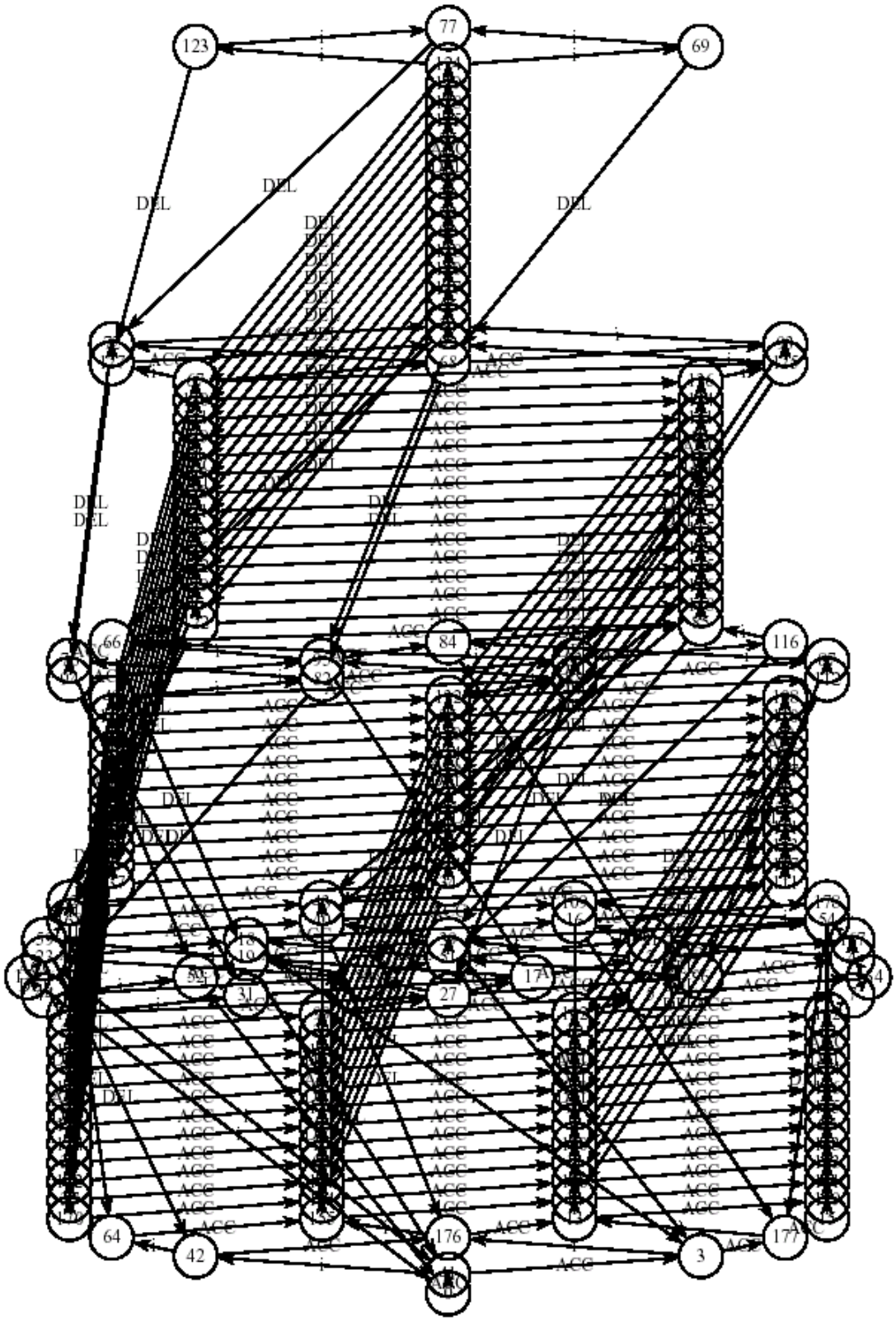

Figura 20. Gráfico de estados e transições reduzidos do protocolo SCTP no estabelecimento de quatro associações simultâneas

4.2.2. Comportamento Obtido Através da Interação entre Diversos SEGs

Este item tem como função apresentar a evolução dos estados e transições que se obteve com as simulações com o aumento do número de SEGs envolvidos nas associações. Como conclusão da simulação, poderemos observar a convergência do protocolo, pois todas as propriedades que demonstram esta convergência foram confirmadas, porém o número de estados e transições foi demasiadamente grande.

A partir das interações com três SEGs não é mais possível a visualização dos gráficos, apenas daqueles obtidos após o processo de redução de estados e transições. O número de estados alcança níveis em que o gráfico nada mais é que um borrão visual. Apesar do protocolo convergir, ele excede a capacidade de análise a que se propõe a metodologia aqui abordada. Esta característica de respostas, com excessiva quantidade de estados e transições, força a utilização da análise somente através dos resultados apresentados pela Tabela 9.

