PROJETO FORMAL DE PROTOCOLOS DE ACESSO AO MEIO E DE ROTEAMENTO PARA REDES AD-HOC

Renato Bagatelli

DISSERTAÇÃO 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. José Ferreira de Rezende, Dr.

____

Prof. Sérgio Luiz Cardoso Salomão, D. Sc.

____

Prof. Julius Cesar Barreto Leite PhD.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2003

BAGATELLI, RENATO

Projeto Formal de Protocolos de Acesso ao Meio e de Roteamento para Redes Ad-Hoc [Rio de Janeiro] 2003

XIV, 135 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2003)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Protocolos de Acesso ao Meio e Roteamento
2. Métodos Formais
3. Redes Ad-Hoc

I. COPPE/UFRJ II. Título (série)

À minha família, pelo continuo incentivo.

Agradecimentos

A Deus.

Aos meus pais que me ensinaram a importância do conhecimento e do trabalho.

Ao meu orientador Professor Aloysio de Castro Pinto Pedroza, pelo estímulo, incentivo, parceria e confiança para a realização deste trabalho.

Ao Instituto de Pesquisa e Desenvolvimento pela oportunidade.

Aos meus chefes e colegas de trabalho pela compreensão e colaboração.

Ao meu amigo David que foi meu parceiro e colaborador durante todo o curso.

Ao meu amigo Daniel, que me incentivou e colaborou para o meu ingresso no mestrado.

A todos os meus professores que sem a menor sombra de dúvida muito contribuíram para a conclusão deste trabalho.

Aos meus amigos da COPPE com os quais compartilhei momentos alegres e difíceis em busca de nossos objetivos. São muitos, mas vou tentar me lembrar deles: Saulo, Kleber, Viviane, Alexandre, Jaime, Fabrício, Gil, Avalle, Valentim, Roman, Gardel, Fagundes, Rubi, Eric, Pedro, Doc, Bernardo, Márcio, Marcial, Granato, Belém, Aline, Paulo, Artur, Luís, Baiano, André, Roberta, Roberta Guinther, Ivana, LG, Sidney, Ingrid, Glauco, Isaac, Beto e Guto.

Aos professores que participaram da Comissão Examinadora.

A todos os funcionários do Departamento que de uma forma ou de outra contribuíram para o êxito deste trabalho.

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

PROJETO FORMAL DE PROTOCOLOS DE ACESSO AO MEIO E DE ROTEAMENTO PARA REDES AD-HOC

Renato Bagatelli

Abril/2003

Orientador: Aloysio de Castro Pinto Pedroza

Programa: Engenharia Elétrica

Este trabalho apresenta um estudo dos protocolos mais utilizados para roteamento e acesso ao meio em redes *ad-hoc*, fazendo uma análise das características essenciais para o funcionamento mais eficiente dentro da arquitetura sugerida. Os protocolos DSR e LANMAR foram utilizados para roteamento e o MACAW para acesso ao meio.

Foi utilizada a técnica de métodos formais, com o uso da linguagem LOTOS e uma ferramenta apropriada, para o teste, a simulação, a verificação e a validação dos protocolos. Além disso, foram verificadas propriedades essenciais para garantia de funcionamento com correção, proporcionando interligação funcional dos protocolos dentro da arquitetura proposta.

Para atingir estes objetivos foi criada uma metodologia para desenvolvimento de forma incremental dos experimentos com os protocolos, onde a cada passo foi incrementada a complexidade da rede e das estações desta, até que o protocolo sob teste se tornasse o mais genérico possível, simulando o funcionamento que mais se aproximasse do real para uma rede *ad-hoc*.

A partir dos experimentos foi possível obter algumas conclusões que levaram a propor modificações nos protocolos de interesse.

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

FORMAL DESIGN OF MEDIUM ACCESS CONTROL AND ROUTING PROTOCOLS FOR WIRELESS AD HOC NETWORKS

Renato Bagatelli

April/2003

Advisor: Aloysio de Castro Pinto Pedroza

Department: Electrical Engineering

This work presents a theoretical study of the most employed medium access control (MAC) and routing protocols in wireless ad hoc networks, detaching and analyzing their essential features for a more efficient deployment on the suggested framework. DSR and LANMAR were chosen to perform routing tasks, as well as MACAW for MAC purposes.

Formal description techniques (FDT) were used, by the employment of LOTOS formalism to design specifications over a suitable development package, performing testing, simulation, verification and validation tasks. Essential properties were also verified, in order to guarantee not only an error-free protocol usage, but also a correct layer interaction as idealized by the proposed framework.

As a requirement to fulfill the aim of this work, an incremental development methodology was created for usage over a series of experiments with the chosen protocols. This methodology allowed an increase of complexity on stations and networks under a step-by-step approach up to the description of protocol specifications as general as possible. By the end, final experiments described a network deployment close to that expected for real scenarios.

Some conclusions were drawn, allowing proposals of modifications on the specified framework protocols.

Sumário

Resumo v

Abstract iv

Lista de Acrônimos x

Lista de Figuras xii

Lista de Tabelas xiv

1. Introdução 01

1.1. Rede *Ad-hoc*..01

1.2. Aspectos de Roteamento em Redes *Ad-hoc*................................02

1.3. Protocolos de Acesso ao Meio...03

1.4. Projeto...03

2. Arquitetura para Redes *Ad-hoc* 05

2.1. Introdução...05

2.2. Modelo das Estações e da Rede..05

2.2.1. Arquitetura da Rede Proposta...07

2.3. Descrição dos Protocolos de Roteamento em Redes *Ad-hoc*......09

2.4. Descrição dos Protocolos de Acesso ao Meio, Protocolos de Suporte aos Protocolos de Roteamento....................................18

2.5. Comentários..20

3. Projeto 22

3.1 Introdução..22

3.2 Metodologia...23

3.3 Especificação e Verificação...24

3.3.1. Linguagem de Descrição Formal LOTOS............................25

3.3.1.1. Bibliotecas em LOTOS.. 27

3.3.2. CADP (Pacote de Desenvolvimento Caesar/Aldebaran)......27

3.4. Especificação e Verificação Formal dos Protocolos....................30
3.4.1. Protocolo MACAW..30
3.4.2 Descrição do Serviço MACAW..31
3.4.3. Consolidação dos Resultados das Simulações para
o Protocolo MACAW de uma Forma Geral.........................32
3.4.4. Modificações no Protocolo de Acesso ao Meio MACAW.....32
3.5. Especificação Formal e Verificação do Protocolo DSR................35
3.5.1. Protocolo DSR..35
3.5.2 Descrição do Serviço DSR..36
3.5.3. Consolidação dos Resultados das Simulações para
o Protocolo DSR de uma Forma Geral...............................36
3.5.4. Modificações no Protocolo de Roteamento DSR.................37
3.6. Especificação e Verificação do Protocolo LANMAR.....................38
3.6.1. Protocolo LANMAR...38
3.6.2. Descrição do Serviço LANMAR...40
3.6.3. Consolidação dos Resultados das Simulações para
o Protocolo LANMAR de uma Forma Geral........................41
3.6.4. Modificações no Protocolo de Roteamento LANMAR..........41
3.7. Comentários...42
4. Resultados 44
4.1. Introdução...44
4.2. O Protocolo de Acesso ao Meio MACAW....................................44
4.2.1. Resultado das Simulações do Protocolo MACAW..............47
4.2.1.1. Simulação com o Protocolo MACAW Original........47
4.2.1.2. Simulações Intermediárias....................................51
4.2.1.3 Duas Estações Transmissoras e Receptoras
Simultaneamente com Processo de Sincronização..53
4.2.2. Resumo das Etapas dos Experimentos...............................57
4.2.3. Conclusões e Modificações sobre o Experimento do
Protocolo MACAW..60
4.3. O Protocolo de Roteamento DSR...61
4.3.1. Simulações do Protocolo Roteamento DSR........................61
4.3.1.1. Simulação com Duas Estações.............................61
4.3.1.2. Simulações Intermediárias....................................63

4.3.1.3. Simulação com Passagem de Parâmetros para Três Estações....69
4.3.1.4. Simulação de Estações Completas, com Utilização de Passagem de Parâmetros e Temporizadores em cada Evento....75
4.3.2. Consolidações dos Resultados para o Protocolo DSR....77
4.3.3. Resumo dos Experimentos do Protocolo DSR....84
4.3.4. Conclusões dos Experimentos do Protocolo DSR....86
4.4. O Protocolo de Roteamento LANMAR....88
4.4.1. Simulações do Protocolo de Roteamento LANMAR....90
4.4.1.1. Simulação com Duas Estações....91
4.4.1.2. Simulações Intermediárias....92
4.4.1.3. Estações Líderes se Comunicando com qualquer Estação da Rede....98
4.4.2. Resumo dos Experimentos e Análise dos Resultados Obtidos....101
4.4.3. Conclusões sobre os Experimentos com o LANMAR....105
4.5. Comentários sobre os Experimentos....106
5. Conclusão **107**
Referências Bibliográficas **111**
Apêndice A - Especificação Formal em LOTOS do Protocolo DSR....**118**
Apêndice B - Biblioteca em LOTOS dos Parâmetros do Protocolo DSR....**134**

Lista de Acrônimos

AbACK: *Acknowledgment Broadcasting;*

ABROAD: *Adaptive Meium Access Control (MAC) Protocol for Reliable in Broadcast;*

ADAPT: A Dynamically Self-Adjusting Media Acess Control Protocol;

ACK: *Acknowledgment;*

AODV: *Ad-hoc On-Demand Distance Vector Routing;*

BCG: *Gráfico de Código Binário;*

BRP: *Bordercast Routing Protocol;*

CADP: *Pacote de Desenvolvimento Caesar/Aldebaran;*

CATA: *Channel Access Control in Ad-Hoc Networks*

CbCTS: *Clear Broadcasting Request to Clear;*

CCS: *Calculus Comunnication System;*

CHMA: Channel-Hopping Multiple Access;

CSP: *Calculus Sequencial Processes;*

CTS: *Clear to Send;*

DbDS: *Data Send Broadcasting;*

DS: *Data Send;*

DSDV: *Destination-Sequenced Distance-Vector;*

DSR: *Dynamic Source Routing Protocol;*

ELUDO: Environnement LOTOS de l'Université d'Ottawa;

ERP: *IntErzone Routing Protocol;*

FAMA: *Floor Acquisition Multiple Access;*

FDT: *Formal description techniques*;

FSR: *Fisheye State Routing;*

HRMA: *Hop Reservation Multiple Access;*

IARP: *IntrAzone Routing Protocol;*

IEC: *International Electrotechnical Commission;*

IETF: *Internet Engineering Tesk Force;*

INRIA: Institut National de Recherche em Informatique et em Automatique;

ISO: *International Organization for Standardization;*

LANMAR: *Landmark Routing Protocol for Large Scale Ad-Hoc Networks;*

LOTOS: *Language of Temporal Ordering Specification;*

LTS: *Labelled Transition Systems;*

MAC: *Media Access Protocol;*

MACA: *Multiple Access with Collision Avoidance;*

MACA-BI: *MACA (by invitation);*

MACA-CT: *Common-Transmitter-Based Multiple Access with Collision Avoidance;*

MACAW: *Media Access Protocol for Wireless LAN's;*

MANET: *Mobile Ad-Hoc Networks;*

MPR: *Multipoint Relays;*

ns: *Network Simulator;*

OLSR: *Optimized Link State Routing Protocol;*

OSI: *Open System Interconnection;*

PDU: *Service Data Unit;*

QoS: *Quality of Service;*

RbRTS: *Request Broadcasting Request to Send;*

RTS: *Request to Send;*

SDU: *Protocol Data Unit;*

TBRPF: *Topology Broadcast Based on Reverse-Path Forwarding;*

Tcl: *Tool Command Language;*

TGV: *Test Generation with Verification Technology;*

VASY: *Validation of Systems;*

VERIMAG: *Vérification et temps réel;*

XTL: *Linguagem Temporal eXecutável;*

ZRP: *The Zone Routing Protocol (ZRP) for Ad-Hoc Networks.*

Lista de Figuras

Figura 01 - Modelo da Rede...06
Figura 02 - Modelo da Rede em Células...07
Figura 03 - Estruturação dos Protocolos em Camadas.........................09
Figura 04 - Rede de Petri do Protocolo Macaw Original........................46
Figura 05 - Diagrama de Processos da Versão Original do Protocolo MACAW...49
Figura 06 - Gráfico de Estados e Transições do Protocolo MACAW Original...50
Figura 07 - Diagrama de Processos da Quarta Versão do Protocolo, Incluindo Retransmissão entre Estações............................52
Figura 08 - Estruturas de Decisão e de Trocas de Mensagens Dentro de Cada Estação..55
Figura 09 - Gráfico de Estados e transições do Protocolo MACAW, Modificado...56
Figura 10 - Primeira Implementação com Dois Usuários e Dois Processos Simples..62
Figura 11 - Protocolo DSR sem Passagem de Parâmetros com Estruturas de Reencaminhamento de Rotas.......................64
Figura 12 - Simulação de Cinco Estações através de Processos Simples..66
Figura 13 - Protocolo DSR com Passagem de Parâmetros entre Três Estações...71
Figura 14 - Protocolo DSR com Passagem de Parâmetros para Três Estações...72
Figura 15 - DSR com Passagem de Parâmetros e temporizadores......74
Figura 16 - Esquema Simplificado da Estrutura Interna de Decisão e Troca de Mensagens de cada Estação Autônoma...........76
Figura 17 - Arquitetura do Protocolo LANMAR.....................................90
Figura 18 - Representação dos Processos de uma Célula com Uma

Estação Líder e Uma Subordinada.....................................94
Figura 19 - Representação dos Processos de uma Estação Subordinada..95
Figura 20 - Estados e Transições Convergentes para Duas Células e Quatro Estações com Processo de Sincronismo Inserido...97
Figura 21 - Representação da Rede com Duas Células e Quatro Estações..99
Figura 22 - Estados e Transições Convergentes para Duas Células, Onde as Estações Líderes Podem se Comunicar com as Demais Estações da Rede..100

Lista de Tabelas

Tabela 01 - Resumo das Funções Eventos mais Utilizados na Linguagem LOTOS..26
Tabela 02 - Tipos de Iterações..27
Tabela 03 - Resumo das Simulações do Protocolo MACAW..................59
Tabela 04 - Resultados Obtidos nos Experimentos do Protocolo DSR....85
Tabela 05 - Evolução do Experimento com o Protocolo LANMAR.........102

1. Introdução

1.1. Redes *Ad-hoc*

Uma rede *ad-hoc* pode ser definida [1] como um sistema autônomo de plataformas móveis. Tais redes operam sem infraestrutura fixa, sem estações rádio base. Os nós são eles próprios, cada um deles, roteadores e/ou controladores da rede.

Na mobilidade em redes *ad-hoc*, as estações, dispositivos, devem agir cooperativamente para roteamento do tráfego e se adaptar a rede para estados altamente dinâmicos dos enlaces e destes dispositivos [2], com isto a comunicação entre estações distantes de uma rede *ad-hoc* requer roteamento com múltiplos saltos.

A principal dificuldade em redes sem fio é a característica das estações se movimentarem independentemente umas das outras [3]. Esta mobilidade causa diversos tipos de falhas nos enlaces, além de alguns enlaces serem ativados desnecessariamente, congestionando a rede.

A efetividade do protocolo de roteamento adaptativo depende da oportunidade e detalhes das informações da topologia da rede disponível para ele no momento da busca da rota [4], ou do encaminhamento do pacote. Um grande fluxo de mensagens pode facilmente saturar a rede, aonde informações chegam tarde por causa da latência do meio. Minimizar informações é crucial para a eficiência de operação de uma rede.

As redes móveis sem fio de múltiplos saltos são a maneira mais prática para provimento em baixo custo e rápido desdobramento no terreno, com auto-organização da rede, necessário para comunicação em locais onde a infraestrutura não existe [1,5], ou não é possível ou, é onerosa ou ainda, foi destruída por algum problema ou, por algum acidente natural.

O uso deste tipo de rede exclui planejamento estratégico em campos de batalha [6], em áreas de emergência, em equipes de resgate e para locais de remota localização [5]. Muitas destas áreas possuem aplicações em alta demanda, tendo pouca tolerância a retardos, *jitter*, perdas e corrupção das informações. Assim, a habilidade da rede para prover determinada qualidade de serviço [7], QoS, depende das propriedades da rede [8] no que se refere aos itens acima, além da carga de tráfego da rede e do algoritmo de controle e roteamento da mesma.

A rede, o protocolo, deve tentar minimizar o tamanho, a duração e a troca de mensagens para controle e manutenção dos canais de comunicação. O protocolo deve considerar o consumo de potência, a segurança, a autenticação da informação, a probabilidade limitada de detecção da informação que são pontos chaves de interesse em redes sem fio *ad-hoc* de múltiplos saltos.

1.2. Aspectos de Roteamento em Redes *Ad-hoc*

Numa rede *ad-hoc* com centenas de estações, duas estações podem estar muito próximas, serem vizinhas uma da outra num instante e no momento seguinte podem estar em bordas opostas, nos extremos opostos da rede. Estas estações podem estar, por exemplo, uma dentro de um veículo e a outra com uma pessoa caminhando, ambos trafegando com velocidades distintas e variáveis dentro da rede, podendo estar em qualquer lugar a qualquer momento [3].

A obtenção do endereço de uma estação por uma outra, por onde e através de quais estações este consegue transmitir dados para um determinado destino, é uma das questões chaves do bom funcionamento das redes *ad-hoc*. Dentro deste contexto foram estudados vários protocolos de roteamento, todos os mencionados no *site* do IETF e diversas literaturas voltadas para esta área entre 20/09/01 até 30/12/02.

Os protocolos reativos, isto é, que fazem a busca de rotas, se já não as possuírem, no momento da necessidade do envio de dados, como o DSR e o AODV, são mais elaborados e mais eficientes, para redes dinâmicas e com alta mobilidade [9, 10].

Protocolos que mantém uma tabela de endereços das estações da rede através de trocas de mensagens periódicas, ou seja, de características pró-ativas, são mais eficientes para redes com menor mobilidade.

Num outro método de roteamento, protocolos como o LANMAR, procuram ser mais eficientes dividindo a rede em células, utilizando protocolos pró-ativos com características diferentes dentro e fora destas, para melhorar sua performance em redes maiores e mais móveis.

1.3. Protocolos de Acesso ao Meio

Para assegurar um bom desempenho da camada de roteamento é necessário que sua camada inferior lhe garanta, com confiabilidade, todos os serviços que esta venha solicitar. Para dar suporte a arquitetura proposta e ter como referência um protocolo da camada de acesso ao meio que proverá os serviços requisitados pela camada superior, foram analisados diversos protocolos, entre eles, os protocolos ADAPT, ABROAD e CATA [11], baseados em contenção probabilística; CHMA e MACA-CT [11], que empregam salto em freqüência e espalhamento de espectro, entre outros.

Neste estudo, a rede proposta tem como necessidade principal, um protocolo que promova o acesso às estações envolvidas na comunicação e que impeça outras de gerarem tráfegos de colisão, principalmente as chamadas "estações escondidas". Estes pontos são principalmente focados pelo protocolo MACAW [12], onde algumas modificações foram inseridas para melhor adaptar a realidade de funcionamento da rede para a arquitetura proposta.

1.4. Projeto

A proposta principal deste trabalho é a de formalizar uma arquitetura genérica para redes *ad-hoc*, possibilitando dar suporte as camadas superiores, permitindo a aplicação, por exemplo, de serviços com garantia de QoS e onde todos os protocolos desta arquitetura pudessem ser testados e avaliados de uma forma científica, cuja validação fosse feita com critérios mensuráveis e não empíricos, assim haveria uma grande diminuição nas horas atribuídas com testes de campo do protótipo ao final do projeto [13].

Seria uma forma puramente científica de formalizar, analisar e testar um projeto composto de um conjunto de protocolos interagindo entre si, entre camadas e através das suas PDUs e SDUs [14].

Para tanto foram avaliados diversos protocolos onde três protocolos foram escolhidos, o MACAW para acesso ao meio, que estudava com profundidade científica o efeito do terminal escondido, um dos principais problemas em redes móveis e *ad-hoc*.

Para os protocolos de roteamento, a rede foi abordada de duas formas:

-para redes de tamanho pequeno a mediano, de 2 até 10 estações, a rede foi tratada com protocolos reativos, especificamente o protocolo DSR;

-para redes maiores e/ou cujas estações pudessem ser divididas em células hierarquicamente organizadas, o protocolo a ser utilizado foi o LANMAR.

Este trabalho segue o seguinte roteiro: capítulo 02, descrição da arquitetura da rede *ad-hoc*, incluindo a descrição dos protocolos estudados, testados e implementados; capítulo 03, projeto composto de metodologia, especificação, verificação e modificações propostas e inseridas nos protocolos MACAW, DSR e LANMAR; capítulo 04, resultados obtidos com as simulações e conclusões de ordem experimental; capítulo 05, conclusão final.

2. Arquitetura para Redes *Ad-hoc*

2.1 Introdução

Este capítulo apresenta a modelagem da arquitetura da rede, juntamente com seus aspectos qualitativos e quantitativos, bem como definições sobre as estações de comunicação desta.

Apresenta também um diagrama de camadas que ilustra a troca de diretivas, PDUs e SDUs [14], inter e entre camadas da rede, exemplificando como seria a interconexão destes protocolos [15] internamente a cada estação e a troca de mensagens entre células ou, mesmo outras estações.

Aqui são descritos os protocolos estudados e feita a justificativa para os escolhidos, tendo como elemento de julgamento as propriedades essenciais e necessárias para um funcionamento eficiente para as redes *ad-hoc*.

Definem-se aqui as propriedades principais para um bom desempenho de cada protocolo da rede, salientando a característica de busca reativa de rotas para os de roteamento, e a resolução do problema da estação escondida para os de acesso ao meio.

Neste capítulo, o item 2.2 descreve o modelo da rede e a arquitetura em camadas; o item 2.3 faz uma descrição dos protocolos de roteamento para redes *ad-hoc*; o item 2.4 descreve os protocolos de acesso ao meio estudados e o item 2.5 faz um comentário geral sobre este capítulo, interligando-o com o capítulo seguinte.

2.2 Modelo das Estações e da Rede

Para fins de estudo, considera-se como modelo de cenário da rede um sistema que possua estações dotadas de totais capacidades de transmissão e recepção, cada uma destas com um padrão independente de mobilidade, deslocando-se de forma restrita a uma área tal que a potência dos transmissores permita o estabelecimento de enlaces ponto a ponto, pelo menos, até o centro da rede, ou da célula.

Todos as estações têm capacidade de processar todos os protocolos da arquitetura proposta, sendo dotados do poder de autogerência, o que lhes possibilita iniciar a busca de novas rotas e montar tabelas destas, para análises posteriores. As estações dispõem de autonomia para se desligarem, independentemente de estarem inseridos em rotas ou serem núcleos de células, podendo sair e entrar na rede várias vezes e em locais diferentes.

A figura 01 ilustra uma rede simples e a figura 02, uma rede dividida em células, as duas são modelos para desenvolvimento dos experimentos com os protocolos descritos no item 2.3.