A ferramenta de simulação não conseguiu reduzir o grafo com um SEG interagindo com outros cinco; porém, como resposta, o aplicativo CADP informou que o protocolo estava livre de *deadlocks*. A capacidade de funcionamento do *software* é da ordem de dezenas de milhões de estados e transições, mas, para reduzir este grafo em particular, a ferramenta precisa gerar outros estados e transições para testes de redução de todo o grafo. Este mecanismo de funcionamento fez com que estas quantidades ultrapassassem a capacidade de análise da ferramenta.

4.3. Comentários sobre os Experimentos

O capítulo apresentou as simulações e os resultados alcançados neste estudo, ressaltando que os resultados obtidos foram conseguidos passo a passo e em várias simulações dentro de cada processo de validação.

O estudo do comportamento com o aumento dos participantes nas simulações, levou, a um número muito elevado de estados e transições, muitas vezes não representáveis graficamente, mesmo após a aplicação da propriedade de redução. Entretanto, como até este ponto o protocolo já estava bem estabelecido e consolidado, seus mecanismos de troca de mensagens já são bastante robustos para garantir que ele convirja, mesmo para uma quantidade de SEGs na rede bem maiores do que cinco.

É importante salientar que em todas as fases do projeto foram testadas a existência de *deadlocks*, se o protocolo era vivo e reiniciável e se respeitava as propriedades desejadas para o serviço. Como todas as simulações convergiram com este tipo de abordagem, cabe aqui salientar que a próxima fase de projeto, não abrangida por este trabalho, seria a inclusão da ação de temporizadores nas simulações.

Na maioria dos protocolos bem estruturados, toda mensagem enviada dispara um temporizador, que conta por um determinado tempo até ser reiniciado ou desativado pela chegada da resposta aguardada. Caso esta não chegue dentro do período de temporização, o ciclo de envio da mensagem repete-se por um certo número de vezes, determinado estatisticamente para cada situação. Não obtendo êxito no aguardo da resposta, o protocolo abandona aquela mensagem e envia para quem de direito alguma mensagem de erro.

Em LOTOS o temporizador é um conceito abstrato, e, na sua forma representativa generalizada, permite abstrair quaisquer valores para o tempo de resposta. Assim, a linguagem LOTOS testa todas as possibilidades de sucesso ou insucesso do protocolo, e, a cada envio de mensagem, é posta uma alternativa de temporização, que levará o protocolo a repetir novamente aquela mensagem com alternativas de sucesso ou não.

Capítulo 5

Conclusão

Este trabalho apresentou um processo de verificação formal para protocolos envolvidos na oferta de garantias de segurança à comunicação em sistemas móveis de 3ª Geração. Em particular, analisou-se o protocolo SCTP, atuando como o protocolo de transporte numa arquitetura de segurança baseada no IPSec.

Embora o SCTP seja ainda pouco conhecido no seio da comunidade de redes, já é considerado um fato consumado em telecomunicações e aparece em praticamente toda literatura moderna dessa área. Como mencionado anteriormente, o SCTP é item indispensável na integração entre as redes SS7 e aquelas baseadas em IP. Como essa integração anda a passos largos e as novas redes de sinalização telefônica tendem a ser parcial ou totalmente baseadas em IP, o SCTP surge como interessante alternativa ao transporte não só de sinalização, mas também de dados.

Ainda que o SCTP seja um protocolo novo, e contenha características modernas, absorveu as lições aprendidas nas últimas décadas com os demais protocolos de transporte, em particular os da arquitetura TCP/IP. Assim, o comitê SIGTRAN procurou tornar o SCTP adequado ao uso na *Internet* pública, sem perder de vista o seu objetivo inicial, que era o suporte ao transporte de sinalização telefônica.

O SCTP tem potencial para substituir, com muitas vantagens, o TCP. E, se alguma das extensões de confiabilidade parcial que vêm sendo propostas for aprovada, seja no protocolo oficial, seja através das implementações, também tem potencial para substituir o UDP em boa parte de suas aplicações atuais.

Na análise, simulação e validação do comportamento do protocolo SCTP, foi

empregada a linguagem LOTOS. Com base nessa linguagem, a ferramenta CADP possibilitou a correta validação da especificação do protocolo sem a necessidade da sua implementação. Apresentamos como especificar a fase de estabelecimento e de encerramento de associações SCTP, metodologia esta que pode ser empregada na análise de qualquer estrutura pertencente a uma camada de serviços de segurança da arquitetura de sistemas móveis de 3ª Geração.

Esta metodologia tem como base um processo de validação das propriedades de segurança de um protocolo. Esta validação envolve a verificação da quantidade dos seus estados e transições, bem como a confirmação das suas propriedades observacionais. Essa confirmação é realizada pela comparação entre o protocolo e o serviço respectivo, modelado também na linguagem LOTOS. Como foi mostrado, a equivalência observacional entre protocolo e serviço foi atendida. A equivalência forte não foi observada devido à simplificação empregada na definição do serviço, que resultou num diagrama com número de estados muito menor que o do protocolo, além de ter sua numeração alterada, de modo que a evolução interna de protocolo e serviço não é passível de comparação.