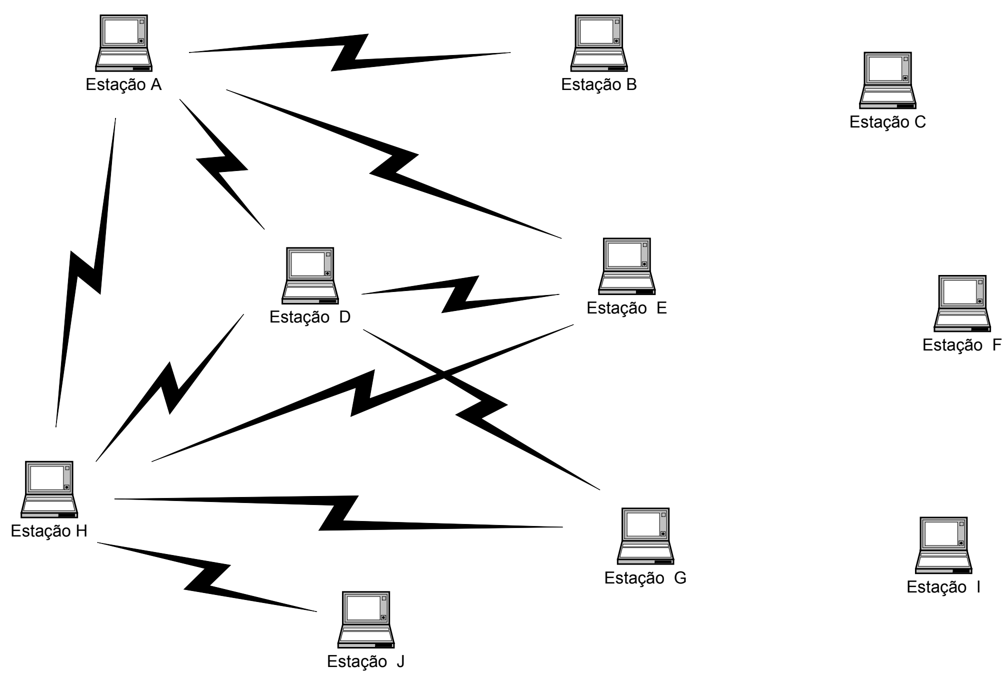

Figura 01 - Modelo da Rede

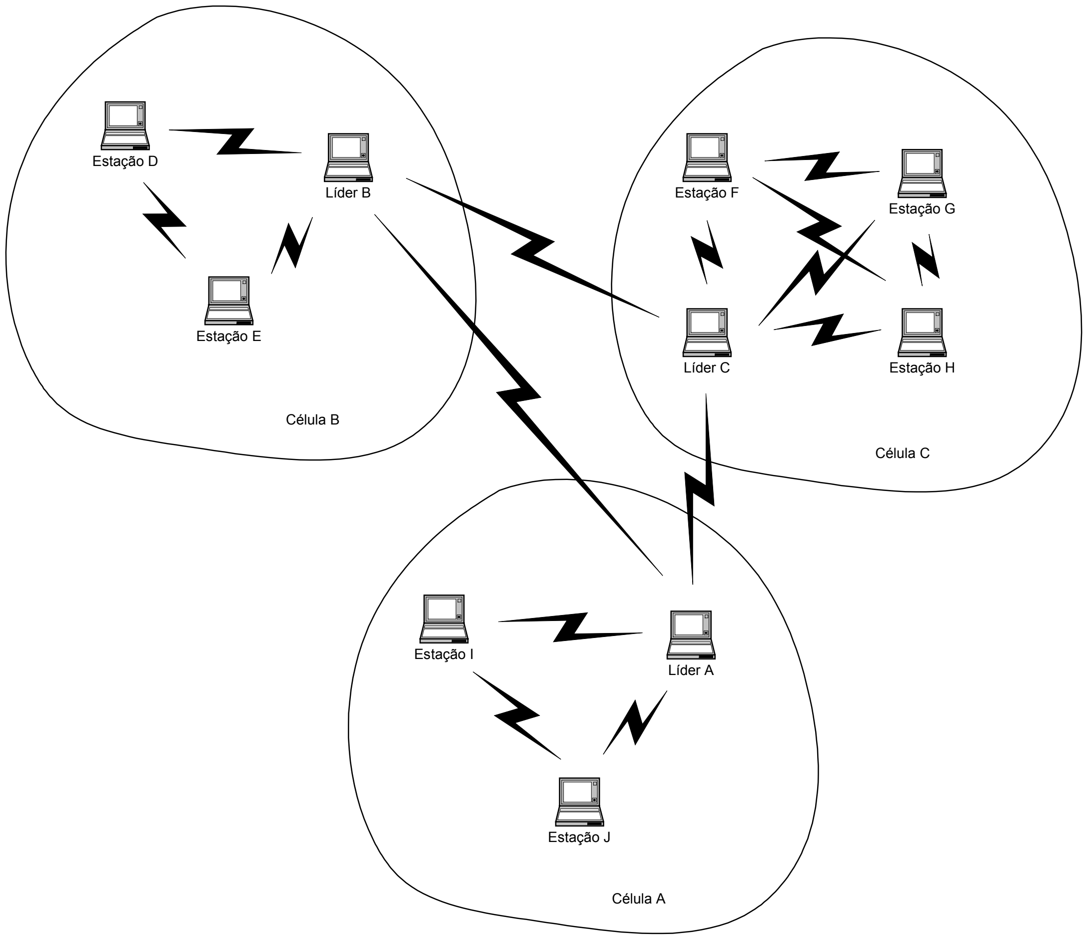

Figura 02 - Modelo da Rede em Células

2.2.1. Arquitetura da Rede Proposta

Dois problemas se destacam ao se planejar a especificação da arquitetura da rede: a dimensão exageradamente grande do conjunto de alternativas, fazendo com que o conjunto de estados e de soluções tenda a infinito, tornando o problema de difícil abordagem matemática e a impossibilidade de representação explícita de restrições temporais, *time out*, quantitativas, na linguagem de especificação formal utilizada, o que se mostra como obstáculo à expressão de todas as funcionalidades dos protocolos.

O tempo, *time out*, é representado dentro de cada processo, ou melhor, entre a troca de cada mensagem, em cada iteração como uma ação alternativa na falha da mensagem anterior, não impedindo com isso, de ser calculado como uma ação alternativa. A inserção do tempo desta forma permite a correta representação de eventos que respeitam uma seqüência temporal, mediante o emprego de enlaces extras para fins de sincronização. Com isso, garante-se que restrições de causalidade não são violadas, como, por exemplo, o disparo de um temporizador do nó origem do protocolo de roteamento [16] deve ocorrer tão somente em caso de perda de pacotes ou mudança de rota, o que se verifica através da sincronização de canais entre a ação de perdas de pacotes e o processo tempo que represente tal situação.

O diagrama de camadas mostrado, na figura 03, representa toda troca de mensagens inter e entre camadas através de suas SDUs e PDUs.

Há de se salientar que a camada de aplicação é solicitada por algum “usuário”, através das mensagens ACCept e DELiver e esta por sua vez encapsula e replica tal solicitação a camada de roteamento, assim por diante, até que a camada de acesso ao meio faça a solicitação através do meio físico e todo este processo é repetido do lado do receptor.

É necessário lembrar que o diagrama da figura 03 foi mostrado apenas para duas estações, para facilitar a visualização do mecanismo de funcionamento dos protocolos, mas como será mostrado posteriormente no item 3.3, o número de estações e usuários vão bem além de dois. Assim, toda comunicação é ponto a ponto, mas toda mensagem será propagada por inundação na rede, ou seja, ponto a multiponto.

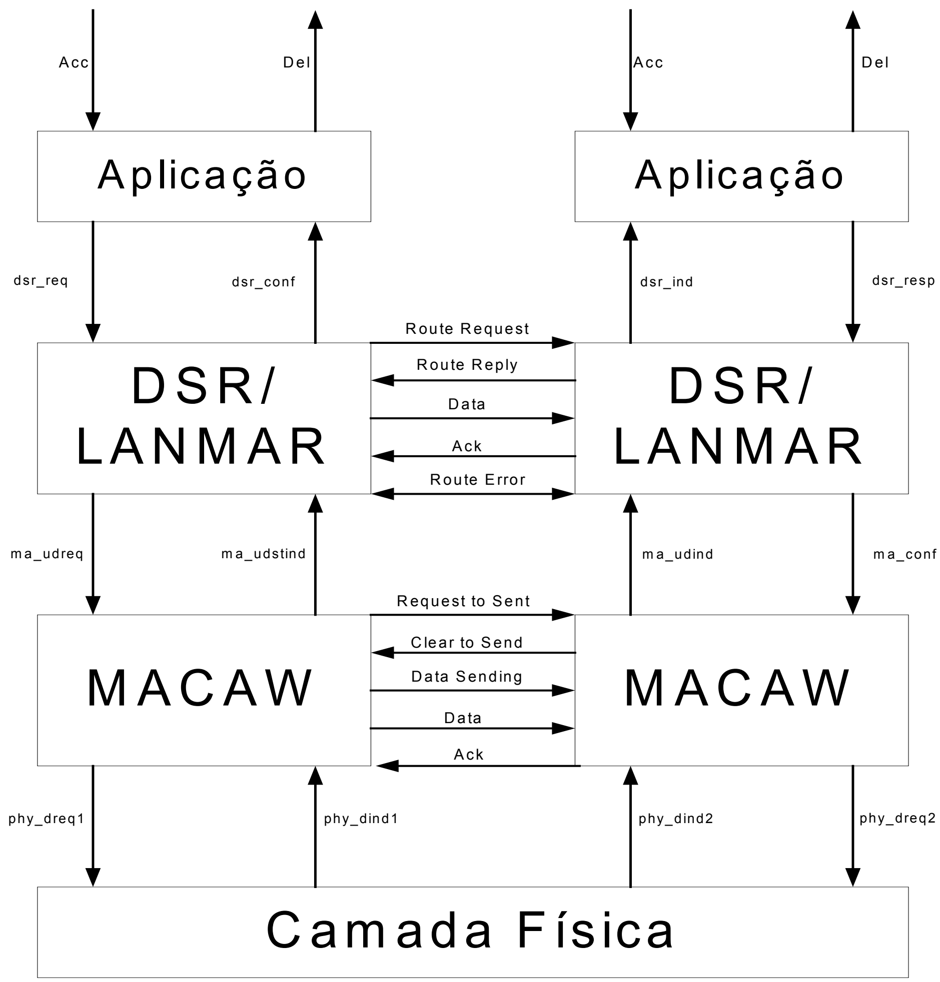

Figura. 03 - Estruturação dos Protocolos em Camadas

2.3. Descrição dos Protocolos de Roteamento em Redes *Ad-hoc*

Uma rede *ad-hoc* é muito dinâmica por natureza, ou seja, há uma movimentação muito grande das estações e entre as estações em direções aleatórias. Estações podem estar muito próximas umas das outras num instante e em seguida podem estar em bordas opostas da rede. Podem estar sendo conduzidas por vários meios de locomoção e com velocidades diferentes, caracterizando uma rede com diversos usuários distintos, com características e necessidades bem peculiares.

Estas situações podem ser descritas para algumas dezenas de estações ao mesmo tempo, acrescentando ainda o fato de algumas estações se desligarem e se religarem em outro ponto qualquer da rede, o número de posições das estações na rede e a quantidade de combinações de vizinhos nó a nó por unidade de tempo é infinita.

Esta diversidade de formações das estações na rede impossibilitaria qualquer nó de ter armazenado consigo, em memória, a posição, o endereço, de cada estação em tempo real. Para que a informação de posição de cada estação não seja um problema é necessário que se tenha uma maneira de obtê-la no momento oportuno e sem acarretar excesso de espaço de armazenamento em memória dentro de cada estação.

A obtenção do endereço de uma estação por uma outra, por onde e através de quais estações esta consegue transmitir dados para um determinado destino, é uma das questões chaves do bom funcionamento das redes *ad-hoc*. Esta incumbência foi legada a camada de rede, através de seus protocolos de roteamento que são divididos basicamente nos seguintes tipos [4]:

a) Pró-ativo: O protocolo de roteamento periodicamente distribui informações de roteamento através da rede para pré-computar caminhos para todos os possíveis destinos para aquela estação. Este método funciona bem para redes semi-estáticas e pouco móveis, mas não é bom para redes grandes dinâmicas ou altamente dinâmicas;

b) Inundação: Os pacotes são enviados em *broadcasting* para todos as estações com a esperança de que cheguem ao seu destino. O excessivo tráfego gerado não é bom para redes grandes ou altamente dinâmicas;

c) Reativo: Quando um nó deseja enviar uma mensagem, faz uma busca através da rede, de estação para estação, para determinar uma rota para o destino desejado. Em função da demanda e do tamanho da rede, gera um grande volume de fluxos de dados com o controle de tráfego e na procura de rotas;

d) Roteamento dinâmico baseado em células: A rede é dinamicamente organizada em células, que por sua vez pode estar dentro de outras células hierarquicamente organizadas, hierarquias horizontais e verticais. O objetivo é dividir o problema para que se possa obter uma efetiva estabilidade da topologia da rede. Geralmente utiliza protocolos pró-ativos dentro de suas células e reativo entre células.

Numa análise geral das formas de roteamento descritas acima a de se notar que cada uma tem seu ponto forte em determinado tipo de rede, por causa de uma determinada característica, bem peculiar; que por ironia também lhe é a causa da sua inviabilidade em outros determinados tipos de redes.

A maioria dos protocolos de roteamento acaba sendo construída, escrita, tendo como característica principal uma das enunciadas anteriormente, o que se pode fazer é submetê-los a testes, analisando suas características, suas falhas e a viabilidade de modificações destas para melhor atendimento das necessidades gerais de qualquer, ou da maioria, ou de um determinado tipo de rede *ad-hoc* na qual se deseja trabalhar.

Os protocolos de roteamento estão documentados em *drafts* no IETF, no grupo MANET. Eles utilizam uma combinação das formas de roteamento anteriormente citadas.

São eles:

TBRPF (*Topology Broadcast Based on Reverse-Path Forwarding*) [17] é um protocolo que combina as formas pró-ativa e inundação, porém com alguma otimização, por meio da divisão de seu mecanismo de roteamento em dois:

Descoberta de Vizinhos - este se preocupa em descobrir os nós vizinhos dentro de um raio de alcance determinado por um certo número de saltos. Aos nós vizinhos, propaga a tabela de nós e interfaces disponíveis ativas (dizendo se os enlaces são bidirecionais ou unidirecionais) ou inativos, através de troca de mensagens *HELLO*.

Módulo de Roteamento - procura propagar a parte da topologia que compreende o ramo da rede ao qual aquele nó pertence, ou seja, propaga partes diferentes da tabela de roteamento de acordo com cada enlace, deixando de sobrecarregar a rede. Além disso, propaga apenas as modificações sofridas pela rede. A atualização total das tabelas só será feita em instantes programados.

FSR (*Fisheye State Routing*) [18] trata-se de um protocolo pró-ativo e por inundação a um só tempo, utiliza o método do "olhar de peixe". Neste mecanismo, são muito intensas as trocas de mensagens e tabelas de roteamento para os nós próximos ao nó emissor; sua intensidade vai diminuindo conforme vai aumentando a distância em relação ao nó de origem da mensagem. Assim, quanto mais distantes os nós, suas tabelas são refeitas em intervalos de tempo maiores, diminuindo a precisão com a distância. Porém, estes nós também usam este mesmo mecanismo no sentido inverso. Com isso, existe uma boa precisão nas tabelas próximas aos nós origem e destino e uma precisão média nos nós intermediários.

AODV (*Ad-hoc On-Demand Distance Vector Routing*) [19] é um protocolo reativo que trata a rede como um todo. Os nós propagam mensagens *HELLO* periodicamente para manter atualizada as tabelas e estados destes. A busca de novas rotas é feita por meio de mensagens *Request* e *Reply* para formar as tabelas de roteamento. Este mecanismo utiliza o artifício de um número seqüencial para cada requisição de rota, o qual deverá ser crescente em relação às anteriores emitidas por aquele nó. Da mesma forma, o nó destino ao recebê-la, enviará seu número seqüencial na resposta. Estes números seqüenciais são a forma de correção de erros do protocolo.

DSDV (*Destination-Sequenced Distance-Vector*) [20] é um protocolo pró-ativo no qual todo nó mantém para cada estação da rede um conjunto de rotas, para avaliação da menor distância. Transmite, de tempos em tempos, seus endereços pela rede para atualização das tabelas de roteamento contida nas estações. Estas tabelas atribuem um número seqüencial para cada rota, que é transmitido junto com a origem e o destino para cada pacote de dados enviado.

OLSR (*Optimized Link State Routing Protocol*) [21] é um protocolo onde cada nó elege alguns nós de sua vizinhança para formar o seu conjunto MPR *multipoint relays,* por exemplo, o nó N forma o conjunto (MPR(N)). Da mesma forma, todos os demais nós da rede fazem o mesmo, determinando diversos conjuntos MPRs.

Assim, os diversos nós e MPRs fazem parte de outros diferentes conjuntos MPRs distribuídos pela rede. Estes MPRs por sua vez, montam na sua tabela os nós que lhe atribuíram esta designação. Com esta estrutura formada, os nós só enviam e recebem mensagem pelo seu respectivo MPR e estes MPRs se comunicam entre si e com seus respectivos nós que o elegeram. Desta forma, qualquer nó que receba um pacote de um outro nó, que não seja seu MPR, não propagará este pacote adiante. Este mecanismo tem como objetivo diminuir a inundação de mensagens sobre a rede.

ZRP (*The Zone Routing Protocol (ZRP) for Ad-Hoc Networks*) [22] este protocolo trabalha com a idéia de zonas de roteamento, onde cada estação é o centro de sua zona cujo tamanho é variável em função das características de estabelecimento e funcionamento da rede. Cada estação ao criar sua zona de roteamento estabelece a quantos “saltos” está a borda de sua zona. Definido o tamanho da borda e da zona, esta estação guarda os endereços de todos os seus vizinhos e assim fazem todos as demais estações da rede. O ZRP não possui elemento centralizador, todas as estações estabelecem suas zonas de roteamento. Feito isto, o ZRP define três tipos de protocolos que funcionaram em três áreas distintas desta rede:

-IARP (*IntrAzone Routing Protocol*) que trabalha internamente àquelas zonas determinas pelas estações da rede [23].

-ERP (*IntErzone Routing Protocol*) que trabalha externamente e entre àquelas zonas determinas pelas estações da rede [24].

-BRP (*Bordercast Routing Protocol*) que trabalha sobre as bordas daquelas zonas determinas pelas estações da rede [25].

Estes três também são protocolos de roteamento descritos em *drafts* do IETF para redes *ad-hoc*.

IARP [23], em seu *draft* são definidas diretrizes básicas para este tipo de protocolo, porém não se afirma realmente qual deverá ser utilizado. Comenta apenas que, pelo fato da zona ser formada por uma estação e esta trocar mensagens para o estabelecimento da mesma, obrigatoriamente ela deverá conhecer o destino de todas as demais estações presentes na zona. Assim, necessariamente deve ser utilizado um protocolo pró-ativo como os descritos anteriormente, por exemplo, o FSR e o OLSR.

IERP [24], este protocolo se preocupa com a comunicação entre zonas previamente estabelecidas. Ele não tem necessariamente que saber a localização de todas elas, com isto o *draft* que o define traça como diretrizes básicas para este tipo de protocolo aquelas características definidas anteriormente como reativas e utiliza para exemplificar, alguns dos protocolos já definidos pelo IETF como, por exemplo, o DSR e o AODV.

BRP [25], este protocolo trabalha sobre as bordas daquelas zonas determinadas pelas estações da rede. Faz a conexão e o interfaceamento entre os protocolos IARP e IERP. Tem características híbridas, tanto pró-ativo (internamente as zonas), quanto reativo externamente as zonas estabelecidas. É o elemento de tradução, ou melhor, de ajustamento entre protocolos internos (IARP) e externos (IERP) as zonas de roteamento.

LANMAR (*Landmark Routing Protocol for Large Scale Ad-Hoc Networks*) [26], este protocolo divide a rede em células com elementos centralizadores e hierarquicamente distribuídos ao longo da rede. Apresenta o conceito de grupo de células, com um nó desempenhando a função de líder para cada célula. O LANMAR tem por objetivo otimizar redes cujos grupos de trabalho podem se dividir em conjuntos, as células podem ser grupos de combate em um campo de batalha, equipes médicas ou de defesa civil em meio a uma catástrofe, grupos de trabalho e/ou pesquisa em uma planta industrial/comercial onde se deseja comunicação e transferência de dados em tempo real.

O protocolo LANMAR cita a utilização do FSR [16] internamente às células e o mecanismo de número seqüencial de identificação de rotas, conforme descrito no DSDV [20], para as estações líderes de células.

Quando a rede se inicia, as células ainda não existem, cada estação começa a trocar mensagens com seus vizinhos para a decisão do líder das células a serem formadas. A estação vencedora, que pode ser a estação com o maior número de vizinhos alcançados, maior potência de sinal, localização geográfica, etc, se identifica como líder e monta sua tabela com todas as estações da célula. Estas trocam mensagens entre si, utilizando protocolos como o FSR, para montar suas tabelas com as rotas das estações vizinhas. Deste momento em diante toda ligação feita dentro da célula é baseada na tabela de rotas montada sem a utilização da estação líder como intermediária.

Nas comunicações com nós fora da célula, as estações subordinadas enviam mensagens de busca para seus respectivos líderes de célula, estes difundem as mensagens, aos demais líderes da rede. Como todos as estações líderes têm a tabela de rotas com as estações participantes de sua célula esta mensagem é transmitida diretamente de líder para líder e estes informam a melhor rota a ser seguida pela mensagem, chegando ao usuário final de forma transparente.

As células também podem estar em níveis hierárquicos distintos, aumentando a árvore hierárquica da rede.

O LANMAR usa estas estratégias com o objetivo de aumentar a eficiência da rede, por meio da divisão desta em células, permitindo o fluxo de mensagens entre estas apenas através de seus líderes, gerando hierarquia de estações e de mensagens [14].

DSR (*Dynamic Source Routing Protocol*) [14] é um protocolo reativo que se utiliza de dois mecanismos principais: a descoberta e a manutenção de rotas e será explicado com mais detalhe a seguir. Este protocolo é essencialmente reativo, de mais simples implementação que o AODV.

O DSR tem melhor desempenho que o AODV em redes cujas estações se movimentam com velocidades baixas até moderadas; em velocidades altas o AODV é mais eficiente, porém o *overhead* gerado por ele é bem maior, devido à troca de mensagens *HELLO* para atualização de posição das estações. Esta estrutura não existe no DSR, por isso seu *overhead* será menor em qualquer condição de funcionamento da rede [27].

O DSR tem como característica minimizar as tabelas de rotas e conseqüentemente a necessidade de grande espaço em memória. As trocas de mensagens são ocasionais e os pacotes são menores. Atualmente, é o protocolo mais antigo do IETF para roteamento *ad-hoc*, o de especificação mais completa, o mais difundido e o mais testado dentre os existentes [8, 28].

Por todas estas atribuições este foi o protocolo de roteamento reativo escolhido, haja vista a arquitetura da rede proposta ser a mais genérica possível, o que credencia o DSR como um dos mais fortes candidatos a testes e simulações considerando as conclusões dos trabalhos referenciados anteriormente [8, 28].

O protocolo de roteamento dinâmico pela fonte, DSR, funciona baseado em duas diretrizes principais: a descoberta e a manutenção de rotas [14, 29].

Descoberta de Rotas (*Route Discovery*)

Quando um nó recebe um pacote, ele o coloca em um *buffer* para transmissão *Send Buffer*; dada a inclusão do endereço de destino no pacote, o nó tenta enviá-lo para o destinatário final através da rede.

Em primeiro lugar, promove-se uma consulta à sua tabela de rotas *Route Table*, para saber se possui uma rota conhecida para aquele destino. Caso não disponha, o nó inicia o processo de descoberta de rotas *Route Discovery*, por meio da difusão do pacote de requisição de rota *Route Request*, a qual contém o nó de origem, o de destino e a identificação da mensagem, Id.

Cada nó da rede que recebe o pacote de requisição de rota verifica se aquele pacote já fora recebido anteriormente, através da verificação do Id e do nó origem: caso afirmativo, a mensagem é descartada; caso contrário, o nó transmite um pacote de reconhecimento *ack* ao nó de origem, além de verificar se ele próprio é o nó de destino ou se ele dispõe uma rota para este, armazenada em seu *Route Table*. Caso exista uma rota armazenada ou ele próprio seja o nó procurado, é transmitida uma mensagem de resposta *Route Reply* para a origem com o seu endereço e o caminho a seguir. Se não for encontrada uma rota ao nó de destino, o nó coloca o seu endereço como o próximo nó intermediário seqüencial e retransmite, por inundação.

O próximo nó que receber a mensagem fará o mesmo, até que o *Route Request* seja encaminhado ao nó de destino, o qual transmitirá um *Route Reply* para o nó origem, com a seqüência dos nós intermediários pelos quais a mensagem de requisição passou até chegar ao destino. Cada nó intermediário no encaminhamento do *Route Reply* armazena a rota nele contida, compondo sua própria *Route Table.*

Manutenção de Rotas (*Route Maintenance*)

A manutenção de rotas é feita da seguinte forma: cada nó que transmite o pacote é responsável por armazenar a confirmação de recebimento pelo nó seguinte. Seja, por exemplo, uma rota entre os nós A e E, assim discriminada: A - B - C - D - E: neste caso, o nó A deve receber a confirmação de resposta *ack* do nó B, o nó B aguarda o *ack* do nó C e assim até o nó D, que armazena a confirmação enviada pelo nó E, sem a necessidade de retransmissão destas confirmações até o nó de origem.

Caso haja uma ruptura em meio à rota, o nó que encaminhou o pacote ou mesmo a requisição de rota, ao não receber o *ack* de seu nó adjacente, após o disparo de seu temporizador, procura em sua *Route Table*, uma nova rota para reenviar novamente a mensagem ou executa um novo *Route Request,* dentro de um limite de tentativa.

Expirado o *time out* da última tentativa, o nó emite ao seu antecessor uma mensagem de erro *Route Error,* mesmo depois de já ter enviado seu *ack*. O nó antecessor terá o mesmo procedimento de seu sucessor, procurando estabelecer uma nova rota; se não obtiver êxito, propaga o *Route Error*" ao respectivo antecessor, em direção a origem. Quando uma estação identifica uma rota falha, dentre as existentes em seu *Route Cache*, torna-a inativa e calcula outras possíveis para aquele destino.

2.4. Descrição dos Protocolos de Acesso ao Meio, Protocolos de Suporte aos Protocolos de Roteamento

Para dar suporte a arquitetura proposta e ter como referência um protocolo da camada de acesso ao meio, que proverá os serviços requisitados pelo protocolo de roteamento, foram analisados diversos protocolos da camada dois.

Existem os mais variados protocolos de acesso ao meio para redes *ad-hoc*; alguns tratando este acesso probabilisticamente, outros interagindo diretamente com os nós para identificar as estações que estão transmitindo naquele momento, esperando o instante de tempo adequado para estabelecimento da comunicação.
Pode-se citar:

-o FAMA [30] garante transmissões livres de colisões usando detecção de portadora;

-o MACA [31] utiliza o mecanismo de troca de mensagens RTS-CTS-"DATA". Este protocolo apresenta o problema do terminal escondido [12], em estações que estejam próximas do transmissor, por estas não escutarem a resposta do receptor, podendo gerar colisão de pacotes;

-o MACA-BI [11] descreve modificações no RTS do protocolo MACA [31], tornando mais robusto o controle de colisão dos pacotes;

-os protocolos CHMA e MACA-CT [11] empregam salto em freqüência e espalhamento de espectro, o que requer uma capacidade maior de processamento das estações da rede;

-o HRMA [32, 33] é utilizado para múltiplos acessos e tem como principal forma de controle de acesso ao meio a passagem da permissão de transmissão entre estações adjacentes. Emprega salto em freqüência e espalhamento de espectro, o que a permite fazer múltiplos acessos num determinado instante e que também a faz requerer uma maior capacidade de processamento das estações da rede;

-os protocolos ADAPT, ABROAD e CATA [9] são baseados em contenção probabilística, não são eficientes em redes cujas estações são muito móveis, de diferentes capacidades de processamento, com uma utilização muito variável dos meios de comunicação e com alto índice de ingresso e saída de nós na rede.

Pode-se observar que existem inúmeros protocolos de acesso ao meio, com as mais diversas características, tentando resolver os mais variados problemas do controle de acesso ao meio em redes sem fio *ad-hoc*. Para descoberta de rotas em protocolos de roteamento reativos é preciso que o protocolo de acesso ao meio, antes de tudo, minimize o problema do terminal escondido, visto que a rede *ad-hoc* é muito dinâmica, apresentando, portanto, clientes e servidores altamente dinâmicos. Assim, diminuir as colisões, a troca e o reenvio de mensagens desnecessárias é essencial para se minimizar o *overhead* na rede.

Outro ponto de destaque é que cada terminal poderá estar prestando um determinado serviço e requisitando outro, completamente diferente, conferindo à rede uma infinidade de possibilidades de tráfego, em intervalos de tempo variáveis. O tratamento estatístico de alguns protocolos não absorve todas essas possibilidades ao longo do tempo. Portanto, há a necessidade de um protocolo que promova o acesso às estações envolvidas na comunicação e que impeça outras de gerarem tráfegos de colisão, principalmente as chamadas "estações escondidas". Estes requisitos de projeto são especificados no protocolo MACAW [12], justificando sua adoção na arquitetura proposta.

O MACAW [12] é um protocolo de acesso ao meio para a arquitetura de redes sem fio dotadas de um simples canal, podendo ser classificado como de múltiplo acessos e baseado na escuta, verificação e tomada de posse da transmissão no canal. Ele é derivado do protocolo MACA, que utiliza a estrutura de troca de pacotes e um algoritmo de *back-off* exponencial binário para selecionar o tempo de retransmissão, um *time out*, de cada pacote, onde ele presume que houve uma colisão, o que nem sempre é verdadeiro[31]. O MACAW pode ser descrito como a evolução do MACA, pela inserção do DS, do ACK e do novo algoritmo de *back-off*, isto lhe proporcionou uma maior eficiência, comprovada experimentalmente em [12].

No protocolo MACAW, é inserido um pacote DS (*Data Sending*) para fazer uma pré-ocupação do canal, diminuindo a taxa de colisão, comprovada experimentalmente em [12]. O ACK foi inserido como resposta para informar do fim da transmissão e da desocupação do canal.

A rede começa com todos os nós em silêncio e na escuta do canal de comunicação. Quando um nó recebe um pacote e deseja transmiti-lo, escuta o canal e, após verificar que não há nenhuma mensagem trafegando, transmite um pacote RTS (*Request to Send*) contendo seu endereço, o do nó destino e o tamanho da mensagem. Ao receber o RTS, o nó destino emite imediatamente um CTS (*Clear to Send*) confirmando o tamanho da mensagem a ser recebida. Ao receber o CTS, o nó de origem emite um DS, que é um pacote pequeno e rápido. Com o envio do pacote DS, o canal fica ocupado, o destinatário se prepara para receber os dados e as outras estações da rede, que escutaram esta troca de pacotes, acionam seu algoritmo de b*ack-off*, que determinará um novo tempo de espera - isto impede que um outro nó tente fazer uma comunicação.

É importante observar que as estações permanecem escutando o canal e ao ouvirem qualquer pacote trafegando, imediatamente acionam seu algoritmo de *back-off* .

Após o emissor transmitir o DS, são transmitidos os dados; desta forma, o emissor espera a confirmação do receptor através de um ACK. Após a troca de mensagens e dados, as estações voltam para o estado inicial de escuta.

2.5. Comentários

Foram estudados alguns dos principais protocolos de acesso, com as mais diversas características. Para dar suporte a descoberta de rotas em protocolos de roteamento reativos é preciso que este tipo de protocolo, antes de tudo, minimize o problema do terminal escondido, diminuindo as colisões, o reenvio e a troca de mensagens desnecessárias, condições essenciais para se amenizar o problema de *overhead* na rede. Permitindo que o tráfego de informações flua sem congestionamentos, possibilitando uma maior eficiência na busca de rotas, mecanismo fundamental para o bom funcionamento dos protocolos de roteamento.

Para o MACAW, existe a necessidade de um estudo mais profundo visando às transmissões por inundação [12], tal abordagem será feita neste trabalho, com o intuito de criar novas propostas para transmissão por inundação usando este protocolo e outros com iguais características.

Para os protocolos de roteamento foi dado enfoque a protocolos que pudessem atender a rede independente do tamanho desta. Para redes pequenas e pouco móveis a maioria dos protocolos cumpre a finalidade, mas para redes grandes é essencial à utilização de um protocolo que utilize tabelas de rotas de forma mais inteligente e faça busca de rotas com menos *overhead*.

Por isso a proposta da utilização do protocolo LANMAR para os variados tipos de rede. O líder de célula criado neste tipo de arquitetura também propicia a eleição de um detentor dos endereços dos serviços residentes naquela célula, já prevendo a utilização de protocolos trabalhando com descoberta de serviços da camada cinco, facilitando a difusão dos serviços com QoS para as demais estações das outras células da rede.

Na descrição do IETF para o LANMAR, o roteamento dentro e fora das células é feito por protocolos pró-ativos modificados, em função de sua posição intra ou inter célula. Para a ligação entre as células, este trabalho propõe a utilização de protocolos de roteamento reativo, como o DSR e o AODV, por serem mais eficientes para redes dinâmicas [9,10].

Foi proposto o DSR por ser de mais simples implementação, além de evitar o *overhead* acrescentado pelo AODV, quando da troca de suas mensagens *HELLO*.

Como outras vantagens em relação ao AODV, o DSR troca mensagens apenas sob demanda e os pacotes são menores. Atualmente, é o protocolo mais antigo do IETF para roteamento *ad-hoc*, o de especificação mais completa, o mais difundido e o mais testado dentre os existentes [8, 9, 10, 28]. Tal protocolo será objeto de estudo para o roteamento de estações inter células, em apoio ao protocolo LANMAR, que também será objeto de estudos e de experimentos.

3. Projeto

3.1. Introdução

Existem áreas do conhecimento que é possível prever todas as hipóteses de funcionamento em determinado ambiente, nas mais variadas condições de utilização. Sendo assim, existirá um conjunto de soluções bem claro, conciso e completo. O mapa de soluções possíveis é um número finito e razoável do ponto de vista computacional.

Para protocolos distribuídos, o conjunto de estados e de soluções tende para infinito e pode-se mostrar que a simples verificação de propriedades essenciais, tal como *deadlocks* é um problema bastante complexo [34].

Para validar um protocolo é necessário o uso de ferramentas que testem suas características em todas as hipóteses possíveis da rede, até as mais improváveis. Este trabalho utiliza métodos formais, porque estes são calcados em princípios matemáticos que permitem uma modelagem e posteriormente, verificação e análise de forma genérica, precisa e sem ambigüidades.

No caso de protocolos para redes *ad-hoc*, onde não há um responsável pela rede, um gerente de rede, o algoritmo é distribuído, o que causa certas complicações para uma concisa verificação. Na abordagem de sistemas com algoritmos distribuídos se deve, antes de tudo, verificar se ele atende a algumas propriedades tais como, se existem *deadlocks*, *livelocks*; se é vivo, inicializável, entre outras.

Neste capítulo, o item 3.2. apresenta uma metodologia da forma de desenvolvimento aplicada neste projeto; o item 3.3 a especificação dos protocolos, com explicação da linguagem LOTOS, algumas de suas bibliotecas e a ferramenta CADP; o item 3.4. Especificação e Verificação do Protocolo MACAW; o item 3.5. Especificação e Verificação do Protocolo DSR; o item 3.6 Especificação e Verificação do Protocolo LANMAR e o item 3.7 os comentários finais do capítulo, fazendo alusão aos resultados obtidos com as simulações do capítulo seguinte.

3.2. Metodologia

A escolha de protocolos que atendam a arquitetura da rede proposta é o primeiro passo deste projeto e podem ser encontrados em *drafts* do IETF [35] na Internet.

Definidos os protocolos, as linguagens de simulação e as ferramentas a serem utilizadas, parte-se para a modelagem da rede, dos nós e do ambiente de simulação. Como por exemplo, a distribuição dos nós na rede, o número de nós, velocidade dos nós, o ambiente de transmissão dos pacotes, taxa de transmissão, número de canais disponíveis, taxa de ocupação dos canais, etc.

Todos os parâmetros serão limitados pelas características de cada protocolo, o não atendimento das necessidades do projeto implica na modificação ou troca dos mesmos.

O ponto principal de formação da rede e do formalismo, para escrita do protocolo em LOTOS, é que cada um deles foi especificado considerando que o protocolo da camada imediatamente inferior a sua lhe prestava os serviços necessários e corretos, sem problemas ou erros, para o seu bom funcionamento. Assim, cada um deles teve seu tratamento em separado, dividindo o problema em várias partes e posteriormente interligados pela troca de PDUs e SDUs entre si, baseado-se no modelo de camadas mostrado na figura 03, do item 2.2.

É necessária uma boa modelagem do nó, internamente, peculiar a cada protocolo e com as funcionalidades básicas que atendam a todos os protocolos ao mesmo tempo. Este deve garantir que seu funcionamento interno seja consistente com todos eles, simultaneamente.

É necessário atribuir algumas considerações a cada protocolo para balizar a descrição destes e para melhorar a clareza dos testes e simulações.

Por exemplo, na modelagem do protocolo MACAW considerou-se que ele é um protocolo de acesso ao meio para a arquitetura de redes sem fio para um simples canal, canal único, baseado na escuta, verificação e tomada de posse da transmissão do canal. Suas mensagens, pacotes, de controle são consideradas pequenas.

Nos protocolos LANMAR e DSR o canal de comunicações entre os nós era confiável e disponível a todos, já provido pelo protocolo imediatamente inferior a ele, o MACAW.

Todos os serviços oferecidos pelas camadas inferiores a eles estão disponíveis e funcionam corretamente.

Segue um roteiro da metodologia empregada neste projeto:

-pesquisa e definição da arquitetura da rede *ad-hoc* e dos protocolos que satisfaçam a arquitetura escolhida;

-modelagem formal em LOTOS básico e verificação de propriedades essenciais ao funcionamento do protocolo. A cada nova versão do protocolo é necessária a validação do mesmo, para que ele possa ser modificado sempre com incremento em complexidade das estações e do número destas na rede. Quando seus mecanismos de funcionamento já estão dominados para uma rede de características próximas as reais, inicia-se a próxima etapa do projeto;

-modelagem formal em LOTOS completo, com passagem de parâmetros entre estações, e verificação de propriedades essenciais a funcionalidades do protocolo, sendo necessária a validação do mesmo para que o protocolo possa ser modificado, incrementando em complexidade as estações e o número destas na rede, até uma boa representação final da rede.

3.3. Especificação e Verificação

Antes de ser feita uma especificação formal de qualquer dos protocolos é necessário uma breve explicação da linguagem de especificação formal LOTOS e da ferramenta CADP. Ambos utilizados para especificações, verificação e validação que referenciaram todas as conclusões e propostas de modificações sobre os protocolos objetos de análises deste trabalho.

3.3.1. Linguagem de Descrição Formal LOTOS

LOTOS (*Language of Temporal Ordering Specification)* [36] é uma técnica de descrição formal padronizada pela ISO (*International Organization for Standardization)* em 1988 [37] para especificação e verificação de sistemas distribuídos e concorrentes de forma geral. Seu projeto foi motivado pela necessidade de existência de uma linguagem que permitisse a elaboração de especificações de protocolos e serviços a um só tempo exatas e independentes de implementação, dotada de uma semântica formal e que desse suporte à verificação e validação de propriedades dos sistemas especificados.

A premissa fundamental de LOTOS [36, 37] 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. Como decisão de projeto em meio aos trabalhos de padronização, LOTOS dispõe de dois componentes básicos: um para representação de dados, outro para descrição de processos e interações.

O componente de dados de LOTOS é dedicado à representação de domínios de valores *sorts*, funções matemáticas *operation*s e expressões algébricas *equations*, as quais são agrupadas em estruturas de dados *types*. Sua notação é derivada da linguagem de especificação de tipos de dados abstratos ACT ONE [38].

O 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 de CCS [39] e CSP [40].

LOTOS, embora destinada originalmente à descrição formal dos protocolos da arquitetura OSI [14, 41], tem sido aplicada igualmente nos trabalhos de validação [42, 43, 44] de diversos sistemas complexos [40]. No ano de 2001, foi concluído um trabalho de revisão e extensão da linguagem, definindo a chamada *Enhanced-LOTOS*. O propósito desta nova linguagem, documentado através do padrão ISO/IEC 15437 [37], é trazer maior simplicidade e expressividade à linguagem LOTOS, especialmente pela maior facilidade na definição das estruturas de dados e a adição do conceito de tempo quantitativo em sua semântica.

Segue a tabela 01 com um resumo das funções e ventos utilizados, em linguagem LOTOS [45, 46], para a descrição dos protocolos a qual se refere esta pesquisa.

Função	**Simbologia**	**Descrição**
Process instantiation	*P[g1,g2,...,gn]*	Instanciação de processos, onde g1, g2, ..., e gn são eventos trocados entre processos.
Exit	*Exit*	Encerramento de processo com sucesso.
Stop	*Stop*	Parada de processo
Endproc	*endproc*	Delimitador do fim da estrutura do processo.
Choice	*A [] B [] C*	Escolha entre os processos A ou B ou C, com igual probabilidade entre elas.
Interleaving	*A|||B|||C*	Funcionamente de processos independentes e em paralelo
Full Synchronization	*A||B||C*	Sincronismo entre processos
General Synchronization	*A|[g1,g2,...,gn]|B*	Sincronismo entre processos através dos eventos g1, g2, ..., e gn.
Hide	*hide g1,g2,...,gn in*	Os eventos g1, g2, ..., e gn serão escondidos na apresentação dos resultados.

Tabela 01 - Resumo das Funções Eventos mais Utilizados na Linguagem LOTOS

Segue a tabela 02 com os tipos de iterações entre eventos, dentro e entre processos [45, 46].

Processo A	**Processo B**	**Condição de sincronização**	**Tipo de iteração**	**Efeito**
g !E1	*G !E2*	Valor de (E1) = Valor(E2)	Casamento de valores	sincronização
g !E1	*G ?x:t*	Valor de (E1)	Passando valores	x = Valor de (E1)
g ?y:u	*G ?x:t*	**t** e **u** são do mesmo tipo	Geração de valores	x = y = E1, onde E1 é algum valor do tipo **t** que é igual a **u**.

Tabela 02 - Tipos de Iterações

3.3.1.1. Bibliotecas em LOTOS

Para toda especificação de protocolo em LOTOS, com passagem de parâmetros, é necessário que se construa uma biblioteca com os parâmetros que farão parte da troca de mensagens pelo protocolo. Esta deve ser criteriosa, pois ela 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.

3.3.2. CADP (Pacote de Desenvolvimento Caesar/Aldebaran)

O Pacote de Desenvolvimento Caesar/Aldebaran é um conjunto de ferramentas de engenharia para protocolos dedicada a compilação, simulação, verificação e teste de descrição formal na linguagem LOTOS, padrão ISO 8807 [37].

O CADP [40, 47] foi desenvolvido pela VASY, a INRIA Rhone-Alpes/DYADE e o laboratório Verimag e é dividido da seguinte forma:

Caesar: é um compilador da especificação em LOTOS para um programa em C ou em LTS para ser verificado usando as ferramentas de bissimulação e/ou avaliadores de lógica temporal.

Aldebaran: é uma ferramenta para verificação de sistemas de comunicação em LTS (*labelled transition systems*), ou seja, transições de máquinas rotuladas. Possui as ferramentas de bissimulação e avaliadores de lógica temporal 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.

Caesar.adt: é usado quando o protocolo em LOTOS trafega dados. Este compilador translada a parte de dados e operações, na especificação em LOTOS, dentro de bibliotecas de tipos e funções, respectivamente, em "C".

Caesar.ident: é um programa que checa a sintaxe da descrição em LOTOS e identifica por onde começar a leitura, da maneira mais fácil e coerente.

Open/Caesar: é uma linguagem, independente do meio, que permite o usuário definir programas para execução, simulação, verificação (parcial, durante a execução, etc) e geração de testes. Permite ao usuário criar seu próprio ambiente de simulação, inserindo seus próprios módulos para suprir suas necessidades. Como por exemplo: ferramentas de execução randômica, ferramentas para busca de seqüência, ferramentas de avaliação de mu-calculus.

BCG: O Gráfico de Código Binário é um formato para representação do Sistema de Transições Rotuladas, ou LTS (*Labelled Transition Systems*), que utiliza a representação binária com técnicas de compressão que reduz o LTS em 20 vezes, se comparado ao ASCII. Várias ferramentas existem para este formato, tais como:

BCG_io: converte o formato BCG em diversos outros tipos;

BCG_draw: cria um gráfico de duas dimensões que representa o BCG num desenho de estados e transições;

BCG_edit: é um editor iterativo que permite a modificação manual do gráfico gerado pelo BCG_draw;

BCG_open: estabelece um "*gateway*" entre o formato BCG e o ambiente Open/Caesar.

XTL (Linguagem Temporal eXecutável): é uma linguagem de programação funcional desenhada para permitir a implementação de vários operadores de lógica temporal, tais operadores são avaliados sobre um LTS codificado em BCG. A linguagem XTL define tipos especiais para um conjunto de estados, transições e rótulos do LTS.

Eucalyptus [48] é uma interface gráfica com o usuário escrita em Tcl/Tk [49], que integra o CADP a diversas outras ferramentas, como APERO [50] (Liege, Bélgica), ELUDO [51] (Ottawa, Canadá), FC2TOOLS [52] e VISCOPE [53] (França).

Das ferramentas descritas anteriormente as mais empregadas foram Caesar, Caesar.adt e Aldebaran, sendo as três propriedades seguintes as mais testadas e empregadas [38]:

Equivalência observacional: todo comportamento externamente observado de determinado processo pode ser igualmente realizado por uma ou mais ações de outro processo; esta relação foi utilizada para verificar se os protocolos especificados para cada camada representavam os serviços discriminados inicialmente como requisitos do projeto.

Equivalência forte: toda ação interna de um processo deve ser igualmente realizada por uma ação interna de outro processo; esta relação foi utilizada para verificar a relação entre implementações incrementais dos diferentes protocolos.

Minimização: 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.

3.4. Especificação e Verificação Formal dos Protocolos

3.4.1. Protocolo MACAW

Na especificação elaborada, foram definidos processos que implementam o protocolo MACAW, interligados por um meio de comunicação simples e exercitados por usuários que implementam troca de dados. A arquitetura empregada para o protocolo MACAW é descrita em LOTOS como, por exemplo, pela expressão de comportamento a seguir:

```
specification MACAW [ACC, DEL, … , ma_conf] : noexit
library     Boolean, NaturalNumber, MACAWLIB2     endlib
behaviour
  hide phy_dreq1, phy_dind1, … , ma_udreq, ma_udstind, ma_conf in
  ((USER1 [ACC,ma_udreq, ma_udstind]
    |||    USER2 [DEL,ma_udind, ma_conf])
   |[ma_udreq,ma_udind,ma_udstind, ma_conf]|
   (MACAW1[ma_udreq, … , phy_dreq1, phy_dind1] (RTS of pdu_type)
    |||    MACAW2[ma_udreq,..,phy_dind2] (RTS of pdu_type))
   |[phy_dreq1,phy_dind1,phy_dreq2,phy_dind2]|
    MEDIUM[phy_dreq1,phy_dind1,phy_dreq2,phy_dind2])
```

```
where
        process USER1[ACC,ma_udreq ...
```

A descrição em LOTOS do protocolo MACAW contém um parâmetro de domínio infinito - os dados transmitidos entre as estações (tipo DATA). Para restringi-lo e permitir a verificação do modelo, utilizou-se um subconjunto das mensagens possíveis, sem prejuízo do modelo. Então, utilizando os compiladores Caesar e Caesar.adt, foram gerados os LTS tanto para o protocolo como para o serviço prestado pela camada.

3.4.2 Descrição do Serviço MACAW

```
specification Macaw_Service [ACC, DEL] : noexit
  library     MACAWLIB    endlib
behaviour
Service [ACC, DEL]
where
  process Service [ACC, DEL] : noexit :=
    ACC ?Data:data_type;
      DEL !Data;
        Service [ACC, DEL]
  endproc
endspec
```

3.4.3. Consolidação dos Resultados das Simulações para o Protocolo MACAW de uma Forma Geral

Obteve-se que os modelos de protocolo e de serviço descritos apresentam equivalência observacional, demonstrando que o protocolo especificado atende aos requisitos do serviço. Os modelos foram submetidos à verificação de equivalência forte, visto que o teste está disponível na ferramenta Aldebaran. O fato de não ter satisfeito este último teste já era esperado, pois naturalmente não foram descritas as ações internas na especificação do serviço MACAW. Além disso, o protocolo foi sempre testado quanto da possibilidade de haver *deadlocks*, se era vivo e reinicializável; condições essenciais para o prosseguimento do projeto.

Estes testes e procedimentos foram feitos em todas as fases do projeto do protocolo MACAW. Em cada experimento foram testadas todas as propriedades e condições descritas anteriormente, desta forma o projeto do protocolo crescia em complexidade apenas a partir de tais condições satisfeitas.

Com o aumento da complexidade, algumas modificações e observações foram feitas para o MACAW e estas estão descritas no item a seguir:

3.4.4. Modificações no protocolo de acesso ao meio MACAW

Como descrito no artigo que propõe o protocolo MACAW [12] este otimiza o protocolo MACA [31] através da troca de RTS-CTS-DS-(*Data*)-ACK mas funcionando entre nós móveis numa rede com uma estação central (estação rádio-base).

Para a rede proposta neste trabalho a idéia de estação central não condiz com a arquitetura onde todas as estações são independentes e podem se comunicar, ou seja, existe a possibilidade da troca de mensagens por todas e entre todas as estações que desejarem se comunicar, indistintamente.

Para este tipo de rede a descrição do protocolo MACAW [12] não funcionaria, nem os anteriormente descritos no item 2.4.

Na literatura existem protocolos para acesso com multi-canais que utilizam "janelas" para sincronizar a troca de mensagens por suas estações nestes multi-canais, trabalham por salto em freqüência como o HRMA.

Para dar suporte aos protocolos de roteamento deste trabalho existem duas possibilidades para os protocolos de acesso ao meio:

-os multi-canais para comunicação entre células e intercélulas, para os protocolos de roteamento que dividem a rede em células, como o LANMAR;

-os de acesso ao meio através de rádios mono canal, para os protocolos que não dividem a rede para o roteamento, como o DSR.

Como dito anteriormente, o protocolo MACAW trata apenas de comunicação em redes que trabalham com estações gerentes centralizadoras, para que ele funcione numa rede onde todos os nós têm a mesma probabilidade de enviar e receber informações; além disso, que funcionam em múltiplos saltos, há a necessidade de algumas modificações nos campos de endereçamento das mensagens.

Nos protocolos de roteamento existem mensagens por inundação, tal como as de descoberta de rotas; e as de endereços definidos, de múltiplos saltos.

Para as mensagens por inundação, estas devem ser diferentes, com estruturas que as identifiquem daquelas que possuem endereços definidos, para que qualquer estação, ao recebê-la, saiba de antemão que deverá propagá-la se não for ela o destinatário final. Para tal, bastará analisar o campo do endereço destino constante nesta mensagem.

O nó ao receber a mensagem RbRTS(nó de origem, nó de destino), enviará o CbCTS(nó de origem, nó intermediário, nó de destino) para que o nó enviador continue a comunicação e posteriormente envie os dados através do DbDS(nó de origem, nó intermediário, nó de destino), *Data*(mensagem propriamente dita). O nó receptor encaminhará o conteúdo do DbDS para a camada de roteamento que o analisará e se o conteúdo for para esta estação, ela responderá com um AbACK(nó de origem, nó de destino). Se não for ele o destino, repetirá o processo até ser encontrado o nó destinatário.

Para mensagens de endereços definidos a mensagem RTS (nó de origem, nó de destino) terá os campos com os nós de destino e de origem para fazer a troca direta entre transmissor e receptor. O nó destino responderá com um CTS(nó de origem, nó de destino) e posteriormente receberá um DS(nó de origem, nó de destino), *Data*(Mensagem), onde o campo *Data* terá a mensagem propriamente dita. A resposta será feita com um ACK (nó de origem, nó de destino).

OBS.: Tanto para mensagens por inundação quanto para aquelas com endereços definidos, a mensagem AbACK, ou ACK, poderá ser respondida apenas pela estação destino da mensagem e repassado estação a estação até chegar à estação origem, fazendo com que todas as estações intermediárias fiquem paradas, sem trabalhar, esperando o AbACK ou ACK, correspondente. Isto pode ser necessário para redes densamente povoadas ou com células pequenas.