Os resultados obtidos demonstram que a descrição formal é de grande contribuição para a validação de segurança dos protocolos no sistema móvel de 3ª Geração e da evolução de seu comportamento com o aumento do número de entidades envolvidas no processo.

A afirmação de não haver *deadlocks* no caso em que se aumenta o número de entidades envolvidas é consistente, porque a metodologia de simulação empregada testou todas as características, possibilidades e peculiaridades de operação do protocolo, através de simulações anteriores com menos entidades, e estas sempre convergiram. Como cada SEG adicional inserido nessas simulações era exatamente igual aos descritos nas simulações anteriores, é possível concluir que a convergência é garantida em virtude das várias etapas concluídas com êxito para este tipo de simulação.

Este trabalho consistiu na aplicação de uma metodologia já estabelecida, a de especificação de protocolos empregando técnicas de descrição formal e a sua validação por ferramentas automatizadas, à análise de protocolos de segurança que ainda não foram testados no sistema móvel de 3ª Geração e, assim, verificando as garantias do atendimento aos requisitos de segurança que venham a ser estabelecidos.

Como possíveis trabalhos futuros, envolvendo o SCTP, podem-se mencionar:

- A realização de testes de avaliação de desempenho das implementações

existentes do SCTP, em comparação com as do TCP, e verificação desse desempenho em redes sujeitas a perdas de pacotes; e

- A proposição de mudanças nos algoritmos de controle de congestionamento, situação que não foi analisada neste trabalho, para a melhoria do desempenho em determinados ambientes, como a proposta de JUNGMAIER *et al.* (2001).

Referências Bibliográficas

3GPP TS 33.102 V5.2.0, 2003, *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture*, 3GPP Organizational Partners.

3GPP TS 33.120 V4.0.0, 2001, *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Principles and Objectives*, 3GPP Organizational Partners.

3GPP TS 33.210 V6.2.0, 2003, *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Network Domain Security; IP network layer security*, 3GPP Organizational Partners.

ALAMGIR, R., ATIQUZZAMAN, M., IVANCIC, W., 2002, "Effect of Congestion Control on the Performance of TCP and SCTP over Satellite Networks". In: *Proceedings of the NASA Earth Science Technology Conference*, Pasadena, Jun.

BAGATELLI, R., MOURA, D. F. C., PEDROZA, A. C. P., 2002, “Especificação Formal de uma Arquitetura de Suporte à Descoberta de Serviços em Redes Móveis Ad Hoc”. In: *Anais do V Workshop de Métodos Formais (WMF'2002)*, Gramado, Out.

BAGATELLI, R., 2003, *Projeto Formal de Protocolos de Acesso ao Meio e de Roteamento para Redes Ad-Hoc*. Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BOLOGNESI, T., BRINKSMA, E., 1987, "Introduction to the ISO Specification Language LOTOS", *Computer Networks and ISDN Systems*, v. 14, n. 1, pp. 25-29.

FERNANDEZ, J.C., GARAVEL, H., KERBRAT, A., et al., 1996, “CAESAR/ALDEBARAN Development Package: a protocol validation and verification toolbox”. In: *Proceedings of the Eighth Conference on Computer-Aided Verification (CAV)*, pp. 437-440, New Brunswick, Aug.

FU, S., ATIQUZZAMAN, M., 2002, “Effect of Delay Spike on SCTP, TCP Reno and Eifel in a Wireless Mobile Environment”. In: *Proceedings of the International Conference on Computer Communications and Networks*, Miami, Jul.

GERMEAU, F., LEDUC, G., 1997, “Model-based Design and Verification of Security Protocols using LOTOS”. In: *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols*, New Jersey, Sep.

HARKINS, D., CARREL, D., 1998, *The Internet Key Exchange (IKE)*, IETF RFC 2409.

HOARE, C.A.R., 1985, *Communicating Sequential Processes*. Englewood Cliffs, Prentice-Hall.

HU, F., SHARMA, N.K., 2002, "The Quantitative Analysis of TCP Congestion Control Algorithm in Third-Generation Cellular Networks Based on FSMC Loss Model and its Performance Enhancement". In: *Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, New York, Jun.

JUNGMAIER, A., RATHGEB, E.P., SCHOPP, M., et al., 2001, "SCTP – A Multi-link End-to-end Protocol for IP-based Networks", *International Journal of Electronics and Communications*, v. 55, n. 1 (Jan.), pp. 46-54.

JUNGMAIER, A., RESCORLA, E., TUEXEN, M., 2002, *Transport Layer Security over Stream Control Transmission Protocol*, IETF RFC 3436.

KARN, P., SIMPSON, W., 1999, *Photuris: Session-Key Management Protocol*, IETF RFC 2522.

KENT, S., ATKINSON, R., 1998, *Security architecture for the Internet protocol*, IETF RFC 2401.