Ou feita estação a estação, liberando a estação adjacente para novas trocas de mensagens pela rede.

A diferença entre as mensagens por inundação e as demais é que qualquer estação ao receber uma mensagem do tipo por inundação estabelecerá comunicação com a estação transmissora, independente do endereço nesta contido, e propagará a mensagem para o resto da rede, se seu endereço não for o destino final da mensagem.

Para as mensagens com endereços definidos, apenas a estação que tem seu endereço explicitamente escrito no campo nó destino é que estabelecerá uma comunicação respondendo o RTS (nó de origem, nó de destino), com um CTS (nó de origem, nó de destino), as demais estações simplesmente descartarão a mensagem recebida e permaneceram na escuta.

No caso de mensagens por inundação, a estação ao emitir um RbRTS(nó de origem, nó de destino) só responderá ao primeiro CbCTS(nó de origem, nó intermediário, Nó de destino), no caso de receber mais de um, as demais estações que ainda não responderam, ao ouvirem um CbCTS(nó de origem nó intermediário, nó de destino) desistirão de fazê-lo.

3.5. Especificação Formal e Verificação do Protocolo DSR

3.5.1. Protocolo DSR

O DSR foi descrito em LOTOS, com vários usuários e apresentando alternativas quanto ao número de nós da rede e quanto à utilização ou não de temporizadores. Mostra-se, como exemplo, a arquitetura da descrição para 2 usuários e 5 nós, lembrando que os temporizadores só são inseridos no corpo do programa, sendo mostrados no capítulo seguinte.

```
specification DSR[ACC, DEL, ... , dsr_conf] : noexit]

library    Boolean, NaturalNumber, DSRLIB     endlib

behaviour

  hide phy_req1, ... , dsr_ind, dsr_resp in
    ((User1 [ACC, dsr_req, dsr_conf]
        |||
        User2 [dsr_ind, DEL, dsr_resp])
      |[dsr_req, dsr_conf, dsr_ind, dsr_resp]|
      (DSR1 [dsr_req, ... , dsr_conf]
        |||
        DSR2 [phy_ind21, ... , rRpC]
        |||
        DSR3 [phy_req44, ... , rRpCA]
        |||
        DSR4 [phy_ind25, ... , rRpC]
        |||
        DSR5 [phy_ind51, ... , rRpD2C])
        |[phy_req1, ... , rRpCA]|
```

Medium [phy_req1, ... , rRpCA])

where

process USER1[ACC,dsr_dreq ...

Então, utilizando os compiladores Caesar, foram gerados os sistemas de transições rotuladas, tanto para o protocolo como para o serviço prestado pela camada.

3.5.2 Descrição do Serviço DSR

```
specification DSR_Service [ACC, DEL] : noexit
  library DSRLIB endlib
behaviour
  Service [ACC, DEL]
where
  process Service [ACC, DEL] : noexit :=
    ACC ?Data:data_type;
      DEL !Data;
        Service [ACC, DEL]
  endproc
endspec
```

3.5.3. Consolidação dos Resultados das Simulações para o Protocolo DSR de uma Forma Geral

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

Estes testes e procedimentos foram feitos em todas as fases do projeto do protocolo DSR. Em cada experimento, em cada incremento de complexidade do protocolo, foram testadas todas as propriedades e condições anteriormente mencionadas, desta forma o projeto do protocolo crescia em complexidade apenas a partir destas condições satisfeitas.

Com o aumento da complexidade, algumas modificações e observações foram feitas para o protocolo e estas estão descritas no item a seguir.

3.5.4. Modificações no Protocolo de Roteamento DSR

Em primeiro lugar, foram utilizados apenas documentos oficiais do IETF para especificação do protocolo em LOTOS;

Foram feitas especificações das partes mais essenciais do protocolo, em várias fases, em linguagem para especificação formal, LOTOS, para verificação de consistência, convergência e estruturação do protocolo;

Todas as estruturas que não estavam formalmente definidas nos documentos IETF foram aqui definidas para os devidos testes de simulação com as ferramentas do CADP;

Todos os testes com o protocolo, escrito em LOTOS, foram feitos em ambiente de simulação cujas condicionantes do meio, que seria a propagação em espaço livre, eram mais rigorosos que num caso real, como por exemplo, o fato de todas as estações terem probabilidades iguais de receberem uma mensagem, inclusive a própria emissora, caso muito mais rigoroso que o real. Estas hipóteses forçaram a escrita do protocolo com um maior grau de detalhamento para o tratamento destas possibilidades;

Tendo como referência o *draft* do IETF, para o DSR foram definidas estruturas de teste para a tomada de decisão, para convergência da troca de mensagens das diversas fases de funcionamento do protocolo, sejam elas: busca de rotas, o *Route Request*, encaminhamento da busca, resposta da busca, o *Route Reply*, encaminhamento da resposta, resposta de rotas, estações inexistentes, o *Route Error*, encaminhamento de dados, resposta e encaminhamentos de *ACKs*, etc.

Como efetivo trabalho foram criadas várias estruturas de decisão para tratamento de todas as hipóteses tratadas nos documentos do IETF, além daquelas impostas pela linguagem LOTOS, o que possibilitou a constatação da convergência do protocolo e da conclusão de que o campo de identificação, ID, não é realmente necessário na busca e descoberta de rotas, ou qualquer outra fase do protocolo. Isto porque em LOTOS não foi necessário nenhum teste deste campo em nenhuma das estruturas do mesmo. Assim se conclui que o ID só será necessário para as estações que habilitarem as rotas nos seus *Route Caches*, visto que o ID será utilizado para a indexação destas rotas nestes *Route Caches*. Particularmente para armazenamento de rotas redundantes que serão apenas diferenciadas pelo seu ID.

Esta abordagem quanto ao ID fará com que o processamento na troca de mensagens do protocolo diminua, em alguns casos em até um terço, deixando esta parte do processamento para quando a estação estiver armazenando o vetor de rotas em seu *Route Cache*.

3.6. Especificação e Verificação do Protocolo LANMAR

3.6.1. Protocolo LANMAR

O protocolo LANMAR foi descrito em LOTOS, com vários usuários e apresentando alternativas quanto ao número de nós e de células na rede, além da utilização ou não de temporizadores. Mostra-se, como exemplo, uma arquitetura descrita para 4 nós, cada qual com seu usuário, lembrando que os temporizadores são inseridos apenas no corpo do programa, sendo mostrados no capítulo seguinte.

```
specification LANMAR [ACC, phy_req3,  ..., sincro, sinc4, DEL] : noexit

library

        Boolean, NaturalNumber, LANMARB

endlib

behaviour

(*  hide  phy_req3,  ..., LANMAR_resp1, sincro, sinc4 in *)

(((sincronizador [ACC, sinc1, sinc2, sinc3, sinc4, jaera]
        |[sinc1, sinc2, sinc3, sinc4, jaera]|
        (NoDoD1[ACC, phy_req3,  ... , DEL ](ACP of DATA_TYPE, ...)
        |||
        NoDoD2[ACC, phy_req3,  ... , LANMAR_resp1, sincro, DEL]
        |||
        NoDoA1[ACC, ... , sincro, DEL ](ACP of DATA_TYPE, ...)
        |||
        NoDoA2[ACC, phy_req3,  ..., LANMAR_resp1, sincro, DEL]
        |||
        NoDoLAD1[ACC, ... , sincro, DEL ](ACP of DATA_TYPE, ...)
        |||
        NoDoLAD2[ACC, phy_req3,  ..., LANMAR_resp1, sincro, DEL]
        |||
        NoDoLA1[ACC, ... , sincro, DEL ](ACP of DATA_TYPE, ...)
        |||
        NoDoLA2[ACC, phy_req3,  ..., LANMAR_resp1, sincro, DEL]
        |||
        LANMAR1[ACC, phy_req3,  ..., sincro, DEL](A1 OF ID_TYPE)
        |||
```

```
        LANMAR11[ACC, phy_req3,  ..., LANMAR_resp1, sincro, DEL]
        |||
        LANMAR2[ACC, phy_req3,  ...,  sincro, DEL](A1 OF ID_TYPE)
        |||
        LANMAR21[ACC, phy_req3,  ..., LANMAR_resp1, sincro, DEL]))
        |[ phy_req3,  ..., LANMAR_resp1, sincro]|
        MEDIUM2 [phy_req3,  ..., LANMAR_resp1, sincro]))
    where
            process NoDoD1[ACC, phy_req3,  ...
```

Então, utilizando os compiladores Caesar, foram gerados os sistemas de transições rotuladas tanto para o protocolo como para o serviço prestado pela camada.

3.6.2. Descrição do Serviço LANMAR

```
specification LANMAR_Service [ACC, DEL] : noexit
  library LANMARLIB endlib
behaviour
  Service [ACC, DEL]
where
  process Service [ACC, DEL] : noexit :=
    ACC ?Data:data_type;
      DEL !Data;
        Service [ACC, DEL]
  endproc
endspec
```

3.6.3. Consolidação dos Resultados das Simulações para o Protocolo LANMAR de uma Forma Geral

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

Estes testes e procedimentos foram feitos em todas as fases do projeto do protocolo LANMAR. Em cada experimento, além de incremento em complexidade, foram testadas todas as propriedades e condições anteriormente mencionadas, desta forma o projeto crescia em detalhes, possibilidades, apenas a partir de tais condições satisfeitas.

Com o aumento da complexidade, algumas modificações e observações foram necessárias para um melhor desempenho do protocolo e estas estão descritas no item seguinte.

3.6.4. Modificações no protocolo de Roteamento LANMAR

A ressalva principal deste protocolo se refere ao "draft" do IETF para o protocolo LANMAR, este define apenas as características gerais para a estruturação de uma rede em células, com seus respectivos líderes. Define que dentro das células, para as estações nelas residentes, deveriam ser instalados protocolos pró-ativos modificados, tal como o FSR, ou qualquer um dos protocolos pró-ativos especificados nos *drafts* do IETF, reduzindo seus parâmetros para que este se adeqüe ao tamanho da célula.

Para a troca de mensagens entre estações fora de uma mesma célula, o *draft* sugere a utilização de outros protocolos pró-ativos, como o TBRF ou o OLSR, modificado, para trabalhar na difusão de rotas entre líderes de células.

O *draft* não define exatamente como será interligação destes protocolos, apenas sugere algumas possibilidades para possíveis combinações destes.

Nesta pesquisa, através da especificação formal, utilizando a linguagem LOTOS, foi possível definir os pontos de instalação e interligação dos diversos protocolos, como foi mostrado pela figura. 03.

Assim, a partir de premissas básicas definidas no *draft* IETF do LANMAR foi feito todo um desenvolvimento deste protocolo e daqueles que o compõe. Além disso, foram feitas análises destes protocolos e a troca de um deles por um outro já simulado, testado e modificado anteriormente.

A estrutura de formação da rede, utilizando o protocolo LANMAR, foi aqui formalmente especificada e o protocolo DSR foi utilizado para fazer a descoberta e manutenção de rotas entre células, o que transformou o LANMAR para um protocolo híbrido, pró-ativo dentro da célula e reativo entre células.

Isto gerou um ganho em escalabilidade, já que a inserção de um protocolo reativo entre células permite a entrada de novas estações e células na rede, sem a necessidade de informação imediata aos demais nós líderes. Através dos mecanismos de funcionamento do DSR, as atualizações dos endereços de novos elementos na rede serão feitos no decorrer das buscas de rotas que ocorrem sob demanda.

3.7. Comentários

Este capítulo explicou como as especificações formais dos protocolos e de seus respectivos serviços foram submetidas a testes das propriedades como, equivalência observacional, equivalência forte, se eram vivos e reinicializáveis, quanto da possibilidade de haver *deadlocks*, etc. Todas condições essenciais de funcionamento do protocolo que habilitaram o prosseguimento dos experimentos.

Estes testes e procedimentos foram feitos em todas as fases do projeto dos protocolos. Em cada experimento, onde havia um incremento de complexidade, foram testadas todas as propriedades e condições anteriormente mencionadas, de forma que o projeto do protocolo crescia em complexidade apenas a partir de tais condições satisfeitas.

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.

Assim, foram inseridos, progressivamente, em cada versão verificada, elementos de complexidade do protocolo - temporizadores, passagem de parâmetros e tratamento de exceções (nós fora de alcance, laços nas tabelas de roteamento, erros de transmissão) - até a apresentação de um modelo completo de estação e de rede nos últimos modelos das simulações. Com isto, esta série de experimentos culminou com a especificação e verificação de modelos completos de nós, que apresentam transparência em relação ao meio, independência diante dos demais nós da rede, sendo dotados de todas as funcionalidades de transmissão e recepção requeridas para uma estação real em redes *ad-hoc*.

Todas estas ressalvas e conclusões estão mais bem detalhadas no próximo capítulo referente aos resultados obtidos.

4. Resultados

4.1. Introdução

Neste capítulo serão descritos todos os experimentos feitos com os protocolos MACAW, DSR e LANMAR.

De início, os protocolos são descritos por partes, de maneira sucinta e genérica, sem muita profundidade para cada característica de funcionamento. Com o decorrer do experimento partes mais complexas e detalhes são inseridos de forma incremental, até a representação das partes essenciais, senão de todo o protocolo.

A complexidade acima mencionada se refere aos mecanismos de troca de mensagens, das descrições dos nós e da quantidade destes na rede. As estações se tornam cada vez mais detalhadas, agregando cada vez mais complexidade em sua construção. Da mesma forma, a rede aumenta sua quantidade de nós e células, se for o caso.

Toda esta forma de abordagem incremental tem o intuito de, ao final, os experimentos culminarem com uma descrição completa de cada nó na rede e da rede como um grande sistema.

Neste capítulo, o item 4.2. apresenta o experimento com o protocolo de acesso ao meio MACAW; o item 4.3 apresenta o experimento com o protocolo de roteamento DSR; o item 4.4 apresenta o experimento com o protocolo de roteamento LANMAR e o item 4.5 comentários finais sobre os aspectos de interesse dos experimentos.

4.2. O Protocolo de Acesso ao Meio MACAW

A seqüência de simulação segue uma lógica incremental de complexidade, primeiro apresenta a descrição do protocolo como este foi concebido na sua idéia original [12], ver figura 04, em seguida foram sendo implementadas novas modificações para um modelo que atendesse a idéia da rede proposta, ou seja, *ad-hoc*.

Posteriormente uma tabela resumirá toda a evolução do experimento com o protocolo MACAW, o qual terá a evolução explicada, seqüencialmente, no decorrer do texto.

É necessário explicar que todas as fases de evolução da simulação do protocolo MACAW foram descritas com a inclusão de passagem de parâmetros e processos de temporização não foram aqui simulados. Suas conclusões serão mencionadas no capítulo sobre simulações com o protocolo DSR.

O protocolo DSR foi, cronologicamente, o primeiro a ser simulado, por isso as fases de sua simulação estão descritas com mais detalhes no item 4.3, contemplando as conclusões e diferenças entre simulações, utilizando apenas processos simples e/ou que levam em consideração o tempo e/ou passagem de parâmetros, etc.

As conclusões quanto à diferenciação das mensagens e dos campos necessários para uma rápida e eficiente comunicação foram obtidas através da ferramenta de simulação CADP, para uma rede cujos nós tem todas as funcionalidades tanto para a transmissão quanto para a recepção de mensagens. Isto porque a linguagem LOTOS possibilita a simulação da comunicação entre vários nós ao mesmo tempo, como num caso real, onde estações em extremidades opostas de uma célula podem se comunicar desde que o canal rádio esteja livre.

A linguagem LOTOS funciona como uma máquina seqüencial, permitindo a troca de mensagens entre estações e através de um meio de comunicação, desde que estes estejam livres para estabelecer esta conexão. Existe assim, uma simulação do protocolo num caso real, onde estações concorrem para obtenção do meio para a comunicação.

Como foi dito anteriormente, a primeira simulação seguiu exatamente a idéia inicial do protocolo [12], sua rede de *Petri* é mostrada a seguir pela figura 04.

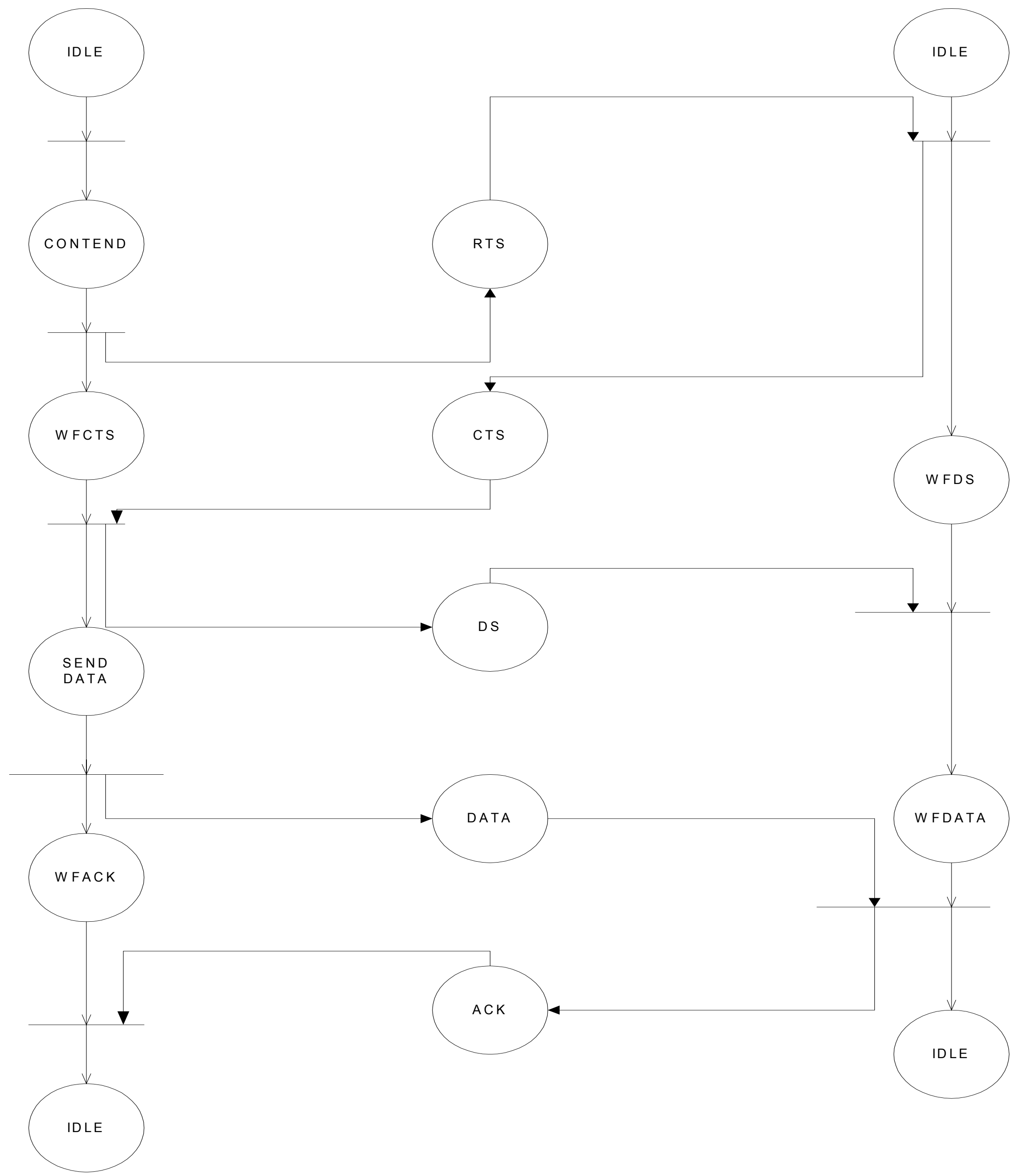

Figura 04 - Rede de *Petri* do Protocolo Macaw Original

4.2.1. Resultado das Simulações do Protocolo MACAW

4.2.1.1. Simulação com o Protocolo MACAW Original

A primeira simulação feita com o MACAW utilizou sua estrutura original, tal qual foi descrita no artigo que o apresentou a comunidade científica [12]. A ilustração em rede de *Petri* do funcionamento deste protocolo foi mostrada na figura 04.

Esta simulação foi feita com uma estação transmissora e uma receptora. A partir de estudos e desta simulação foram constatadas a viabilidade do protocolo MACAW e a necessidade de algumas modificações do mesmo, para suprir a comunicação de várias estações que apresentam comportamento independente e aleatório através da rede.

A figura 05 mostra os vários processos, com suas respectivas trocas de mensagens, ilustrando o protocolo que foi descrito no segundo capítulo e conforme a rede de *Petri* apresentada anteriormente. Observe que são mostradas, sucintamente, apenas as mensagens trocadas entre uma estação transmissora e outra receptora, sempre através do processo MEDIUM, que representa o meio físico.

Os processos User1 e User2, representam os usuários das camadas superiores, de roteamento, que recebem os serviços da camada inferior, através do protocolo MACAW, MACAW1 e MACAW2, inteiramente confiável.

MACAW1 e MACAW2 por sua vez, recebem os serviços da camada inferior através do processo MEDIUM, que neste caso é a representação do meio físico, “espaço livre”. A figura 05 ilustra a descrição anterior, a interação entre processos e a trocas de mensagens como observado na rede de *Petri* da figura 04.

O protocolo é descrito em LOTOS da seguinte forma:

-ao receberem uma mensagem de uma camada acima da sua, os processos USER1 solicita os serviços da camada MACAW;

-os processos MACAW1 e MACAW2 representam o protocolo MACAW dentro de cada estação, funcionando de forma seqüencial e sincronizada;

-os processos internos a MACAW1 e MACAW2 são encadeados e se acionam seqüencialmente;

-quando MACAW2 recebe DATA, ele envia a USER2, que confirma o recebimento e envia ACK para a outra estação;

-a comunicação entre estações se encerra, MACAW2 volta a seu estado inicial.

MACAW1, após confirmar o êxito da comunicação com USER2, também volta a seu estado inicial e o canal está livre para uma nova comunicação.

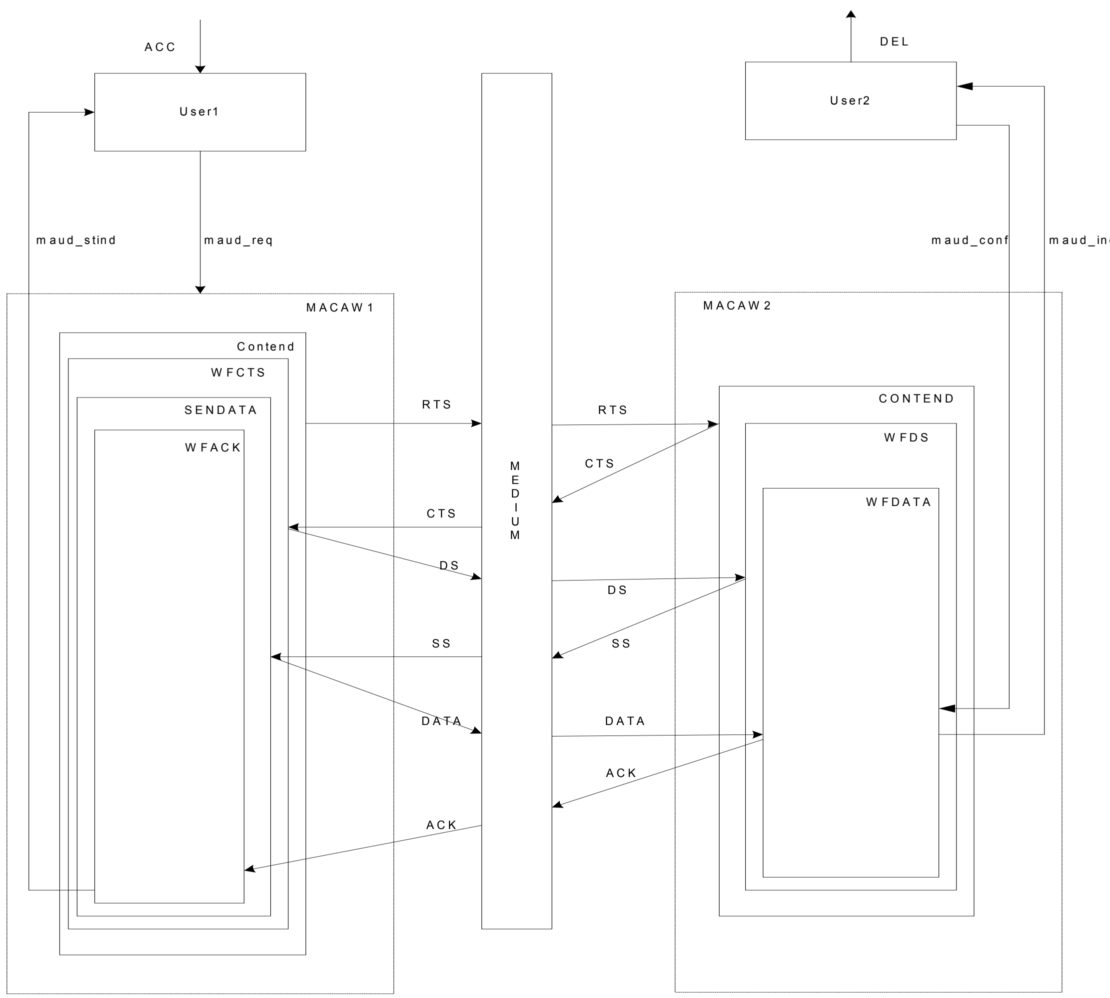

Figura 05 - Diagrama de Processos da Versão Original do Protocolo MACAW

O gráfico de estados e transições da figura 06 foi o primeiro obtido com as ferramentas do CADP, a partir da descrição formal, em LOTOS, do protocolo de acesso ao meio MACAW. Este gráfico representa o número de estados e transições minimizado para o protocolo, isto porque o gráfico de estados e transições sem a utilização da ferramenta de minimização gerou 139 estados e 145 transições, um número excessivamente grande para ser visualizado e analisado. Com a propriedade de minimização do CADP o número de estados e transições foi reduzido para menos de um terço, 44 estados e 46 transições, o que possibilitou uma análise mais qualitativa dos resultados obtidos.

A propriedade de minimização descarta todos os estados e transições redundantes obtidos através da simulação.

Entendem-se como redundantes aqueles estados e transições que possuem o mesmo comportamento, mesma origem e mesmo destino.

A figura 06 mostra a coerência da descrição formal e a confirmação das propriedades descritas no item 3.6.1.1.

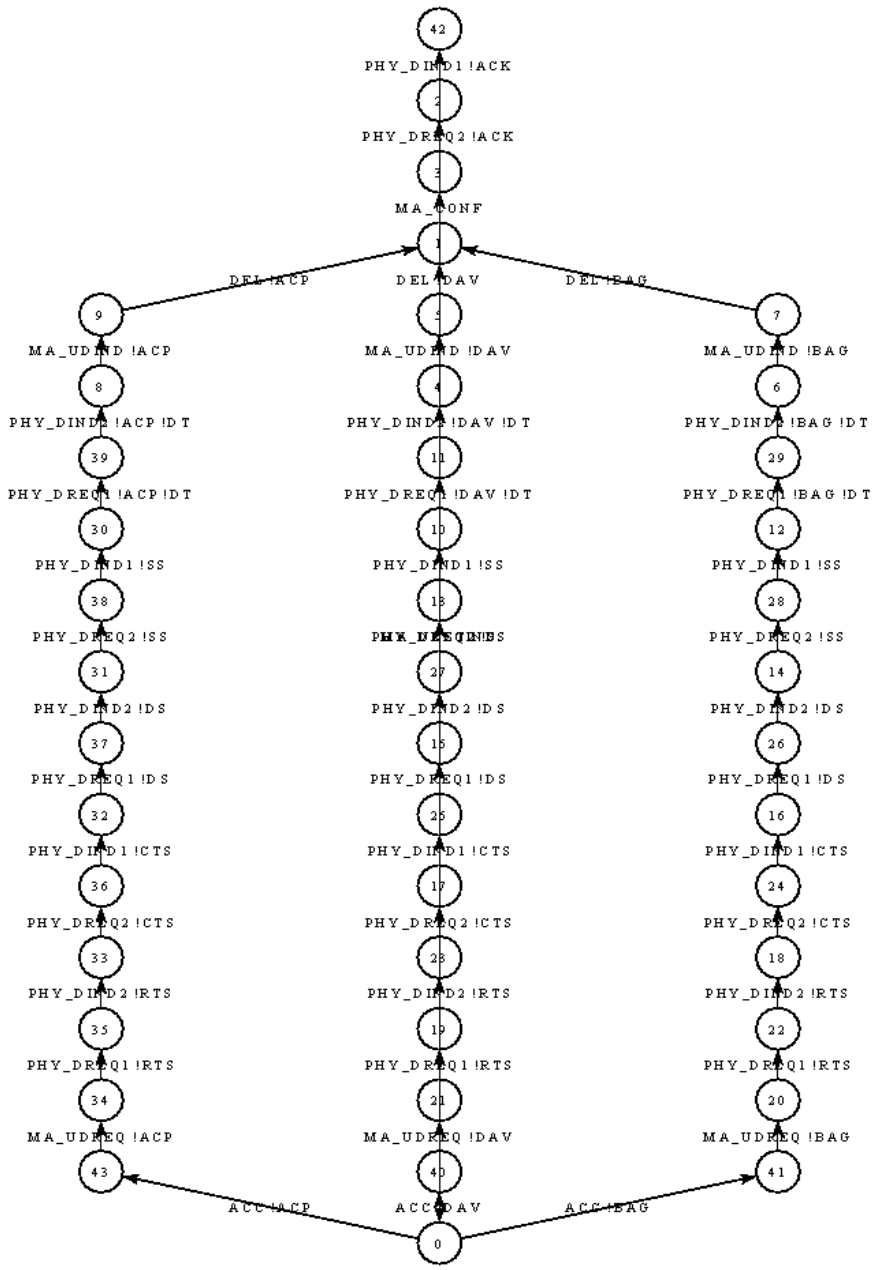

Figura 06 - Gráfico de Estados e Transições do Protocolo MACAW Original

4.2.1.2. Simulações Intermediárias

Foram feitas diversas simulações e destas serão relatadas algumas, para ilustrar o andamento do experimento:

Uma segunda simulação foi feita com um nó emissor e um receptor, porém com uma gama muito maior de mensagens, isto para verificar a diversidade de trocas de mensagens do protocolo e se havia algum problema para "N" trocas distintas entre duas estações.

A simulação seguinte foi feita com algumas das modificações mencionadas no item 3.6.1.2. Apesar da troca de mensagens ainda ser entre duas estações, mantendo a idéia de uma receptora e outra transmissora, já foi inserido a modificação entre mensagens por inundação e com endereços definidos. Assim, o nó ao receber as mensagens RbRTS(nó de origem, nó de destino), ou RTS(nó de origem, nó de destino), já detectará o tipo de mensagem e saberá se deve ou não responder com CbCTS(nó de origem, nó intermediário, nó de destino), ou CTS(nó de origem, nó de destino) e posteriormente esperará o restante dos dados para enviar a camada superior, de roteamento, se for o caso.

Uma quarta simulação foi construída com as modificações mencionadas na simulação anterior, porém com a troca de mensagens entre três estações, mantendo a idéia de uma transmissora e duas receptoras. Com esta configuração pode haver algumas possibilidades, haja vista que toda mensagem por inundação que não for recebida pelo seu nó destino imediato deverá ser, após ser analisada pela camada superior, reencaminhada ao seu verdadeiro destino, também por inundação, e isto permite duas possibilidades para a confirmação da recepção da mensagem:

-pode ser disparado um AbACK(nó de origem, nó de destino) nó a nó, entre nós adjacentes, para confirmação da recepção da mensagem;

-será dado um AbACK(nó de origem, nó de destino) apenas quando o nó destino receber a mensagem, fazendo com que todos os nós intermediários envolvidos na comunicação fiquem esperando a resposta através do AbACK(nó de origem, nó de destino) para reencaminhá-lo ao iniciador da comunicação. Isto pode ser necessário para redes densamente povoadas, com grande disputa pela obtenção do canal de comunicação, podendo gerar rotas com múltiplos saltos desnecessários e até laços nas rotas. A figura 07 mostra um resumo das trocas destas mensagens e dos processos envolvidos nesta simulação para três estações.

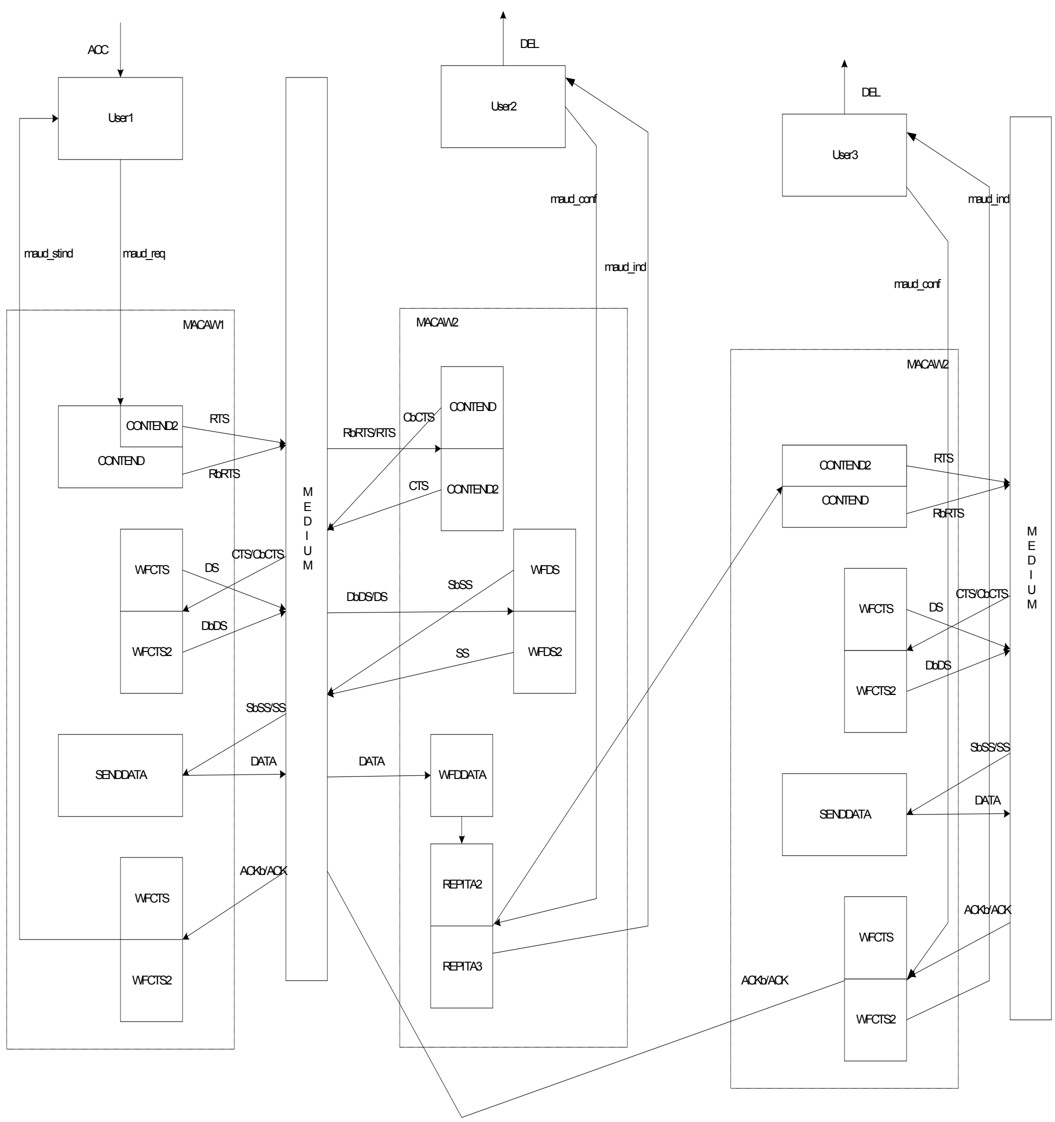

Figura 07 - Diagrama de Processos da Quarta Versão do Protocolo, Incluindo Retransmissão entre Estações

Na simulação posterior foram feitos testes de lógica inter processos, para mensagens cujo destinatário se encontra explícito no corpo do RTS (Nó de origem, Nó de destino). Esta necessidade pode não ser justificada já que a comunicação entre estes nós já foi estabelecida anteriormente, por estruturas de decisão do encaminhador da mensagem, mesmo assim esta possibilidade foi testada para que não houvesse falhas como, por exemplo, “deadlocks”.

Houve outras simulações cujos resultados gerou contribuições pouco significativas para o experimento e por isso não foram citadas.

É importante lembrar, que a evolução incremental em complexidade de cada simulação só era feita, a partir do teste e confirmação de todas as propriedades descritas no item 3.6.1.1..

4.2.1.3 Duas Estações Transmissoras e Receptoras Simultaneamente com Processo de Sincronização

Nesta simulação, todas as modificações descritas anteriormente foram inseridas em cada estação. Deste ponto em diante os nós têm todas as funcionalidades de transmissão e recepção, como num caso real de uma rede *ad-hoc*.

Como conseqüência deste tipo de arquitetura, a de se salientar que:

-para duas estações o protocolo convergiu com um número demasiadamente grande de estados, 550401, e transições, 2160092;

-mesmo após aplicação da minimização, a visualização gráfica não foi possível, o número de estados e transições (8670 e 30736, respectivamente) ainda estavam demasiadamente grandes para uma boa apresentação visual.

Estes valores são razoáveis já que a linguagem LOTOS satura todas as possibilidades de troca de mensagens pelas estações, o que é estritamente necessário para uma boa avaliação, como num caso real.

A grande quantidade de estados e transições obtidos para o protocolo e sua tendência de crescimento exponencial para cada nova modificação, por menor que ela seja, se deve a estrutura de funcionamento do LOTOS. Sua lógica de funcionamento testa todas as possibilidades de trocas de mensagens, por mais remotas que sejam; esta é a característica que traz total segurança para testes e simulações de protocolos, porém é a mesma peculiaridade que impossibilita o teste com muitos nós na rede.

Como ilustração do que foi mencionado, a figura 08 mostra as estruturas de trocas de mensagens e de decisão para a recepção e a transmissão dentro de uma estação. As diversas possibilidades de dados contidos em ***phy_req1*** podem ser combinadas com as diversas de ***phy_ind1***, em diversas fases da troca de mensagens, em diversos pontos do protocolo e em diversas estruturas para cada estação, como por exemplo, os pares abaixo:

phy_req1 !RbRTS !NNO !NOB; *phy_ind1 !RbRTS !NNO !NOB;*
phy_req1!AbACK *!TIPO!NNOINT1;*
phy_ind1!AbACK!TIPO!NNOINT1;
phy_req1 !DSb !NNO !NOB !TIPO !NNOINT1;
phy_ind1 !DSb !NNO !NOB !TIPO !NNOINT1;

Como a obtenção do gráfico minimizado de estados e transições da simulação anterior não permitiu uma visualização nítida para avaliação dos resultados, foi feita uma simulação complementar com as mesmas características, porém com um elemento de sincronização entre as duas estações e através do meio, possibilitando a observação do efeito da troca de mensagens de estação para estação.

Quando o elemento de sincronismo foi inserido o número de estados diminuiu para 82775 e o de transições para 102853, além disto a minimização obteve êxito, pois o número de estados diminuiu para 81 e o de transições para 90. A figura 09 mostra estes resultados, para uma rede cuja arquitetura das estações foi descrita pela figura 08.

A metodologia descrita foi utilizada apenas para facilitar o andamento do experimento e a visualização gráfica dos resultados, o processo de sincronização é dispensável para o funcionamento num caso real.

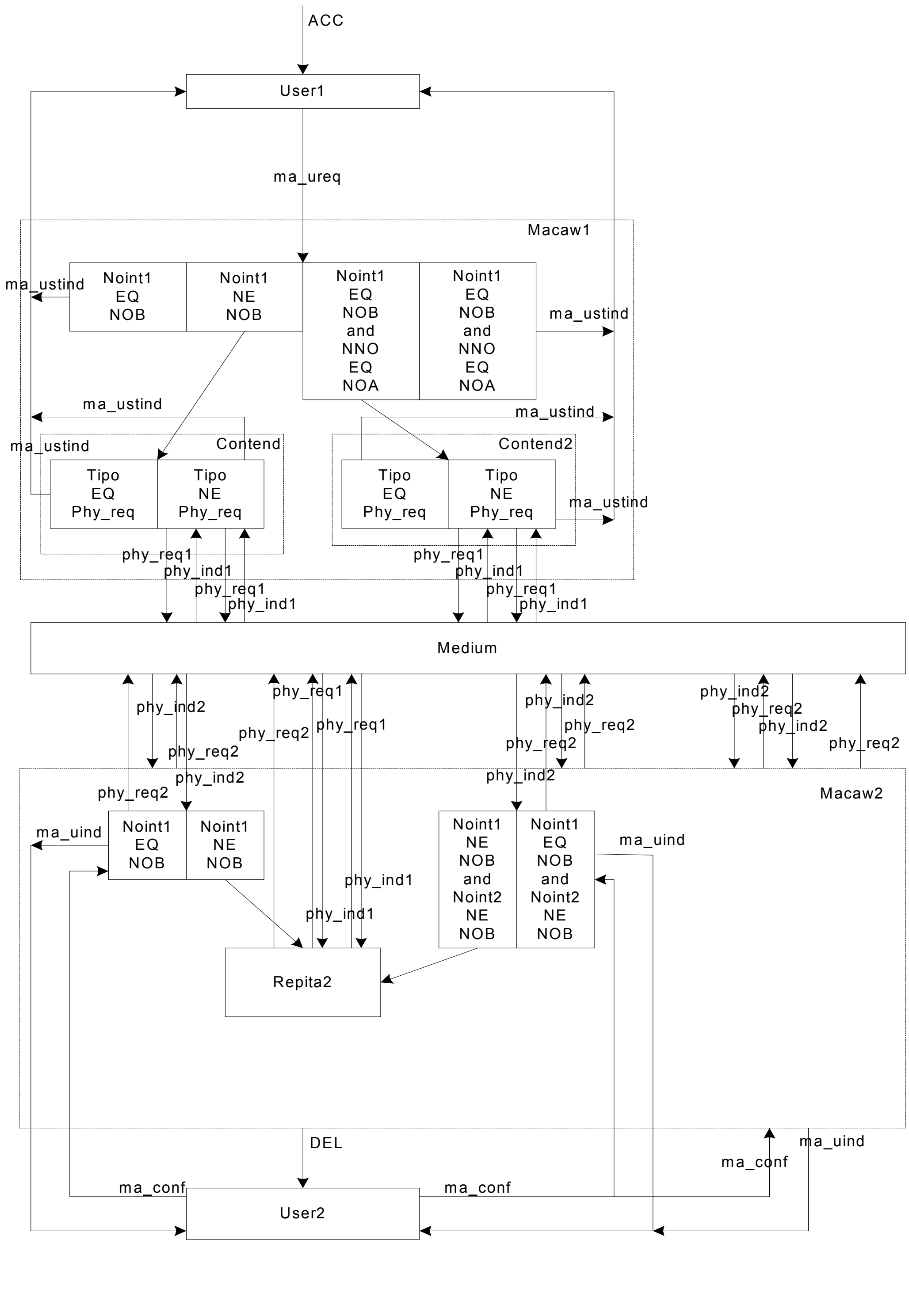

Figura 08 – Estruturas de Decisão e de Trocas de Mensagens Dentro de Cada Estação

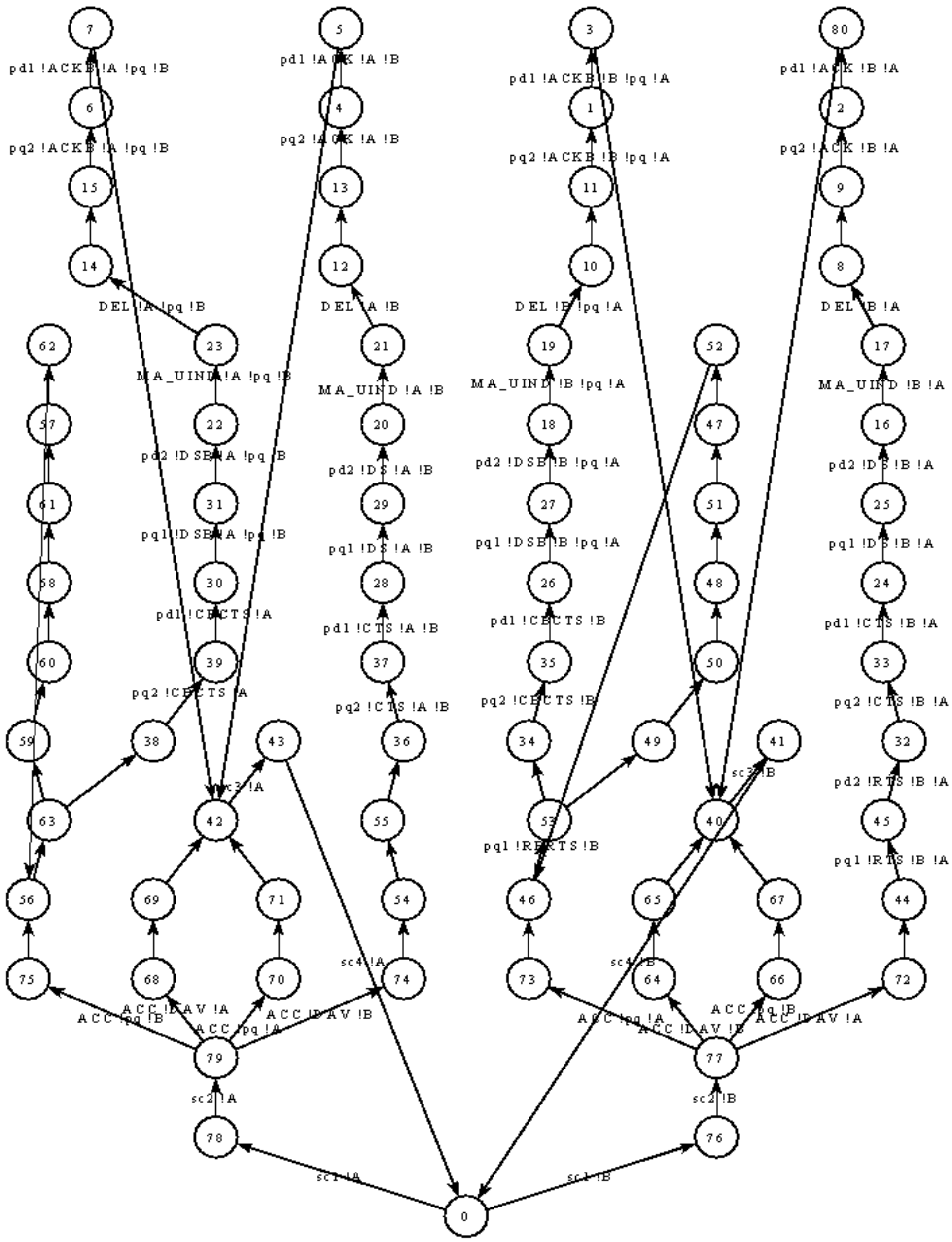

Figura 09 - Gráfico de Estados e Transições do Protocolo MACAW Modificado

4.2.2. Resumo das Etapas dos Experimentos

A primeira simulação foi feita com a estrutura original do protocolo, com uma estação apenas transmissora e uma outra apenas receptora; a segunda simulação foi feita com um nó emissor e um receptor, porém com uma gama muito maior de mensagens; na terceira simulação, apesar da troca de mensagens ainda ser entre duas estações, mantendo a idéia de uma receptora e outra transmissora, já foi inserido as modificações entre mensagens por inundação e com endereços definidos; a quarta simulação foi feita com as modificações mencionadas na simulação anterior e mais uma terceira estação; na quinta simulação foram feitos testes de lógica inter processos, para mensagens cujo destinatário se encontra explícito no corpo das mensagens; na sexta simulação todas as modificações mencionadas nas simulações anteriores foram inseridas em cada estação, deste ponto em diante todos os nós têm todas as funcionalidades de transmissão e recepção; foi feita uma simulação complementar com as mesmas características da simulação anterior, porém com um elemento de sincronização entre as duas estações; as simulações só prosseguiam quando as propriedades descritas no item 3.6.1.1.eram testadas e comprovadas .

Houve ainda uma última simulação onde foram feitos experimentos com três estações completas, com todas as funcionalidades de transmissão e recepção como descrito no experimento 4.2.1.3.. As estações tiveram todas suas funcionalidades testadas duas a duas, e sempre convergiram com números de estados e transições mencionados no experimento anterior. Quando o CADP tentou trabalhar com as com três estações recebendo e transmitindo mensagens simultaneamente, a quantidade de estados e transições ultrapassa a capacidade de manipulação e análise do software, que permite simulações que gerem um número máximo de estados e transições da ordem de dezenas de milhões.

Já foi mostrada, anteriormente, a grande quantidade de estados e transições obtidas para o protocolo e sua tendência de crescimento exponencial para cada nova modificação do mesmo, por menor que ela seja. Isto se deve a estrutura de funcionamento do LOTOS, que testa todas as possibilidades de trocas de mensagens, por mais remotas que sejam. Além disto, é necessário à inclusão de um processo de *time out* para cada envio de mensagem sem sucesso. Assim, para um número grande de estações na rede haverá uma infinidade de possibilidades de combinações, ou seja, uma quantidade extremamente grande de estados e transições.

A seguir é mostrada uma tabela com o resumo das etapas seguidas neste experimento, para o protocolo MACAW.

Como dito anteriormente, a complexidade do protocolo, bem como das estações da rede, cresce à medida que os experimentos vão evoluindo.

Deve-se salientar que a complexidade maior está quando todas estações da rede têm totais possibilidades e propriedades para transmissão e recepção, a partir do sexto experimento, item 11 da tabela 04.

Segue uma sucinta descrição desta tabela para o melhor entendimento do leitor:

-**Número de nós da rede** se refere ao número total de nós desta, quer sejam emissores, ou receptores ou ambos;

-**Quantidade de nós emissores** contabiliza aqueles que tem como propriedade iniciar uma troca de mensagens;

-**Quantidade de nós receptores** contabiliza aqueles que respondem a uma solicitação de troca de mensagens;

-**Número de estados** faz menção ao número de estados gerado pela ferramenta responsável pela obtenção do grafo LTS;

-**Número de transições** faz menção ao número de transições gerado pela ferramenta responsável pela obtenção do grafo LTS;

-O valor **minimizado** indica que o grafo LTS passou por um filtro que retirou seus estados e transições redundantes.