KOH, S.J., JUNG, H.Y., 2003, "Mobile SCTP for IP Mobility Support in All-IP Networks". In: *Proceedings of CIC (Cellular and Intelligent Communications)*, Korea, Oct.

KRAWCZYK, H., BELLARE, M., CANETTI, R., 1997, *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104.

LIANG, Y.W., MOHAPI, S.J., HANRAHAN, H.E., 2002, "Evaluation of SCTP as a Transport Mechanism for CORBA GIOP Messages". In: *Proceedings of the South African Telecommunications Networks and Applications Conference*, Johannesburg, Sep.

LOGRIPPO, L., FACI, M., HAJ-HUSSEIN, M., 1992, "An introduction to LOTOS: learning by examples", *Computer Networks and ISDN Systems*, v. 23, pp. 325-342.

MARCO, G. de, VITO, D. de, LONGO, M., et al., 2003, "SCTP as a Transport for SIP: A Case Study". In: *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Jul.

MEER, J. de, ROTH, R. e VUONG, S., 1992 "Introduction to Algebraic Specifications Based on the Language ACT ONE", *Computer Networks and ISDN Systems*, v. 23, pp.

363-392.

MILNER, R., 1989, *Communication and Concurrency*. Englewood Cliffs, Prentice-Hall.

RAVIER, T., BRENNAN, R., CURRAN, T., 2001, "Experimental Studies of SCTP Multi-homing", *Proceedings of the First Joint IEI/IEE Symposium on Telecommunications Systems Research*, Dublin, Nov.

RIBEIRO, F.J.L., LOPES, J.C.R., PEDROZA, A.C.P., 2003, “Análise dos Processos de Segurança em Sistemas Móveis de 3ª Geração”. In: *Anais da I Escola Regional de Redes de Computadores*, pp. 59-64, Porto Alegre, Set.

RIBEIRO, F.J.L., LOPES, J.C.R., PEDROZA, A.C.P., 2003, “Análise do Estabelecimento de Associações Seguras em Sistemas Móveis de 3ª Geração”. In: *Proceedings of I2TS’2003 – 2nd International Information and Telecommunication Technologies Symposium*, Florianópolis, Nov.

STEWART, R., XIE, Q., MORNEAULT, K., et al., 2000, *Stream Control Transmission Protocol*, IETF RFC 2960.

STONE, J., STEWART, R., OTIS, D., 2002, *Stream Control Transmission Protocol (SCTP) Checksum Change*, IETF RFC 3309.

WANG, K., TRIPATHI, S.K., 1998, “Mobile-End Transport Protocol: An Alternative to TCP/IP Over Wireless Links”. In: *Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies - Infocom*, v. 3, n. NÚMERO (Mar), pp. 1046-1053.

ZHANG, Y., SINGH, B., 2000, “A Multi-Layer IPSec Protocol”. In: *Proceedings of the Ninth USENIX Security Symposium*, Denver, Aug.

Anexo A

Trechos (*chunks*) SCTP

1. INIT

Parâmetro	*Octetos*
Tipo = 0x01	1
Flags = 0x00	1
Comprimento do trecho	2
Etiqueta de inicialização	4
Tamanho da janela de recepção	4
Número desejado de fluxos de transmissão	2
Máximo aceitável de fluxos de recepção	2
TSN inicial (ISN)	4

2. INIT-ACK

Parâmetro	*Octetos*
Tipo = 0x02	1
Flags = 0x00	1
Comprimento do trecho	2
Etiqueta de inicialização	4
Tamanho da janela de recepção	4
Número desejado de fluxos de transmissão	2
Máximo aceitável de fluxos de recepção	2
TSN inicial (ISN)	4
Parâmetro tipo = 0x0007 (*cookie*)	2
Comprimento do parâmetro	2
Cookie	variável

3. COOKIE-ECHO

Parâmetro	*Octetos*
Tipo = 0x0A	1
Flags = 0x00	1
Tamanho do trecho	2
Cookie	variável

4. COOKIE_ACK

Parâmetro	*Octetos*
Tipo = 0x0B	1
Flags = 0x00	1
Comprimento do trecho = 0x04	2

5. SHUTDOWN

Parâmetro	*Octetos*
Tipo = 0x07	1
Flags = 0x00	1
Comprimento = 0x08	2
Confirmação do último TSN contínuo (sem lacunas) recebido	4

6. SHUTDOWN-ACK

Parâmetro	*Octetos*
Tipo = 0x08	1
Flags = 0x00	1
Tamanho = 0x04	2

7. SHUTDOWN-COMPLETE

Parâmetro	*Octetos*
Tipo = 0x0E	1
Flags = {T=0x01}	1
Tamanho = 0x04	2

8. ABORT

Parâmetro	*Octetos*
Tipo = 0x06	1
Flags: {T=0x01}	1
Comprimento do trecho	2
Causas do abortamento (opcional), como parâmetros opcionais	variável