Item	Característica do Protocolo	Número de nós da rede	Quantidade de nós emissores	Quantidade de nós receptores	Número de estados	Número de transições	Minimi-zado
01	protocolo	02	01	01	139	145	Não
02	protocolo	02	01	01	44	46	Sim
03	protocolo	02	01	01	19081	22929	Não
04	protocolo	02	01	01	383	409	Sim
05	protocolo	02	01	01	201	267	Não
06	protocolo	02	01	01	31	34	Sim
07	protocolo	03	01	02	12045	15445	Não
08	protocolo	03	01	02	87	94	Sim
09	protocolo	03	01	02	235843	353991	Não
10	protocolo	03	01	02	123	134	Sim
11	protocolo	02	02	02	550401	2160092	Não
12	protocolo	02	02	02	8670	30736	Sim
13	protocolo	02	02	02	82775	102853	Não
14	protocolo	02	02	02	81	90	Sim
15	protocolo	03	03	03	(1)	(1)	Não
16	protocolo	03	03	03	(1)	(1)	Sim
17	Serviço	02	X	X	6	12	Não

Tabela 03 - Resumo das Simulações do Protocolo MACAW

(1) Informa que estourou a capacidade de trabalho do software CADP.

4.2.3. Conclusões e Modificações sobre o Experimento do Protocolo MACAW

Como uma conclusão final sobre as modificações que foram feitas no protocolo MACAW, para dar suporte aos protocolos de roteamento para redes *ad-hoc*, deve-se citar:

-a inclusão de estruturas para a troca de mensagens por inundação juntamente com as de endereços definidos. Sendo possível a detecção direta do tipo de mensagem, quando qualquer estação receber as mensagens RbRTS(nó de origem, nó de destino), ou RTS(nó de origem, nó de destino), saber se devem ou não responder com CbCTS(nó de origem, nó de destino), ou CTS(nó de origem, nó de destino) e posteriormente esperar o restante dos dados para enviar a camada superior, de roteamento. Desta forma, toda mensagem por inundação que não for recebida pelo seu nó destino imediato deverá, após ser analisada pela camada superior, ser reencaminhada ao seu verdadeiro destino também por inundação e isto permite duas possibilidades para a confirmação da recepção da mensagem:

-pode ser disparado um AbACK(nó de origem, nó de destino) nó a nó, entre nós adjacentes, para confirmação da recepção da mensagem;

-será dado um AbACK(nó de origem, nó de destino) apenas quando o nó destino receber a mensagem, fazendo com que todos os nós intermediários envolvidos na comunicação fiquem esperando a resposta através do AbACK(nó de origem, nó de destino) para reencaminhá-lo ao iniciador da comunicação. Isto pode ser necessário para redes densamente povoadas, onde uma grande disputa para obtenção do canal de comunicação pode gerar rotas com múltiplos saltos desnecessários e até laços nas rotas.

4.3. O Protocolo de Roteamento DSR

O protocolo DSR foi descrito em LOTOS, com vários usuários, apresentando alternativas quanto ao número de nós da rede, quanto à utilização ou não de temporizadores e passagem de parâmetros.

As descrições das fases de projeto foram executadas numa seqüência lógica e cronológica dentro da filosofia de evolução incremental de complexidade do protocolo.

Uma observação importante é que a simulação com dois nós, apesar de parecer desnecessário para o teste do protocolo, se torna importante na medida em que promove a correção da escrita do mesmo. Mais adiante será mostrado que, sem a utilização de temporizadores, o aumento dos nós gera um aumento suave de estados e transições, porém quando se acrescentam estes mesmos temporizadores o número de estados, com o aumento do número de nós, tende a infinito; isto se deve às estruturas de funcionamento do LOTOS, que necessitam de temporizadores sincronizados com as perdas do meio.

4.3.1. Simulações do Protocolo Roteamento DSR

4.3.1.1. Simulação com Duas Estações

Na primeira simulação do DSR, cuja primeira versão possui apenas as funcionalidades básicas do protocolo, são apresentados, sucintamente, os mecanismos de busca e encaminhamento de rotas, *Route Request* e *Route Reply*, através dos processos ROTREQ e ROTREP, respectivamente.

A figura 10 mostra as interações com dois usuários: USER1 e USER2 entre os protocolos DSR1 (descrito apenas como emissor, através do processo ROTREQ) e DSR2 (descrito apenas como receptor, através do processo ROTREP), trocando mensagens através do protocolo MACAW.

Os protocolos interagem entre si, considerando que o protocolo MACAW lhes dá total suporte para o acesso ao meio.

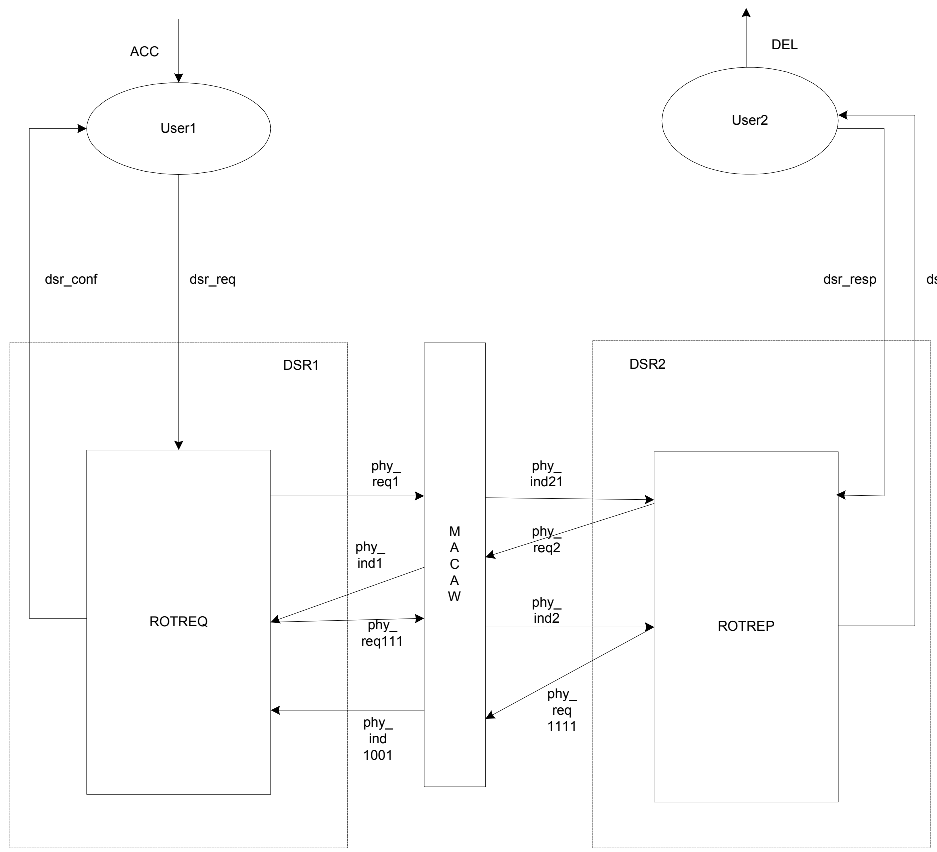

Figura 10 - Primeira Implementação com Dois Usuários e Dois Processos Simples

Nesta fase inicial há troca de mensagens sem troca de parâmetros, não há temporização (*time out)*, o número de nós é apenas dois, sendo um receptor e o outro emissor, tudo isto para facilitar a visualização dos mecanismos do protocolo.

Aqui se iniciou toda a metodologia incremental, foi feito desta forma porque modelos mais elaborados eram de difícil análise e correção. A construção de estruturas simples com incrementos contínuos em complexidade é a maneira mais prática de uma descrição estruturada para a especificação formal de protocolos extensos. A descrição desta maneira tem por objetivo testar a maioria das propriedades descritas no item 3.6.2.1..

4.3.1.2. Simulações Intermediárias

A simulação seguinte foi feita com três nós, sendo dois receptores e apenas um emissor. Os dois nós receptores possuem a capacidade de receber mensagens, analisá-las e as encaminhar para o outro nó, quando ele próprio não é o destino final da mensagem. Estas são as funcionalidades básicas do encaminhamento de rotas e da montagem do vetor de rotas. Neste ponto se insere a descrição de nó intermediário e da montagem das rotas.

A figura 11 descreve uma rede com três estações e seus usuários, USER1, USER2 e USER3. Os processos DSR1, DSR2 e DSR3 estão subdividos em dois processos encadeados.

Em DSR1, o processo ROTREQ é responsável pela busca da rota e o processo ENVDADOS pelo envio da mensagem (dados).

Em DSR2 e DSR3, os processos ROTREP1 e ROTREP2 descrevem o mecanismo para o reenvio de uma mensagem de busca de rotas, ou seja, o reencaminhamento de um *Route Request*, através das mensagens rRpCA e rRpBA.

O tratamento de dados já é possível através dos processos ENVDADOS, RECEBEDADOS1 e RECEBEDADOS2.

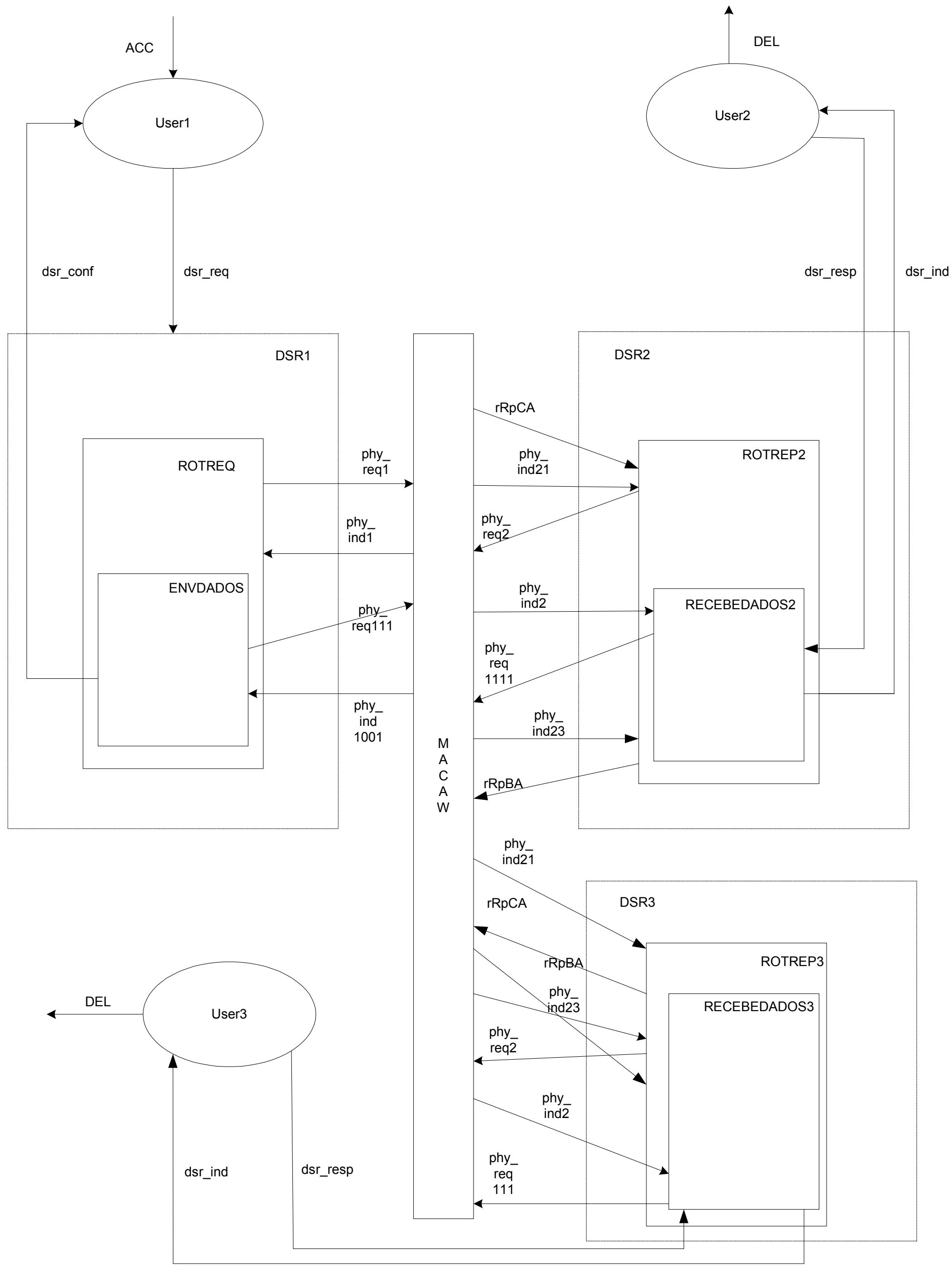

Figura 11 - Protocolo DSR sem Passagem de Parâmetros com Estruturas de Reencaminhamento de Rotas

Foi feita uma simulação com cinco nós, quatro são receptores e apenas um é emissor. Aqui dois dos nós receptores não possuem a capacidade de se comunicar diretamente com o nó emissor, porém são capazes de participar do roteamento das mensagens, participam do *Route Request* e *Route Reply*, encaminham mensagens entre si e para os demais nós da rede. Está característica equivale a nós que estejam distantes do emissor, cujas mensagens só lhes chegariam através de uma rota com múltiplos saltos, sendo também participantes das demais rotas de múltiplos saltos.

Nesta etapa do projeto estava consolidada a mecânica dos mecanismos de busca e confirmação de rotas através das mensagens de *Route Request* e *Route Reply*. O próximo passo seria a inserção de novas características que lhe aproximasse do caso de funcionamento mais real possível.

A figura 12 mostra o gráfico de estados e transições para as cinco estações, como referidas nesta quarta simulação.

Cada ramo (braço) do gráfico mostra uma alternativa de rota para cada nó da rede, todas com origem na estação emissora.

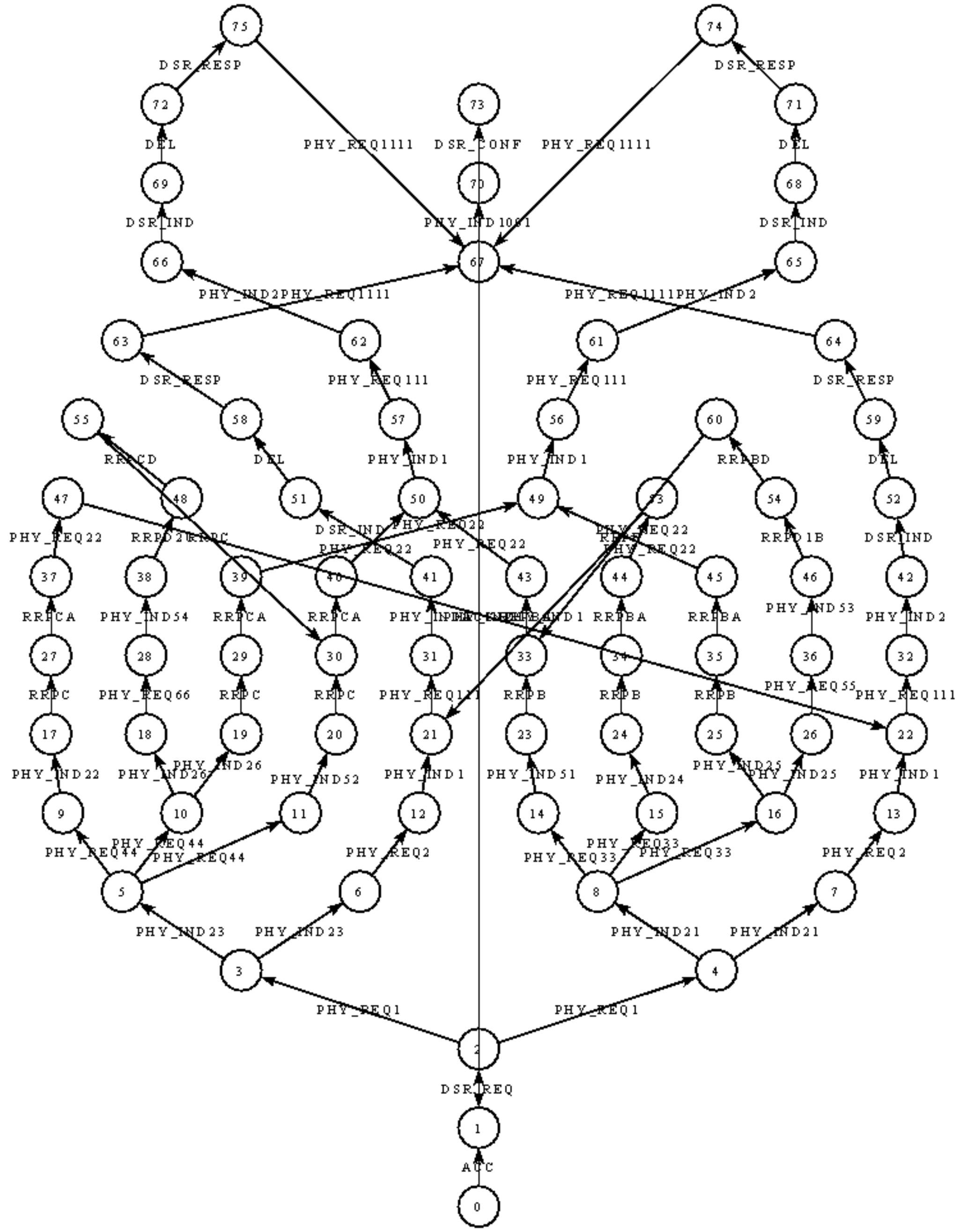

Figura 12 - Simulação de Cinco Estações através de Processos Simples

Até este ponto o protocolo já está bem estabelecido e consolidado, seus mecanismos de troca de mensagens já estão bastante robustos para garantir que ele convergirá mesmo para uma quantidade de nós na rede bem maiores do que cinco. Para isto, seria necessária apenas uma repetição de alguns dos processos já testados nestas simulações.

É importante salientar que em todas as fases do projeto foram testadas a existência de *deadlocks*, se o protocolo era vivo, reinicializável e se respeitava às propriedades descritas no item 3.6.2.1..

Como todas as simulações convergiram para este tipo de abordagem, cabe aqui salientar a necessidade da passagem para a outra fase do projeto que seria a inclusão de temporizadores para as próximas simulações.

Na maioria dos protocolos bem estruturados, a toda mensagem enviada será disparado um temporizador, um *timer*, que conta um determinado tempo e a partir daí repete novamente o envio da mensagem, isto por um certo número de vezes, determinado estatisticamente para cada situação de cada iteração, em cada processo de cada protocolo. Não obtendo êxito, o protocolo abandonará aquela mensagem e enviará para quem de direito a impossibilidade do envio desta.

A linguagem LOTOS testa todas as possibilidades de sucesso ou insucesso do protocolo, a cada envio de mensagem é posto uma alternativa de temporização com sua respectiva perda no meio e posterior *time out*, que levará o protocolo a repetir novamente aquela mensagem com alternativas de sucesso ou não.

Em LOTOS o *timer* é abstrato, sua forma representativa generalizada permite abstrair quaisquer valores para o *timer*. Não há perdas de funcionalidade nem de generalidade, num caso real o *timer* inserido em LOTOS poderá sempre ser substituído por quaisquer valores.

Foram simuladas duas redes, com duas e três estações respectivamente, esta última obteve um número elevado de estados e transições, vide itens 11 e 12 da tabela 04, causando a inviabilidade de análises qualitativas das respostas obtidas. Esta excessiva quantidade de estados e transições, devido a inserção do *timer*, impossibilita qualquer análise gráfica. Isto se deve às estruturas de programação do LOTOS, que necessitam de temporizadores sincronizados com as perdas do meio para impossibilitarem a geração de infinitas alternativas.

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 de passagem de parâmetros nos protocolos, isto para permitir a distinção das mensagens e com isso uma diminuição da quantidade destas, já que conterão os endereços físicos de cada nó, como previsto nos *drafts* do IETF.

Utilizando a passagem de parâmetros já se pode descrever com mais detalhes o protocolo como, por exemplo, montar vetores de rotas, como no caso real do protocolo de roteamento DSR, fazer análise das rotas, dos saltos e da quantidade de nós em cada rota, entre outras. Além disso, permitirá uma maior visualização, melhor rastreamento, quanto ao caminho percorrido por cada mensagem quando trocada entre cada estação.

Com a passagem de parâmetros a escrita dos protocolos em LOTOS se simplifica, a não utilização desde o início da mesma se deve ao fato do pouco domínio da mecânica de funcionamento do protocolo. O uso da passagem de parâmetros nesta fase implicaria na proliferação de erros, o que inviabilizaria a correção do protocolo. Com o domínio desta mecânica e com a inserção de processos temporizadores, a utilização de passagem de parâmetros se torna bem mais atraente e eficaz.

Daqui a diante os protocolos são totalmente reescrito, fazendo a inclusão de parâmetros como nome dos nós, tipos de nós, mensagens, tipos de mensagens, dados, tipos de dados, etc.

Retorna-se a fase inicial do protocolo, como ele seria na prática, para um caso real, utiliza-se novamente a metodologia de um emissor e um receptor apenas para ilustrar as funcionalidades básicas de cada entidade de uma estação da rede *ad-hoc*. Nesta fase os nós apenas trocam mensagens entre si, com possibilidades de rejeição de outros nós e da impossibilidade do envio de mensagens para si mesmo ou de reenvio.

Inicialmente os temporizadores foram inseridos entre as trocas de mensagens como descritos no *draft* do IETF, porém a difícil visualização dos resultados, ocasionada pelos elevados números de estados e transições gerados por estes causou sua desativação. Os temporizadores estarão incluídos nos últimos experimentos desta simulação.

A figura 13 representa com três usuários USER1, USER2 e USER3 em três estações com seus protocolos de roteamento DSR1, DSR2 e DSR3.

DSR1 é composto por processos encadeados onde, RECEBEACK é iniciado por ENVDADOS, que por sua vez é iniciado por ROTREQ, quando do recebimento da rota para encaminhamento da mensagem.

A estrutura OU só iniciará a busca da rota após se certificar que a estação procurada é diferente da emissora, caso contrário enviará a negativa, através do "dsr_conf !NNO", para o usuário requisitante, USER1.

O DSR2 e DSR3 são compostos por processos encadeados, inicializáveis pelos seus sucessores, em resposta as mensagens recebidas sincronamente ao processo DSR1.

DSR2 e DSR3 contém o processo ROTREP, que contém o processo RECEBEDADOS, que por sua vez contém o processo ENVACK.

4.3.1.3. Simulação com Passagem de Parâmetros para Três Estações

Na oitava simulação, o protocolo possui três nós, dois receptores e um emissor, neste caso, há a possibilidade de troca de mensagens entre nós, descoberta de rotas de múltiplos saltos e reenvio destas rotas entre cada nó (destino, intermediário e origem). Nesta forma o protocolo já cobre todas as funcionalidades de busca, descoberta de rotas, resposta de rotas, rotas com múltiplos saltos, rotas inexistentes, estações fora de alcance, etc. Faltando inserir os temporizadores, possibilidades de nós independentes e outras funcionalidades que serão descritas posteriormente.

Uma observação importante é dizer que as estações receptoras ora estão ao alcance da emissora, ora estão inatingíveis por esta, criando a situação de nó intermediário para rotas de múltiplos saltos.

Uma outra observação seria sobre o processo PERDA, este faz o tratamento das rotas inexistentes e das estações fora de alcance das demais. Além disso, encerra o processo de busca, ou de envio de mensagem e sinaliza para o emissor para que este inicialize outro processo de *Route Request*.

A figura 13 mostra a estruturação de dados e do fluxo de mensagens para o incremento das novas possibilidades descritas anteriormente.

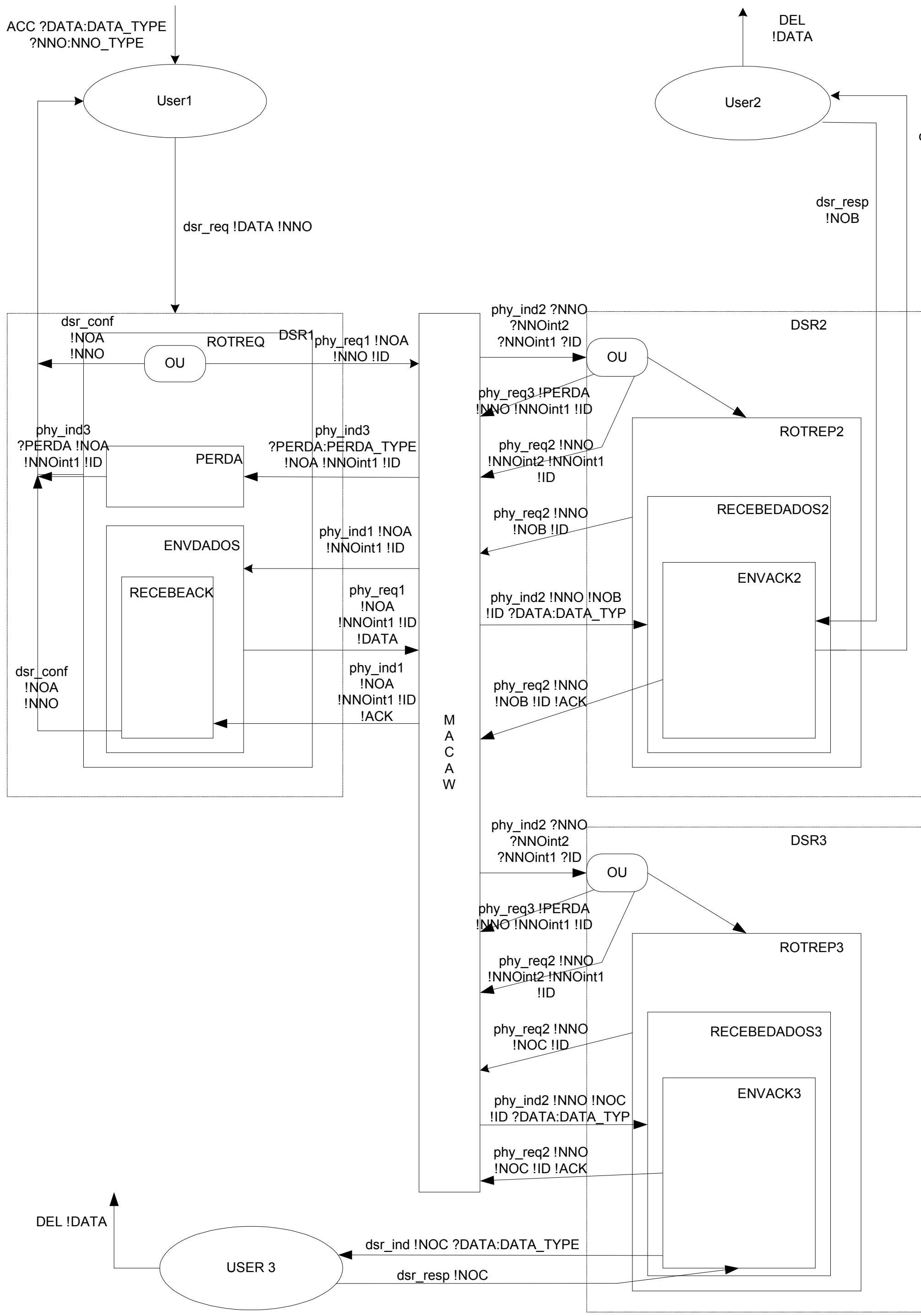

Figura 13 - Protocolo DSR com Passagem de Parâmetros entre Três Estações

O gráfico da figura 14, a seguir, mostra a convergência deste protocolo, com as características descritas anteriormente.

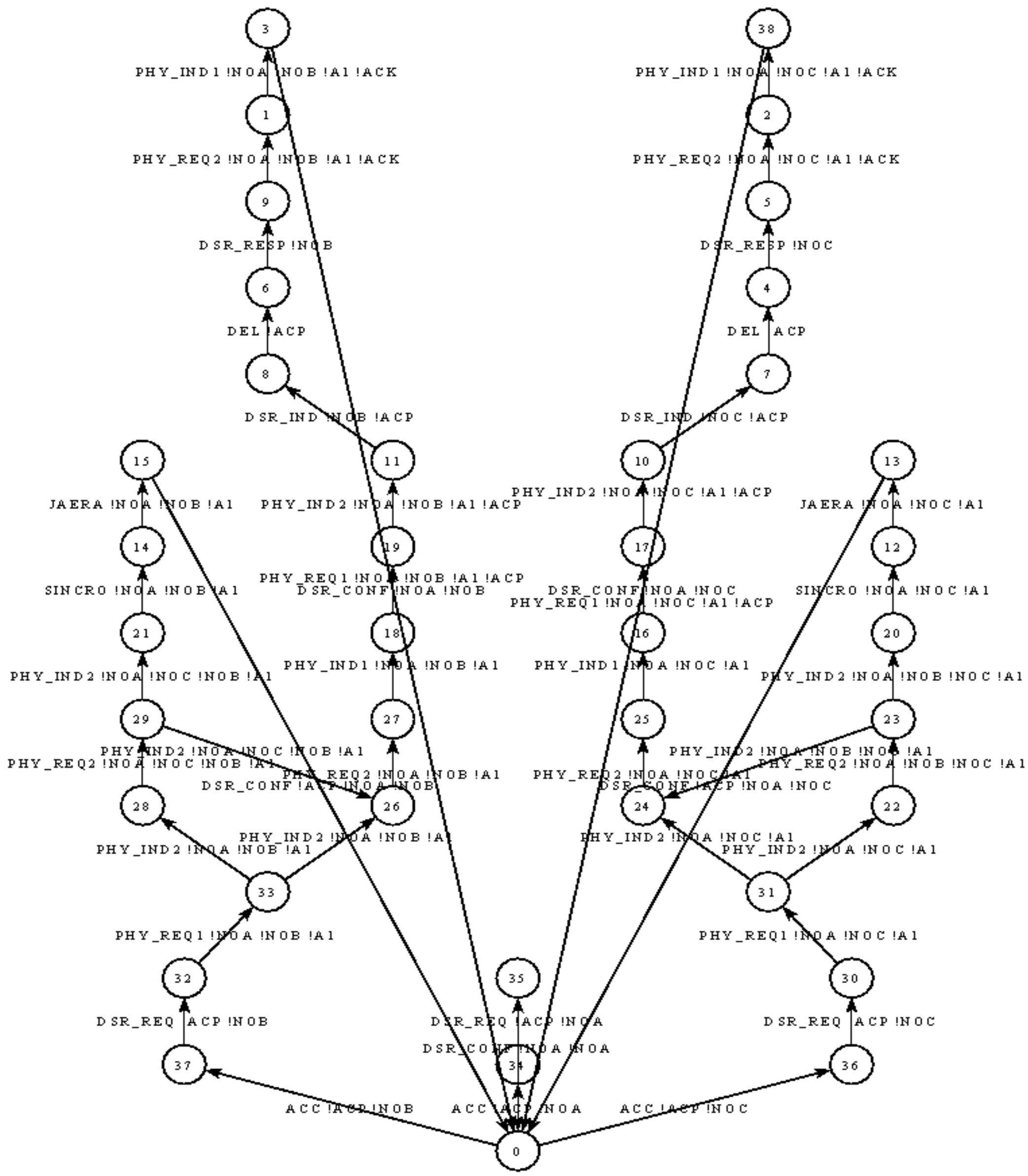

Figura 14 - Protocolo DSR com Passagem de Parâmetros para Três Estações

A partir deste momento o protocolo deverá assimilar todas as funcionalidades para uma estação de rede *ad-hoc*, passo a passo todas as características serão inseridas, começando pelo processo temporizador, que já foi testado e simulado anteriormente. Além disso, aqui estão sendo tratadas as possibilidades de troca de mensagens entre nós, descoberta de rotas de múltiplos saltos e reenvio destas rotas entre cada nó (destino, intermediário e origem). Existe ainda uma peculiaridade na linguagem LOTOS que é o reenvio de uma mensagem por uma estação para si mesmo, uma possibilidade em LOTOS que não ocorre na prática em redes rádio, como foi dito anteriormente, a linguagem LOTOS trata todas as possibilidades, inclusive as mais remotas. Este evento foi tratado através do processo *Perdendo[sincro, jaera]*, que nada mais é do que uma resposta ao nó enviador informando o ocorrido e que a mensagem deve ser novamente propagada pela rede. Este evento não causa falha na análise do protocolo, apenas é redundante.

Outro ponto importante a ser salientado é que nesta fase do projeto, estações que se desligaram da rede, ou estações inexistentes, ou rotas falhas serão informadas, através da mensagem (*dsr_conf !NNO ?AVISO:AVISO_TYPE;*), a camada superior.

Existe também a possibilidade de confirmação do recebimento da mensagem (*phy_ind1 !NNO !NNOint1 !ID !ACK;*) estação a estação, incluindo estações intermediárias das rotas, ou só confirmação entre estações destino e origem. Foi simulada a confirmação estação a estação.

O gráfico de estados e transições da figura 15 mostra todas as implementações descritas anteriormente, confirmando a convergência da simulação, além de ser viva, inicializável e não possuir *deadlocks*, como descrito no item 3.6.2.1..

O efeito visual do reenvio das mensagens, que são as interações entre as estruturas de trocas de mensagens e temporizadores, pode ser observado pelas setas de duplo sentido no gráfico seguinte, por exemplo, as setas entre os estados 2 e 72, 3 e 1, 7 e 8, etc. Estas setas mostram um envio sem êxito de uma mensagem, com posteriores reenvios até uma entrega com sucesso.

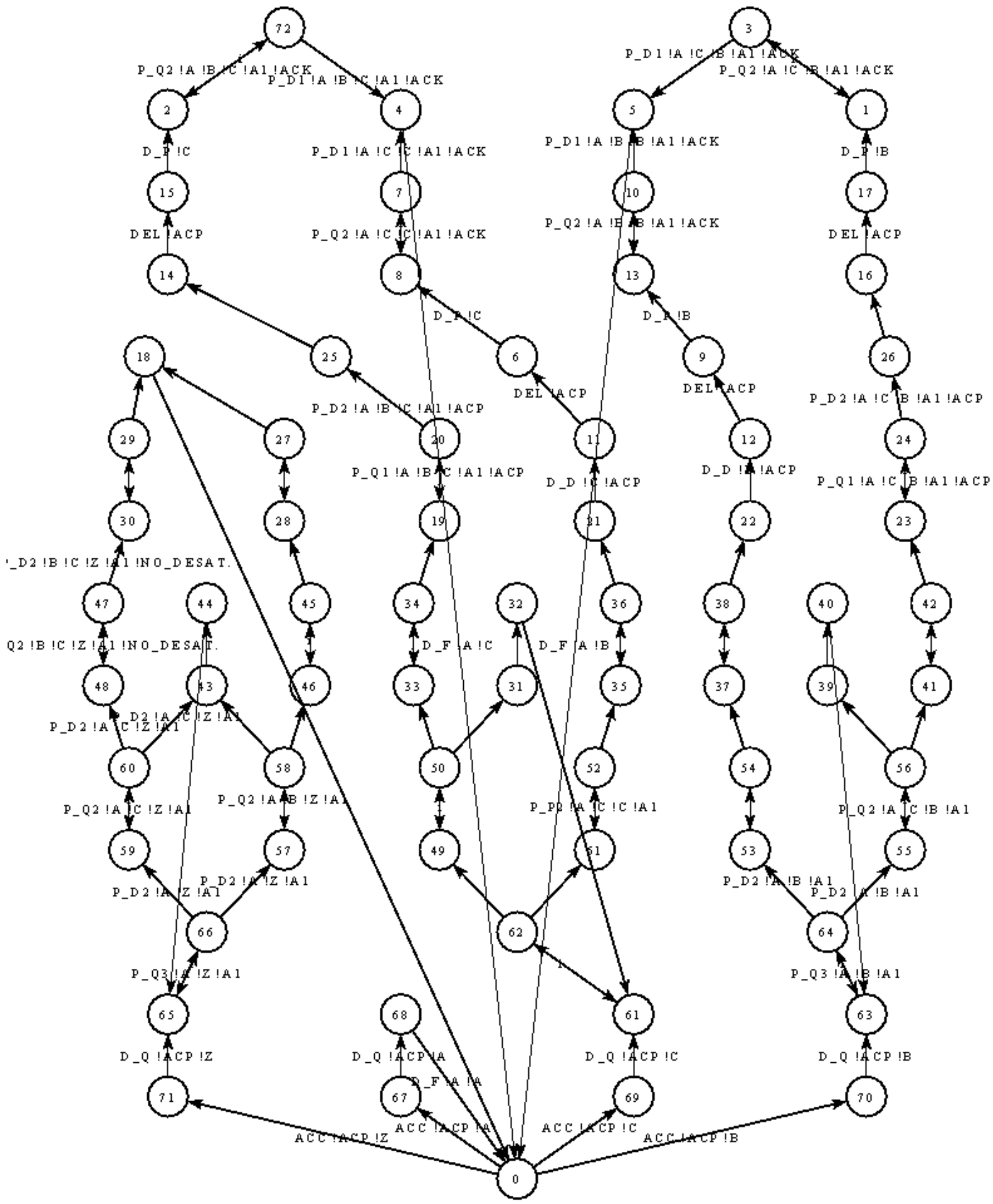

Figura 15 - DSR com Passagem de Parâmetros e Temporizadores

4.3.1.4. Simulação de Estações Completas, com Utilização de Passagem de Parâmetros e Temporizadores em cada Evento

Na décima simulação são inseridas, por completo, todas as funcionalidades de cada estação, desta forma todas as estações poderão desempenhar qualquer função de roteamento na rede, funções do DSR, seja na recepção, seja na emissão de mensagens, ou participar de qualquer rota como estação intermediária. Como é a primeira tentativa de se fazer estas implementações em LOTOS, neste projeto, inicialmente foram utilizados apenas duas estações, ambas receptoras e emissoras. Cada estação teria uma estrutura interna como a descrita na figura 16, observe que a diversidade de alternativas de interações com a outra estação e com o protocolo MACAW gera uma infinidade de possibilidades de trocas de mensagens, conseqüentemente de estados e transições.

É importante lembrar que entre cada interação existe também a possibilidade de um *time out* definido pelas estruturas dos temporizadores.

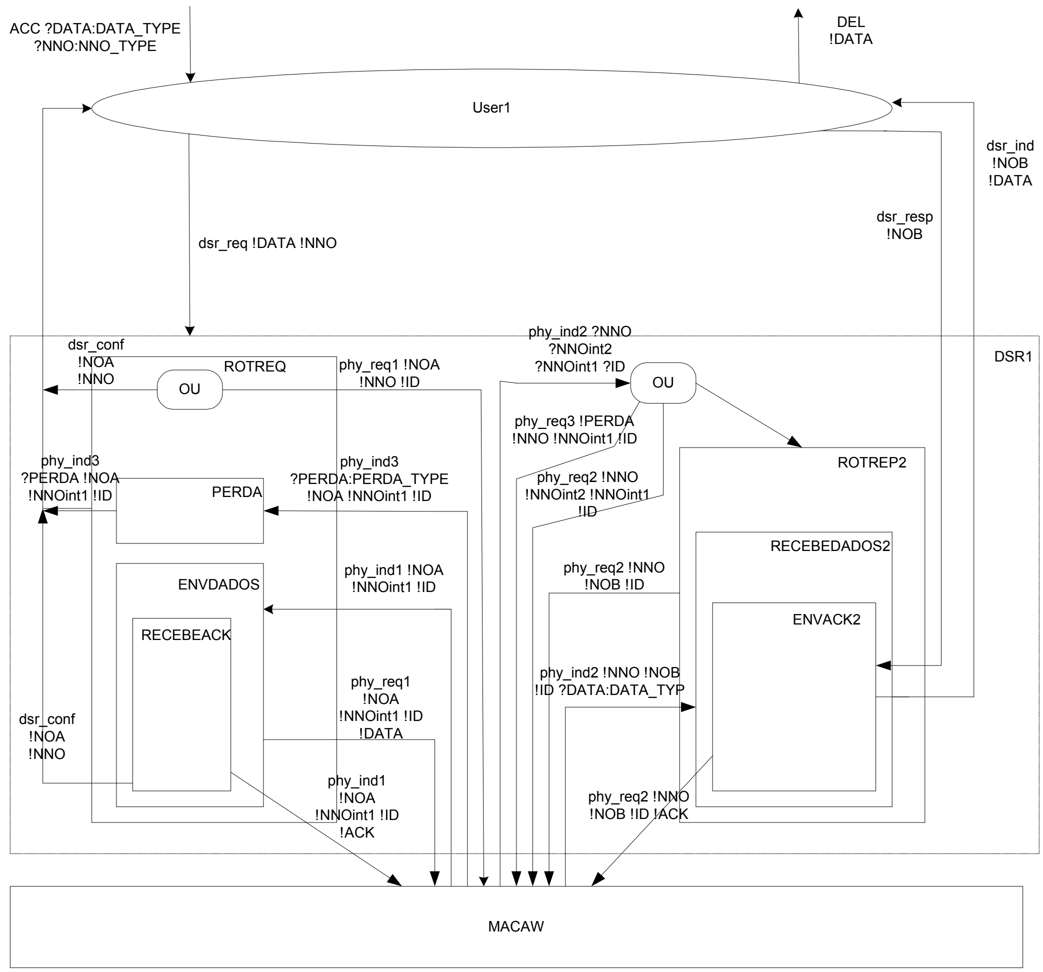

Figura 16 - Esquema Simplificado da Estrutura Interna de Decisão e Troca de Mensagens de cada Estação Autônoma

Como conclusão da simulação, o protocolo convergiu, pois todas as propriedades que demonstram a convergência do protocolo foram confirmadas, porém o número de estados e transições foi demasiadamente grande; 8.052.151 estados e transições 19.307.730.

O software de simulação não conseguiu minimizar o grafo, porém como resposta o aplicativo CADP informou que o protocolo estava livre de *deadlocks*. A capacidade de funcionamento do software está na ordem de dezenas de milhões de estados e transições, para minimizar este grafo a ferramenta precisava gerar outros estados e transições para testes de minimização de todo o grafo. Este mecanismo de funcionamento fez com que estas quantidades ultrapassassem a capacidade de análise do software.

4.3.2. Consolidações dos Resultados para o Protocolo DSR

O que se pode concluir com todos os testes e simulações feitos foram que o protocolo DSR realmente funciona e converge. A contribuição desta pesquisa foi reescrever o protocolo DSR em LOTOS e complementa-lo com as estruturas de decisões e análises que o *draft* do IETF descreve sucintamente, porém não diz como implementar. Tendo como base este *draft* e os das camadas de acesso ao meio foram feitas formalizações, em LOTOS, destas estruturas, posteriormente testadas, aperfeiçoadas através da metodologia de projeto e das fases das simulações mostradas anteriormente.

A seguir estão mostradas as estruturas de inicialização do protocolo DSR em LOTOS, identifica-se nesta estrutura a disposição dos nós e as estruturas que os compõem, bem como a perda de sincronismo e de mensagem no meio, além da representação deste meio físico.

Vêem-se as estruturas típicas de duas estações de redes *ad-hoc*, todas aptas a transmitir e receber dados independentemente, mostradas pelos nós 1 e 2. Estas descrevem todas estruturas das estações, de transmissão (User1, User2) e recepção (User11, User22), além disso as estruturas de envio e recebimento de cada estação; envio (DSR1, DSR2) e recebimento (DSR11, DSR22), além disso estrutura de correção de perda da mensagem (Perdendo) e a estrutura que representa o meio (Medium), futuramente substituída pelo protocolo de acesso ao meio.

As estruturas e processos foram assim divididos para ficarem mais didaticamente explícitos.

```
specification DSR [ACC, phy_req3,  … , DEL] : noexit
library        Boolean, NaturalNumber, DSRLIBB      endlib
behaviour
```

*(*hide phy_req3, ... , jaera in *)*

(((((

*(*N*) User1[ACC, dsr_req, dsr_conf](ACP of DATA_TYPE, NOC of NNO_TYPE)*

|||

User11[dsr_ind, DEL, dsr_resp]

*(*O*))*

|[dsr_req, dsr_conf, dsr_ind, dsr_resp]|

(

DSR1[phy_req3, ... , dsr_conf](A1 of ID_TYPE)

|||

*(*1*) DSR11[phy_req3, ... , jaera](ESTE_NO_ESTA_DESATIVADO of AVISO_TYPE)*

))

|||

((

*(*N*) User2[ACC, dsr_req, dsr_conf](ACP of DATA_TYPE, NOC of NNO_TYPE)*

|||

User22[dsr_ind, DEL, dsr_resp]

*(*O*))*

|[dsr_req, dsr_conf, dsr_ind, dsr_resp]|

(

DSR2[phy_req3, ... , jaera, dsr_conf](A1 of ID_TYPE)

|||

*(*2*) DSR22[phy_req3, ... , jaera](ESTE_NO_ESTA_DESATIVADO of AVISO_TYPE)*

))(|[sincro, jaera]|

(Perdendo[sincro, jaera]))

|[phy_req3, phy_ind3, ... , phy_ind1]|

MEDIUM[phy_req3, ... , phy_ind1]

)

where

(- Inicio da Descricao Completa do No 1 - *)*

process User1[ACC, ...

Na estrutura anterior, o processo principal é o *specification DSR*, este é o processo englobador, o qual abriga todos os demais que representam as estações, como seria o caso de uma rede. Os processos Users e DSRs identificam em cada nó suas estruturas de transmissão, recepção, encaminhamento e recebimento das mensagens. Todas as estruturas de *Route Reply* e *Route Request* se encontram dentro dos processos DSRs como sub-processos destes.

process DSR1[phy_req3, ... , dsr_conf] (ID:ID_TYPE): noexit:=

dsr_req !NOA ?DATA:DATA_TYPE ?NNO:NNO_TYPE;

RouteRequest1[phy_req3, ... , dsr_conf](DATA, NNO, ID)

where

process RouteRequest1[phy_req3, ...

process RouteReply1[phy_req3, ... ,jaera](AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) : noexit :=

phy_rep2 ...

Analisando a estrutura do processo *Route Request* de um dos nós, este processo recebe da estação usuária de origem, dados e a identificação da estação de destino, faz o tratamento da mensagem, insere seus dados e envia por inundação para a rede. A seguir um exemplo de parte da estrutura do processo *Route Request* para um das estações da rede.

RouteRequest3[phy_req3 , ... , jaera, dsr_conf](DATA, NNO, ID)

where

process RouteRequest3[phy_req3, ... , dsr_conf] (DATA:DATA_TYPE, ...) : noexit :=

(([NNO eq NOC]->(dsr_conf !NOC !NNO;

DSR3[phy_req3, ... , dsr_conf](ID)))

[]([NNO ne NOC]-> (let NNOint1 : NNO_TYPE = NNO in

RotReq3[phy_req3, ..., dsr_conf](NNOint1, DATA, NNO, ID))))

where

process RotReq3[phy_req3, ... , dsr_conf] (NNOint1 : NNO_TYPE , DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) : noexit :=

phy_req3 !NOC !NNOint1 !ID;

((jaera !NOC ...

Como descrito no *draft* do IETF, ao receber um *Route Request* para a busca de uma rota, a primeira estação ao recebe-la analisará seus campos e posteriormente reenviará para a rede se o destino não for ele próprio ou responderá com um *Route Reply*, que é mostrado abaixo para um das estações da rede.

process RouteReply3 [phy_req3, ..., jaera] (AVISO : AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) : noexit :=

phy_rep2 !NNO !NNOint2 !NOC !ID;

RecebeDados3[phy_req3, ... , jaera] (AVISO, NNO, NNOint2, ID)

where

process RecebeDados3[phy_req3, ... , jaera] (AVISO:AVISO_TYPE, ...) : noexit :=

((phy_ind2 ... .

Os dados relevantes para os processos de *Route Request* e *Route Reply* são aqueles testados pelas estruturas de decisão, na descrição em LOTOS mostrada anteriormente. Em nenhum momento foi necessário testar os identificadores da mensagem, Id. Estes participam das estruturas de troca de mensagens do protocolo de forma redundante, não tendo nenhuma necessidade nas estruturas de decisão. Porém são salientados pelo *draft* do IETF para o DSR como um dos elementos decisivos para definir se a mensagem *Route Request* já passou por aquela estação.

Estruturas como:

phy_ind2 ?NNO:NNO_TYPE !NOC ?NNOint1:NNO_TYPE ?ID:ID_TYPE ?AVISO:AVISO_TYPE;

phy_ind2 ?NNO:NNO_TYPE !NOC ?NNOint1:NNO_TYPE ?ID:ID_TYPE;

phy_ind2 ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;

Testam e identificam exatamente as estações no vetor de rotas e comparam se esta estação esta presente ou não naquela rota. Como pode se ver pelo campo ”!NOC ” este campo é o que será testado, os demais serão aceitos, bem como o campo “?ID:ID_TYPE“, o que facilitará e simplificará as estruturas de teste do protocolo e conseqüentemente diminuirá a quantidade de dados processados em cada estação.

Isto minimizará as estruturas de processamento do protocolo, já que cada estação testará apenas se seu endereço está presente ou não na rota. Todo o processamento com o identificador, Id, será feito apenas nos *Route Caches* das estações que guardaram aquela rota.

Pode-se observar que as estruturas de identificação implementadas foram criadas sem que realmente estivessem explícitas na forma de implementação nos *drafts* do IETF para o DSR.

Processos para perda de rotas e sincronismo, como o processo "Perdendo [sincro, jaera]", foram inseridos, isto porque, como descrito anteriormente, a linguagem LOTOS testa todas as possibilidades como, por exemplo, a recepção de uma mensagem pela própria estação que a enviou.

Seguem as estruturas de teste para perda de rotas e de sincronismo:

phy_ind2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;

e

([NNOint1 ne NOB]-> (([NNOint2 eq NOB]->
(sincro !NNO !NNOint1 !ID; ...

process Perdendo [sincro, jaera]:noexit :=
sincro ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;
jaera !NNO !NNOint1 !ID;
Perdendo[sincro, jaera]
endproc ...

[]

(phy_ind2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;

(([NNOint1 eq NOB]->

RouteReply2[phy_req3, ... , TIMEOUT_LOSS7
](AVISO, NNO, NNOint2, ID))

[]

([NNOint1 ne NOB]-> (([NNOint2 eq NOB]->

(sincro !NNO !NNOint1
!ID;

DSR2[phy_req3, ... , TIMEOUT_LOSS7] (AVISO)))

[]

([NNOint2 ne NOB]->

(Repita_phy_req2[phy_req3, ...

Estas estruturas são necessárias porque o LOTOS trabalha com troca de mensagens, uma a uma, a cada envio de mensagem deverá ser respondido com uma e apenas uma resposta, neste caso se a própria estação que a enviou receber a mensagem, esta deverá informar ao processo que iniciou todo o sistema a ocorrência deste tipo de erro ocasionado pela linguagem LOTOS e o processo origem deverá reiniciar o sistema novamente.