9. DADOS

Parâmetro	*Octetos*
Tipo = 0x00	1
Flags = {U=0x04}|{B=0x02}|{E=0x01}	1
Comprimento do trecho	2
TSN	4
Número do fluxo	2
Número de seqüencia no fluxo (SSN)	2
Identificador de protocolo dos dados	4
Dados do usuário *(payload)*	variável

10. SACK

Campo	*Octetos*
Tipo = 0x03	1
Flags = 0x00	1
Comprimento do trecho	2
Confirmação do TSN contínuo (sem lacunas)	4
Buffer de recepção ainda livre	4
Número de lacunas = n_l	2
Número de duplicações = n_d	2
Lacuna 1: TSN relativo inicial	2
Lacuna 1: TSN relativo final	2
...	...
Lacuna n_l: TSN relativo inicial	2
Lacuna n_l: TSN relativo final	2
TSN duplicado 1	4
...	...
TSN duplicado n_d	4

11. ERROR

Parâmetro	*Octetos*
Tipo = 0x09	1
Flags = 0x00	1
Tamanho do trecho	2
Causas do erro, como parâmetros opcionais	variável

12. HEARTBEAT

Parâmetro	*Octetos*
Tipo = 0x04	1
Flags = 0x00	1
Comprimento do trecho	2
Tipo de parâmetro = 0x01	2
Comprimento do parâmetro	2
Identifi cador opaco	variável

13. HEARTBEAT-ACK

Parâmetro	*Octetos*
Tipo = 0x05	1
Flags = 0x00	1
Comprimento do trecho	2
Tipo de parâmetro = 0x01	2
Comprimento do parâmetro	2
Identifi cador contido em HEARTBEAT	variável

Anexo B

Especificações LOTOS

1. Especificação do Estabelecimento de uma Associação

```
specification SCTP_Associate [ACC, IP_DataReq1, IP_DataInd1,
IP_DataReq2, IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit


library

SCTP_LIB

endlib


behaviour

   hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2,
SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
   (
    (
     (((
     USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of data_type)
     |||
     USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of data_type)
     )
     |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
     (
     SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(INIT of pdu_type)
     |||
     SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
     )))
    )
```

```
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )


where


process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA:
data_type) : noexit :=

   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocReq !DATA; (* camada de aplicacao solicita associacao ao
SCTP *)
   SCTP_CommunUp ?DATA: data_type; (* SCTP notifica prontidao para
envio de mensagens *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (TAG) (* retorna ao
estado inicial *)

endproc


process USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (PDU: pdu_type) : exit :=

   SCTP_AssocReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (TAG, INIT)

   where

      process SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

         IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
         IP_DataReq1 !DATA !PDU;
         SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (COOKIE, COOKIE_ECHO)

         where
```

```
            process SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

               IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
               IP_DataReq1 !DATA !PDU;
               SCTP_CommunUp !DATA;
               SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (INIT)

            endproc

      endproc

endproc


process SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (PDU: pdu_type) : exit :=

   IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
   IP_DataReq2 !DATA !PDU;
   SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2](COOKIE, INIT_ACK)

   where

      process SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

         IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
         SCTP_AssocInd !DATA;
         SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (TAG, COOKIE_ACK)

         where

            process SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

               SCTP_AssocAck ?DATA:data_type;
               IP_DataReq2 !DATA !PDU;
               IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
               SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (INIT_ACK)

            endproc

      endproc

endproc


process MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2] :
noexit :=

   (IP_DataReq1 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd2 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )
```

```
   []

   (IP_DataReq2 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd1 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

endproc


endspec
```

2. Especificação do Encerramento de uma Associação

```
specification SCTP_Shutdown [ACC, IP_DataReq1, IP_DataInd1,
IP_DataReq2, IP_DataInd2, SCTP_ShutdReq, SCTP_ShutdCompl,
SCTP_ShutdInd, SCTP_ShutdAck, DEL] : noexit


library

SCTP_LIB

endlib


behaviour

  hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2,
SCTP_ShutdReq, SCTP_ShutdCompl, SCTP_ShutdInd, SCTP_ShutdAck in
   (
    (
     (((
     USER_1 [ACC, SCTP_ShutdReq, SCTP_ShutdCompl, DEL] ({} of
data_type)
     |||
     USER_2 [SCTP_ShutdInd, DEL, ACC, SCTP_ShutdAck] ({} of data_type)
     )
     |[SCTP_ShutdReq, SCTP_ShutdCompl, SCTP_ShutdInd, SCTP_ShutdAck]|
     (
     SCTP_A [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1, IP_DataInd1]
(SHUTDOWN of pdu_type)
     |||
     SCTP_B [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2, IP_DataInd2]
(SHUTDOWN_ACK of pdu_type)
     )))
    )
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )


where


process USER_1 [ACC, SCTP_ShutdReq, SCTP_ShutdCompl, DEL] (DATA:
data_type) : noexit :=

   ACC; (* camada de aplicacao recebe um pedido de encerramento da
associacao *)
   SCTP_ShutdReq !DATA; (* camada de aplicacao solicita encerramento
da associacao ao SCTP *)
   SCTP_ShutdCompl ?DATA: data_type; (* SCTP notifica encerramento da
associacao *)
   DEL; (* camada de aplicacao confirma encerramento da associacao *)
   USER_1 [ACC, SCTP_ShutdReq, SCTP_ShutdCompl, DEL] (TAG) (* retorna
ao estado inicial *)

endproc
```

```
process USER_2 [SCTP_ShutdInd, DEL, ACC, SCTP_ShutdAck] (DATA:
data_type) : noexit :=

   SCTP_ShutdInd ?DATA: data_type; (* SCTP notifica encerramento de
associacao em curso *)
   DEL; (* camada de aplicacao confirma encerramento da associacao *)
   ACC; (* camada de aplicacao recebe um pedido de encerramento da
associacao *)
   SCTP_ShutdAck !DATA; (* camada de aplicacao confirma a solicitacao
de encerramento da associacao *)
   USER_2 [SCTP_ShutdInd, DEL, ACC, SCTP_ShutdAck] (TAG) (* retorna ao
estado inicial *)

endproc


process SCTP_A [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1,
IP_DataInd1] (PDU: pdu_type) : exit :=

   SCTP_ShutdReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendShut [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1,
IP_DataInd1] (TAG, SHUTDOWN)

   where

      process SCTP_SendShut [SCTP_ShutdReq, SCTP_ShutdCompl,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

         IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
         IP_DataReq1 !DATA !PDU;
         SCTP_SendShutCompl [SCTP_ShutdReq, SCTP_ShutdCompl,
IP_DataReq1, IP_DataInd1] (TAG, SHUTDOWN_COMPLETE)

         where

            process SCTP_SendShutCompl [SCTP_ShutdReq,
SCTP_ShutdCompl, IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU:
pdu_type) : exit :=

               IP_DataInd1 ?DATA:data_type ?PDU: pdu_type;
               IP_DataReq1 !DATA !PDU;
               SCTP_ShutdCompl !DATA;
               SCTP_A [SCTP_ShutdReq, SCTP_ShutdCompl, IP_DataReq1,
IP_DataInd1] (SHUTDOWN)

            endproc

      endproc

endproc


process SCTP_B [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2,
IP_DataInd2] (PDU: pdu_type) : exit :=

   IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
   IP_DataReq2 !DATA !PDU;
```

```
   SCTP_SendShutAck [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2,
IP_DataInd2](TAG, SHUTDOWN_ACK)

   where

      process SCTP_SendShutAck [SCTP_ShutdInd, SCTP_ShutdAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

         IP_DataInd2 ?DATA:data_type ?PDU: pdu_type;
         SCTP_ShutdInd !DATA;
         SCTP_ShutdAck ?DATA:data_type;
         IP_DataReq2 !DATA !PDU;
         SCTP_B [SCTP_ShutdInd, SCTP_ShutdAck, IP_DataReq2,
IP_DataInd2] (SHUTDOWN_ACK)

      endproc

endproc


process MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2] :
noexit :=

   (IP_DataReq1 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd2 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

   []

   (IP_DataReq2 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd1 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

 endproc


endspec
```

3. Especificação do Serviço de Estabelecimento de uma Associação

```
specification SCTP_AssociateService [ACC, DEL] : noexit

library

SCTP_LIB

endlib

behaviour

     SCTP_AssocServ [ACC, DEL]

where

process SCTP_AssocServ [ACC, DEL] : noexit :=

   ACC;
   DEL;
   SCTP_AssocServ [ACC, DEL]

endproc

endspec
```

4. Especificação do Estabelecimento de Associações com mais de um SEG

4.1. Três SEGs, em associação 1 x 2 (um cliente e dois servidores):

```
specification SCTP_Associate3Seg [ACC, IP_DataReq1, IP_DataInd1,
IP_DataReq2, IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit


library

SCTP_LIB

endlib


behaviour

   hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2,
SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
   (
    (
     (
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
     )
     |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
     (
      SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(INIT of pdu_type)
      |||
      SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
     )
    )
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )
```

```
where


process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA:
data_type) : noexit :=

   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocReq !DATA; (* camada de aplicacao solicita associacao ao
SCTP *)
   SCTP_CommunUp ?DATA: data_type; (* SCTP notifica prontidao para
envio de mensagens *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (TAG) (* retorna ao
estado inicial *)

endproc


process USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (PDU: pdu_type) : exit :=

   SCTP_AssocReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (TAG, INIT)

   where
```

```
      process SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

         IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
         IP_DataReq1 !DATA !PDU;
         SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (COOKIE, COOKIE_ECHO)

         where

            process SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

               IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
               IP_DataReq1 !DATA !PDU;
               SCTP_CommunUp !DATA;
               SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (INIT)

            endproc

      endproc

endproc


process SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (PDU: pdu_type) : exit :=

   IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
   IP_DataReq2 !DATA !PDU;
   SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2](COOKIE, INIT_ACK)

   where

      process SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

         IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
         SCTP_AssocInd !DATA;
         SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (TAG, COOKIE_ACK)

         where

            process SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

               SCTP_AssocAck ?DATA:data_type;
               IP_DataReq2 !DATA !PDU;
               IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
               SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (INIT_ACK)

            endproc

      endproc
```

```
endproc


process MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2] :
noexit :=

   (IP_DataReq1 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd2 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

   []

   (IP_DataReq2 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd1 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

endproc


endspec
```

4.2. Quatro SEGs, em associação 1 x 3 (um cliente e três servidores):

```
specification SCTP_Associate4Seg [ACC, IP_DataReq1, IP_DataInd1,
IP_DataReq2, IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit


library

SCTP_LIB

endlib


behaviour

   hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2,
SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
   (
    (
     (
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
```

```
       USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
     )
     |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
     (
      SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(INIT of pdu_type)
      |||
      SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
     )
    )
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )


where


process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA:
data_type) : noexit :=

   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocReq !DATA; (* camada de aplicacao solicita associacao ao
SCTP *)
   SCTP_CommunUp ?DATA: data_type; (* SCTP notifica prontidao para
envio de mensagens *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (TAG) (* retorna ao
estado inicial *)

endproc


process USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc
```

```
process USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (PDU: pdu_type) : exit :=

   SCTP_AssocReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (TAG, INIT)

   where

      process SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

         IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
         IP_DataReq1 !DATA !PDU;
         SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (COOKIE, COOKIE_ECHO)

         where

            process SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

               IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
               IP_DataReq1 !DATA !PDU;
               SCTP_CommunUp !DATA;
               SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
```

```
IP_DataInd1] (INIT)

            endproc

      endproc

endproc


process SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (PDU: pdu_type) : exit :=

   IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
   IP_DataReq2 !DATA !PDU;
   SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2](COOKIE, INIT_ACK)

   where

      process SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

         IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
         SCTP_AssocInd !DATA;
         SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (TAG, COOKIE_ACK)

         where

            process SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

               SCTP_AssocAck ?DATA:data_type;
               IP_DataReq2 !DATA !PDU;
               IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
               SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (INIT_ACK)

            endproc

      endproc

endproc


process MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2] :
noexit :=

   (IP_DataReq1 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd2 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

   []

   (IP_DataReq2 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd1 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
```

```
   )

endproc


endspec
```

4.3. Cinco SEGs, em associação 1 x 4 (um cliente e quatro servidores):

```
specification SCTP_Associate5Seg [ACC, IP_DataReq1, IP_DataInd1,
IP_DataReq2, IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit


library

SCTP_LIB

endlib


behaviour

   hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2,
SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
   (
    (
     (
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
```

```
      )
     )
     |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
     (
      SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(INIT of pdu_type)
      |||
      SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
     )
    )
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )


where


process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA:
data_type) : noexit :=

   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocReq !DATA; (* camada de aplicacao solicita associacao ao
SCTP *)
   SCTP_CommunUp ?DATA: data_type; (* SCTP notifica prontidao para
envio de mensagens *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (TAG) (* retorna ao
estado inicial *)

endproc


process USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
```

```
de associacao *)
   USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (PDU: pdu_type) : exit :=

   SCTP_AssocReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (TAG, INIT)

   where

      process SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

         IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
         IP_DataReq1 !DATA !PDU;
         SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (COOKIE, COOKIE_ECHO)

         where

            process SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
```

```
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

               IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
               IP_DataReq1 !DATA !PDU;
               SCTP_CommunUp !DATA;
               SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (INIT)

            endproc

      endproc

endproc


process SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (PDU: pdu_type) : exit :=

   IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
   IP_DataReq2 !DATA !PDU;
   SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2](COOKIE, INIT_ACK)

   where

      process SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

         IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
         SCTP_AssocInd !DATA;
         SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (TAG, COOKIE_ACK)

         where

            process SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

               SCTP_AssocAck ?DATA:data_type;
               IP_DataReq2 !DATA !PDU;
               IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
               SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (INIT_ACK)

            endproc

      endproc

endproc


process MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2] :
noexit :=

   (IP_DataReq1 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd2 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )
```

```
   []

   (IP_DataReq2 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd1 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

endproc


endspec
```

4.4. Seis SEGs, em associação 1 x 5 (um cliente e cinco servidores):

```
specification SCTP_Associate6Seg [ACC, IP_DataReq1, IP_DataInd1,
IP_DataReq2, IP_DataInd2, SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd,
SCTP_AssocAck, DEL] : noexit


library

SCTP_LIB

endlib


behaviour

   hide IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2,
SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck in
   (
    (
     (
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
```

```
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
      |||
      (
       USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] ({} of
data_type)
       |||
       USER_6 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] ({} of
data_type)
      )
     )
     |[SCTP_AssocReq, SCTP_CommunUp, SCTP_AssocInd, SCTP_AssocAck]|
     (
      SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1, IP_DataInd1]
(INIT of pdu_type)
      |||
      SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2, IP_DataInd2]
(INIT_ACK of pdu_type)
     )
    )
    |[IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]|
      MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
   )


where


process USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (DATA:
data_type) : noexit :=

   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocReq !DATA; (* camada de aplicacao solicita associacao ao
SCTP *)
   SCTP_CommunUp ?DATA: data_type; (* SCTP notifica prontidao para
envio de mensagens *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   USER_1 [ACC, SCTP_AssocReq, SCTP_CommunUp, DEL] (TAG) (* retorna ao
estado inicial *)

endproc


process USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_2 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
```

```
ao estado inicial *)

endproc


process USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_3 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_4 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_5 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process USER_6 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (DATA:
data_type) : noexit :=

   SCTP_AssocInd ?DATA: data_type; (* SCTP notifica associacao em
curso *)
   DEL; (* camada de aplicacao confirma prontidao para envio de
mensagens *)
```

```
   ACC; (* camada de aplicacao recebe um pedido de associacao *)
   SCTP_AssocAck !DATA; (* camada de aplicacao confirma a solicitacao
de associacao *)
   USER_6 [SCTP_AssocInd, DEL, ACC, SCTP_AssocAck] (COOKIE) (* retorna
ao estado inicial *)

endproc


process SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (PDU: pdu_type) : exit :=

   SCTP_AssocReq ?DATA: data_type;
   IP_DataReq1 !DATA !PDU;
   SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (TAG, INIT)

   where

      process SCTP_SendInit [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

         IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
         IP_DataReq1 !DATA !PDU;
         SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (COOKIE, COOKIE_ECHO)

         where

            process SCTP_SendCookieEcho [SCTP_AssocReq, SCTP_CommunUp,
IP_DataReq1, IP_DataInd1] (DATA: data_type, PDU: pdu_type) : exit :=

               IP_DataInd1 ?DATA: data_type ?PDU: pdu_type;
               IP_DataReq1 !DATA !PDU;
               SCTP_CommunUp !DATA;
               SCTP_A [SCTP_AssocReq, SCTP_CommunUp, IP_DataReq1,
IP_DataInd1] (INIT)

            endproc

      endproc

endproc


process SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (PDU: pdu_type) : exit :=

   IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
   IP_DataReq2 !DATA !PDU;
   SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2](COOKIE, INIT_ACK)

   where

      process SCTP_SendInitAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

         IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
         SCTP_AssocInd !DATA;
```

```
         SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (TAG, COOKIE_ACK)

         where

            process SCTP_SendCookieAck [SCTP_AssocInd, SCTP_AssocAck,
IP_DataReq2, IP_DataInd2] (DATA: data_type, PDU:pdu_type) : exit :=

               SCTP_AssocAck ?DATA:data_type;
               IP_DataReq2 !DATA !PDU;
               IP_DataInd2 ?DATA: data_type ?PDU: pdu_type;
               SCTP_B [SCTP_AssocInd, SCTP_AssocAck, IP_DataReq2,
IP_DataInd2] (INIT_ACK)

            endproc

      endproc

endproc


process MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2] :
noexit :=

   (IP_DataReq1 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd2 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

   []

   (IP_DataReq2 ?Data: data_type ?PDU: pdu_type;
    ((IP_DataInd1 !Data !PDU;
     MEDIUM [IP_DataReq1, IP_DataInd1, IP_DataReq2, IP_DataInd2]
    ))
   )

endproc


endspec
```

5. Especificação da Biblioteca SCTP

```
type PDU is

   sorts PDU_TYPE  (*! implementedby PDUTYPE comparedby CMP_PDUTYPE
printedby PRINT_PDUTYPE *)

   opns
      INIT (*! implementedby INIT constructor *),
      INIT_ACK (*! implementedby INIT_ACK constructor *),
      COOKIE_ECHO (*! implementedby COOKIE_ECHO constructor *),
      COOKIE_ACK (*! implementedby COOKIE_ACK constructor *),
      SHUTDOWN (*! implementedby SHUTDOWN constructor *),
      SHUTDOWN_ACK (*! implementedby SHUTDOWN_ACK constructor *),
      SHUTDOWN_COMPLETE (*! implementedby SHUTDOWN_COMPLETE
constructor *): -> PDU_TYPE

endtype



type DATA is

   sorts DATA_TYPE  (*! implementedby DATATYPE comparedby CMP_DATATYPE
printedby PRINT_DATATYPE *)

   opns
      TAG (*! implementedby TAG constructor *),
      COOKIE (*! implementedby COOKIE constructor *),
      {} (*! implementedby DDD constructor *): -> DATA_TYPE

endtype
```