A ocorrência de busca de uma estação desativada ou inexistente na rede é tratada pela estrutura do processo a seguir, que está inserido nos processos do protocolo DSR.

(phy_ind2 ?NNO:NNO_TYPE !NOB ?NNOint1:NNO_TYPE ?ID:ID_TYPE
?AVISO:AVISO_TYPE;

Repita_phy_req3[phy_req3, ... , TIMEOUT_LOSS7](NNO, ..., AVISO)
) [] ...

A estrutura indica *?AVISO:AVISO_TYPE* para as estações origem *NNO:NNO_TYPE* e intermediárias *NNOint2:NNO_TYPE* que tal estação destino *NNO:NNO_TYPE* não está mais ativa na rede.

A estrutura anterior exemplifica o processo para tratamento de estação perdida do protocolo DSR, sendo ela parte da estrutura *Route Error*.

Os Apêndices A e B mostram a última simulação do protocolo, com sua biblioteca para passagem de parâmetros, ilustrando todas as estruturas descritas neste experimento.

4.3.3. Resumo dos Experimentos do Protocolo DSR

Alguns dos resultados obtidos nas diversas fases da simulação estão descritos na tabela 04, mostrando a seqüência lógica e cronológica dentro da filosofia de evolução incremental em complexidade do protocolo, descrito e mostrado anteriormente.

A tabela 04 é descrita da seguinte forma:

-**Simulação** se refere a protocolo ou o serviço simulado;

-**Nós da rede** se refere ao número total de nós desta, quer sejam emissores, ou receptores, ou ambos, ao mesmo tempo;

-**Nós emissores** contabiliza aquelas estações que tem como propriedade iniciar uma troca de mensagens;

-**Nós receptores** contabiliza aquelas estações que respondem a uma solicitação de troca de mensagens;

-**Estados** descreve o número de estados gerados pela ferramenta responsável pela obtenção do grafo LTS;

-**Transições** descreve o número de transições gerados pela ferramenta responsável pela obtenção do grafo LTS;

-O campo **temporizado**r faz menção se o protocolo tem dentro de sua especificação formal processos que atribuem tempo a cada iteração.

item	Simulação	Nós da rede	Nós emissores	Nós Receptores	Estados	Transições	Temporizador	Parâmetros	Minimizado
01	primeira	02	01	01	15	15	não	não	não
02	primeira	02	01	01	14	14	não	não	sim
03	segunda	03	01	02	36	39	não	não	não
04	segunda	03	01	02	24	27	não	não	sim
05	terceira	04	01	03	51	56	não	não	não
06	terceira	04	01	03	26	31	não	não	sim
07	quarta	05	01	04	76	85	não	não	não
08	quarta	05	01	04	36	45	não	não	sim
09	quinta	02	01	01	54	106	sim	não	não
10	quinta	02	01	01	47	95	sim	não	sim
11	sexta	02	01	02	2231	6154	sim	não	não
12	sexta	02	01	02	1472	4181	sim	não	sim
13	sétima	02	01	01	29	33	não	sim	não
14	sétima	02	01	01	15	16	não	sim	sim
15	oitava	03	01	02	3822	4670	não	sim	não
16	oitava	03	01	02	39	45	não	sim	sim
17	nona	03	01	02	138788	201940	sim	sim	não
18	nona	03	01	02	73	106	sim	sim	sim
19	décima	02	02	02	8052151	19307730	não	sim	não
20	décima	02	02	02	(1)	(1)	não	sim	sim
23	serviço	03	X	X	06	12	não	não	não

(1) Informa que estourou a capacidade de trabalho do software CADP.

Tabela 04 - Resultados Obtidos nos Experimentos do Protocolo DSR

Analisando seqüencialmente os experimentos mostrados pela tabela 04 é observável que a complexidade é ampliada, quando se acrescentam características de emissão e transmissão em cada nó; quando se inseri característica quer seja, com passagem de parâmetros, acréscimo de temporizadores, ou ambos e principalmente quando se amplia o número de estações na rede.

Por estes motivos à evolução dos experimentos seguiu um caminho natural de complexidade, passando primeiro pelos processos simples, seguido do aumento de estações e da complexidade destas; numa fase seguinte a inclusão de temporizadores, posteriormente passagem de parâmetros e finalmente a inserção de todas as características conjuntamente, com o incremento gradual de estações e da complexidade destas, possibilitando inserir ao protocolo um máximo de funcionalidades que o *draft* do IETF lhe atribui.

Seguindo esta metodologia a décima simulação possui todos os requisitos descritos anteriormente e convergiu, porém se tornou complexa demais para ser minimizado como descrito no item 4.3.1.4.2..

Houve ainda uma última simulação para uma rede possuidora de três estações como as que foram descritas no item 4.3.1.4.2., sendo todas aptas a serem receptoras e transmissoras simultaneamente. Após oito dias ininterruptos de simulação foi encerrado o aplicativo, porque mesmo sem haver *deadlocks*, o número de estados e transições foi demasiadamente grande para o CADP, superior a dezenas de milhões, impossibilitando a o término da simulação do protocolo.

A afirmação de não haver *deadlocks* é consistente, porque a metodologia de simulação utilizada testou todas as características, possibilidades e peculiaridades do protocolo, através de simulações anteriores que sempre convergiram. Como a terceira estação inserida nesta simulação era exatamente igual as duas descritas na anterior, além disso todas foram testadas duas a duas, então é possível concluir que a convergência é garantida em virtude das várias etapas concluídas com êxito para esta simulação.

4.3.3. Conclusões dos Experimentos do Protocolo DSR

Em razão da metodologia utilizada e da potencialidade da ferramenta é possível ressaltar os seguintes resultados e considerações:

-em primeiro lugar foram utilizados apenas documentos oficiais do IETF para especificação do protocolo em LOTOS;

-foram feitas especificações das partes mais essenciais do protocolo, em várias fases, utilizando linguagem para especificação formal, LOTOS, para verificação de consistência, convergência e estruturação do protocolo;

-todas as estruturas que não estavam formalmente descritas nos documentos IETF foram aqui definidas para os devidos testes de simulação com as ferramentas do CADP;

-todos os testes com o protocolo, escrito em LOTOS, foram feitos em ambiente de simulação cujas condicionantes do meio, que seria a propagação em espaço livre, mais rigorosos que num caso real, por exemplo, o fato de todas as estações terem probabilidades iguais de receberem uma mensagem, inclusive a própria emissora, caso muito mais rigoroso que o real. Estas hipóteses forçaram a escrita do protocolo com um maior grau de dificuldade para tratamento destas possibilidades;

-como referência o *draft* do IETF para o protocolo DSR foram definidas as estruturas de teste para a tomada de decisão do protocolo quanto a próxima iteração a ser seguida para convergência da troca de mensagens das diversas fases de funcionamento do protocolo, sejam ele: busca de rotas *Route Request*, encaminhamento da busca, resposta da busca *Route Reply*, encaminhamento da resposta, resposta de rotas e estações inexistentes *Route Error*, encaminhamento de dados, resposta e encaminhamentos de *ACKs*, etc;

-como efetivo trabalho foram criadas várias estruturas de decisão para tratamento de todas as hipóteses tratadas nos documentos do IETF, além daquelas impostas pela linguagem LOTOS, o que possibilitou a constatação da convergência do protocolo e da conclusão de que o campo de identificação, ID, não é realmente necessário na busca e descoberta de rotas, ou qualquer outra fase do protocolo. Isto porque em LOTOS não foi necessário nenhum teste deste campo em nenhuma das estruturas do mesmo. Assim se conclui que o ID só será necessário para as estações que habilitarem as rotas nos seus *Route Caches*, visto que o ID será utilizado para a indexação destas rotas nestes *Route Caches*;

-esta abordagem quanto ao ID fará com que o processamento na troca de mensagens do protocolo diminua, deixando esta parte do processamento para quando a estação estiver armazenando o vetor de rotas em seu *Route Cache*.

4.4. O Protocolo de Roteamento LANMAR

O protocolo LANMAR foi testado e simulado seguindo as especificações do *draft* do IETF do grupo MANET.

Conforme este *draft*, o LANMAR tem especificado o FSR como protocolo de roteamento pró-ativo dentro das suas células e um outro protocolo pró-ativo, utilizando um mecanismo de número seqüencial de identificação de rotas, conforme descrito no DSDV [20], entre as células. Estes protocolos não são tão eficientes quando a rede se torna bastante móvel e/ou com o aumento do número de estações e da troca de mensagens na rede. O próprio LANMAR tem sua eficiência ligada a topologia da rede, onde o movimento entre grupos é primordial para o seu melhor funcionamento em relação aos demais protocolos [54], ou seja, a célula representa um grupo onde a velocidade relativa entre as estações componentes deve ser baixa.

Quando a rede começa a receber novas estações, novas células e/ou os nós começam a se "desgarrar" das células, o LANMAR passa a ter desempenho inferior dentre todos os protocolos [54]. Além disso, os protocolos reativos são bem superiores aos pró-ativos nestas situações. Por estes motivos foi proposto e utilizado o protocolo DSR para fazer o roteamento entre as células, como o AODV, ele é essencialmente reativo, vide item 2.3., porém de mais simples implementação.

A estrutura do DSR tem como característica minimizar as tabelas de rotas e conseqüentemente a necessidade de grande espaço em memória. As trocas de mensagens são ocasionais e os pacotes são menores. Atualmente, é o protocolo mais antigo do IETF para roteamento em redes *ad-hoc*, o de especificação mais completa, o mais difundido e o mais testado dentre os existentes [9, 10, 54].

O protocolo AODV faz o uso da troca de mensagens *HELLO*, isto o faz ser mais eficiente para a busca de rotas em redes com alta movimentação de estações, convergindo mais rapidamente que o DSR para este tipo de rede, pois suas tabelas de rotas estão sempre mais atualizadas. Uma rede muito móvel faz com que as trocas de mensagem *HELLO* se tornem muito intensas, causando considerável *overhead* na mesma [9, 10, 54].

O DSR é o protocolo mais recomendado para redes com moderado número de estações e para rotas que não tenham um grande número de saltos, além disso, a quantidade de trocas de mensagens é sempre bem menor que a do protocolo AODV, para qualquer tipo de rede.

Além do exposto anteriormente, as redes estudadas neste trabalho possuem poucas estações, por limitações da própria ferramenta, CADP. O que se pretende é manter uma efetiva coerência e convergência de todos os protocolos simulados, para validar a arquitetura proposta.

Um outro enfoque pode ser observado pelo leitor, quanto a obtenção de dados estatísticos dos protocolos e da rede. A ferramenta CADP não é a mais apropriada para este fim, sendo uma sugestão, inclusive para trabalhos futuros, à utilização do *software* ns-2 de simulação de redes, conforme descrito nos artigos [9,10].

O ponto principal do formalismo para formação da rede e da descrição dos protocolos desta, em LOTOS, é que cada um deles foi especificado considerando que o protocolo da camada imediatamente inferior a sua lhe prestava os serviços necessários e corretos, sem problemas ou erros, para o seu bom funcionamento. O que já foi feito com o protocolo de acesso ao meio MACAW e de roteamento DSR, este último será utilizado entre as células.

Assim, cada protocolo teve seu tratamento em separado e posteriormente foram interligados, pela troca de SDUs e PDUs entre eles, descritos como no modelo em camadas mostrado na figura 03.

Para as composições dos protocolos LANMAR foram utilizados:

-dentro das células um protocolo pró-ativo, como o FSR, para tratamento das rotas e troca de mensagens apenas no interior de cada célula;

-entre células, pelos motivos expostos acima, foi utilizado o DSR.

Quando uma estação não conseguir encontrar um destino, que não consta na sua tabela de rotas, criada pelo FSR no estabelecimento da rede e das células. Esta estação enviará uma requisição para seu líder de célula, para que ele se comunique com os demais líderes das outras células, através do protocolo DSR, para obtenção da rota para aquele endereço destino da estação desejada; apenas as estações líderes têm a capacidade de iniciar o protocolo reativo, DSR. As trocas de mensagens entre os protocolos são mostradas na figura 17.

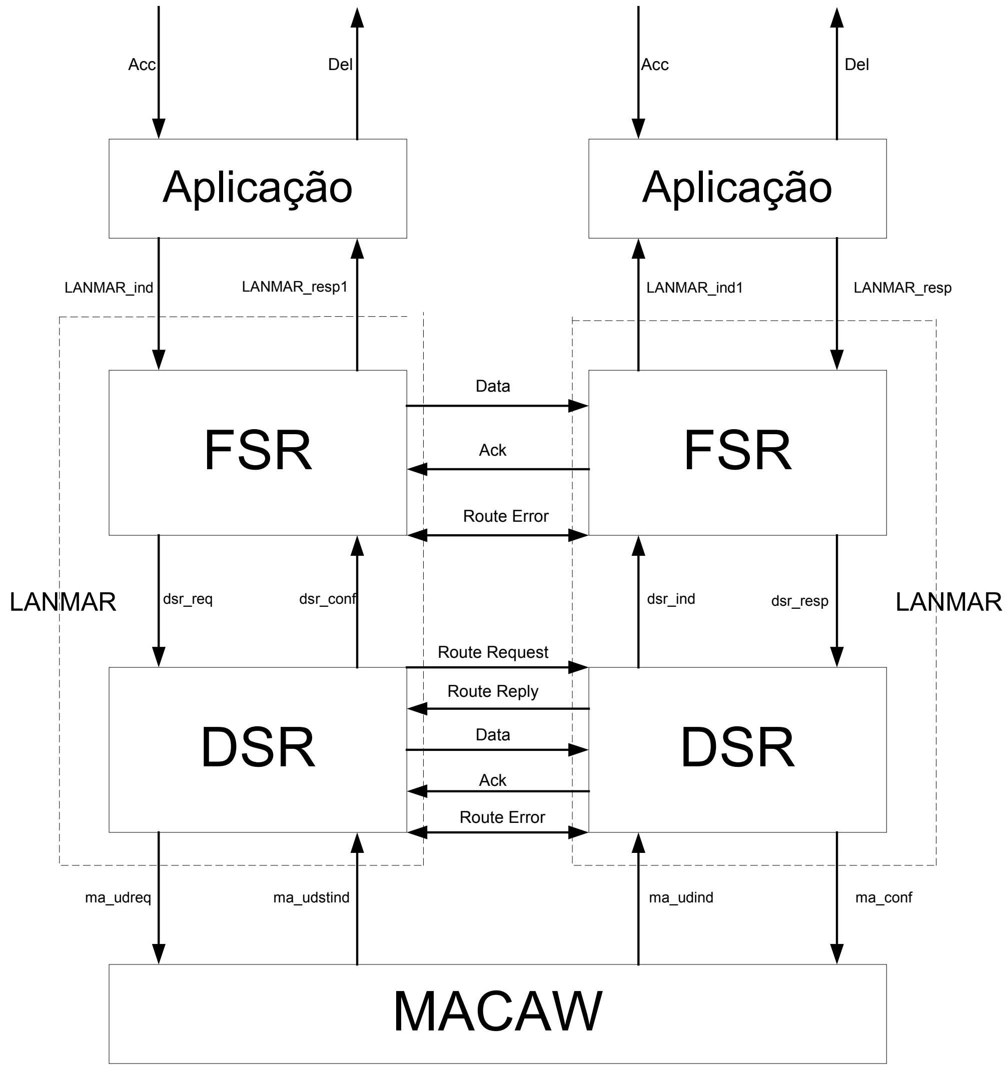

Figura 17 - Arquitetura do Protocolo LANMAR

4.4.1. Simulações do Protocolo de Roteamento LANMAR

Como a descrição de protocolos extensos é um projeto de difícil execução, optou-se por uma descrição incremental, de cada estação, cada processo, enfim, cada característica da rede. A filosofia de um aumento mínimo, passo a passo da descrição do protocolo, possibilita sua análise em diversas variações conceituais e em vários níveis de complexidade.

A estrutura inicial a nível conceitual é sempre bem simples, onde o modelo parecerá óbvio, de fácil análise e a complexidade se mostrará presente no decorrer do experimento.

Nestas simulações não se seguiu o aumento em complexidade de escrita pela linguagem LOTOS; não houveram as fases de escrita do protocolo apenas com processos simples, processos simples mais temporizadores, passagem de parâmetros sem e com temporizadores, etc. Todas as simulações já foram descritas com passagem de parâmetros e posteriormente inseridos temporizadores, porque a metodologia passo a passo, através das diversas fases da linguagem LOTOS já estava bem consolidada e quando necessário se utilizava exemplos dos protocolos anteriores para a retirada de dúvidas.

O aumento em complexidade descrito neste item se refere ao aumento em complexidade de cada estação, de cada célula, do número de estações por células, do número de células na rede, etc.

4.4.1.1. Simulação com Duas Estações

Como ponto inicial dos experimentos com o LANMAR foram, inicialmente, descritas duas estações LAM1 e LAM2, possuindo estruturas e processos que as possibilitem ser emissoras e receptoras entre si. Foram descritas apenas como estações simples em uma rede, fazendo a busca dos endereços das rotas existentes entre si. Apesar da descrição deste protocolo com apenas duas estações parecer óbvia, é necessária para a completa correção da descrição da mecânica do protocolo, para a troca de mensagens entre estações dentro e fora das células.

O gráfico de estados e transições teve sua minimização bem sucedida, porém sua visualização se torna confusa pela inserção de tratamentos de exceções com o objetivo de impedir diversos tipos de trocas de mensagens num protocolo em mero estado inicial. Além disso, já se iniciou a estruturação para o funcionamento de uma rede em células, o banco de dados já contém algumas estações subordinadas, sendo testadas em cada estação que possivelmente será uma futura líder de célula, como descreve o *draft* do IETF do protocolo LANMAR.

Estas alternativas foram, por agora, bloqueadas para facilitar o entendimento da mecânica de funcionamento do protocolo e estarão ativas nas simulações futuras.

4.4.1.2. Simulações Intermediárias

Em outro experimento o nível de complexidade já começa a crescer, as duas estações LAM1 e LAM2 são realmente emissoras e receptoras entre si. Já possuem a forma de futuras líderes de células, possuindo estruturas que armazenam os endereços das líderes de outras células e também de análise para descoberta de rotas com novas líderes.

No experimento seguinte foram estruturadas duas células A1 e A2, onde duas estações LAM1 e LAM2 são as líderes respectivamente destas células, transmitindo e recebendo entre si.

Agora existe também uma estação A, subordinada e apenas receptora da líder da célula A1, ou seja, LAM1.

A figura 18 mostra como seria a estrutura dos processos dentro de uma célula.

Obs.: É preciso lembrar que existe uma terceira estação, líder da célula A2, a LAM2, que interage diretamente com LAM1, por serem líderes de células. A não representação desta estação se fez necessária para que houvesse mais clareza na visualização dos processos da célula A1.

Os processos de cada estação são descritos da seguinte forma:

O usuário da camada imediatamente superior a de roteamento é representado pelo processo User1, este recebe, transmite mensagens e reside na estação LAM1.

A estação LAM2 possui o processo equivalente representado por User2.

Eles recebem mensagens das estações destino pelo ACC e entregam pelo DEL.

Enviam as mensagens recebidas para o protocolo LANMAR que está dividido em dois processos: o de transmissão, aqui representado por LANMAR11, e o de recepção, representado por LANMAR12.

A estação LAM2 possui, respectivamente, os processos LANMAR21 e LANMAR22.

Os processos para o funcionamento de uma transmissão e recepção serão aqui explicados e são equivalentes para LAM1 e LAM2, com exceção dos nomes dos processos, como descrito anteriormente:

Na transmissão:

O processo LANMAR11 ao receber uma mensagem de USER1, envia para o processo NoDoLA1, para analisar se a estação destino é a própria LAM1, ou uma estação que está no alcance do que seria a célula A1, ou uma estação pertence à rede, mas fora das fronteiras da futura célula A1.

Quando a estação destino está dentro da fronteira de LAM1 esta utilizará o processo ProcLA1 para acessa-la, seria o equivalente ao FSR dentro da célula. ProcLA1, que já foi estruturado na experiência anterior, foi desbloqueado e está em funcionamento interagindo com a estação subordinada da líder LAM1, ou seja, a estação A.

Se o destino é a própria estação a mensagem já encontrou o seu destino e será entregue através de "Entrg1".

Se o destino não for nenhuma das duas alternativas acima, o protocolo iniciará uma busca através do processo AnaliseLAM1 que buscará a estação destino.

Após encontrar o destino da mensagem, enviará a mesma através do processo Enviar1 e esperará a confirmação do sucesso da transmissão dentro do processo ReceberACK. Ao receber o ACK, confirmará ao processo USER1 o envio da mensagem e permitirá ao usuário USER1 e ao processo LANMAR1 iniciarem novamente todos os processos, que sejam, receber e enviar novas mensagens.

Na Recepção:

Quando LANMAR12 identificar que existe uma mensagem para usuário residente na estação LAM1, acionará o processo Receber1, para o recebimento efetivo da mensagem. Esta será enviada a seu destino final e confirmará o êxito da entrega de dados com um ACK, enviado pelo processo EnviarACK, o qual reiniciará todo o protocolo.

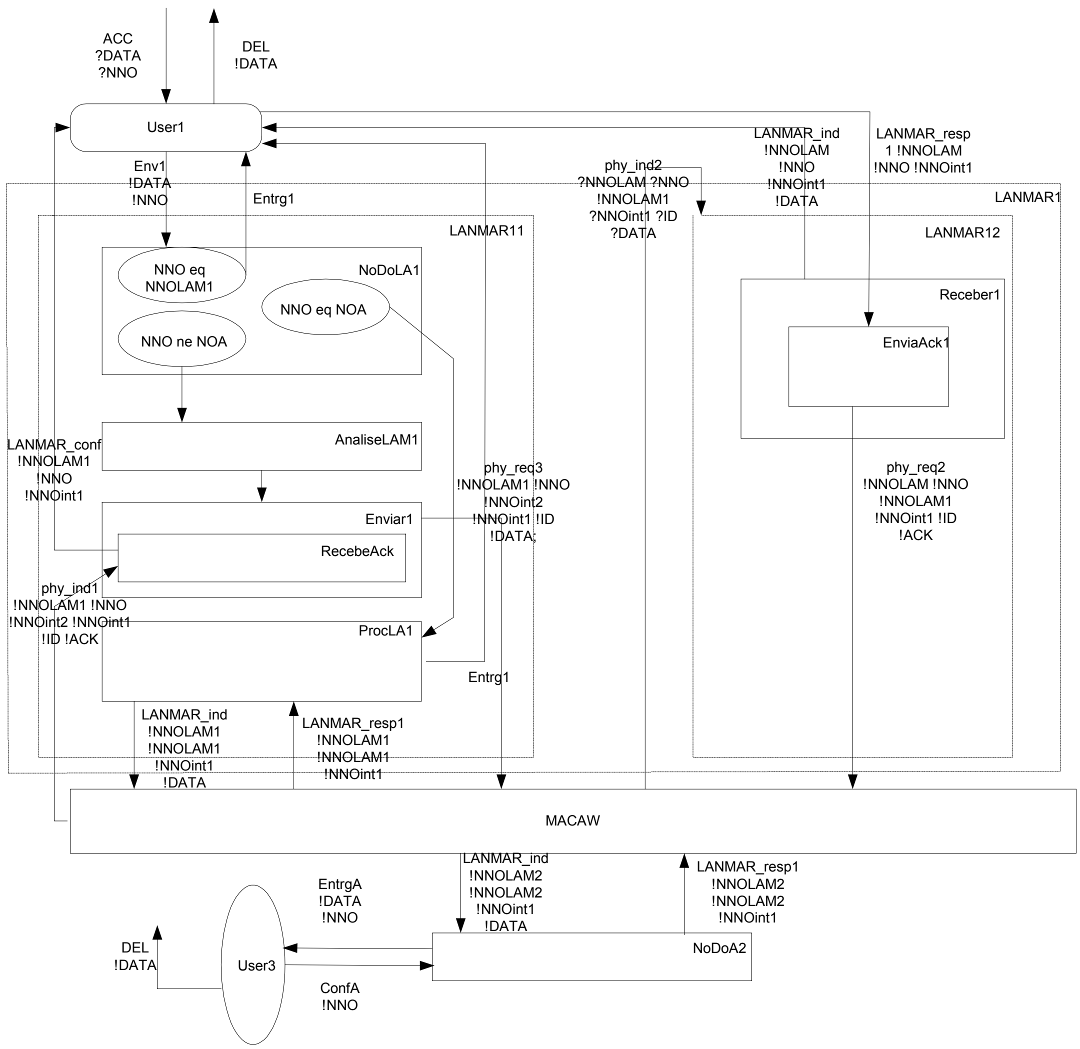

Figura 18 - Representação dos Processos de uma Célula com Uma Estação Líder e Uma Subordinada

No experimento seguinte foram descritas quatro estações estruturadas dentro de duas células A1 e A2, onde LAM1 e LAM2 são líderes, emissores e receptores entre si. Existem duas estações A e D que tem todas as estruturas de transmissão e recepção, mas ainda não transmitem. A estação A é receptora da líder da célula A1 e a estação D é receptora da líder da célula A2.

No sexto experimento, as duas células A1 e A2 com seus líderes LAM1 e LAM2, como na simulação anterior, podem ser emissores e receptores entre si e para suas estações subordinadas A e D. Estas últimas têm todas as estruturas de transmissão e recepção, mas apenas D ainda não transmite, A já transmite e recebe de seu líder LAM1.

A figura 19 mostra um esquema de funcionamento da estação subordinada A, com seus processos e estruturas para transmissão e recepção para sua estação líder.

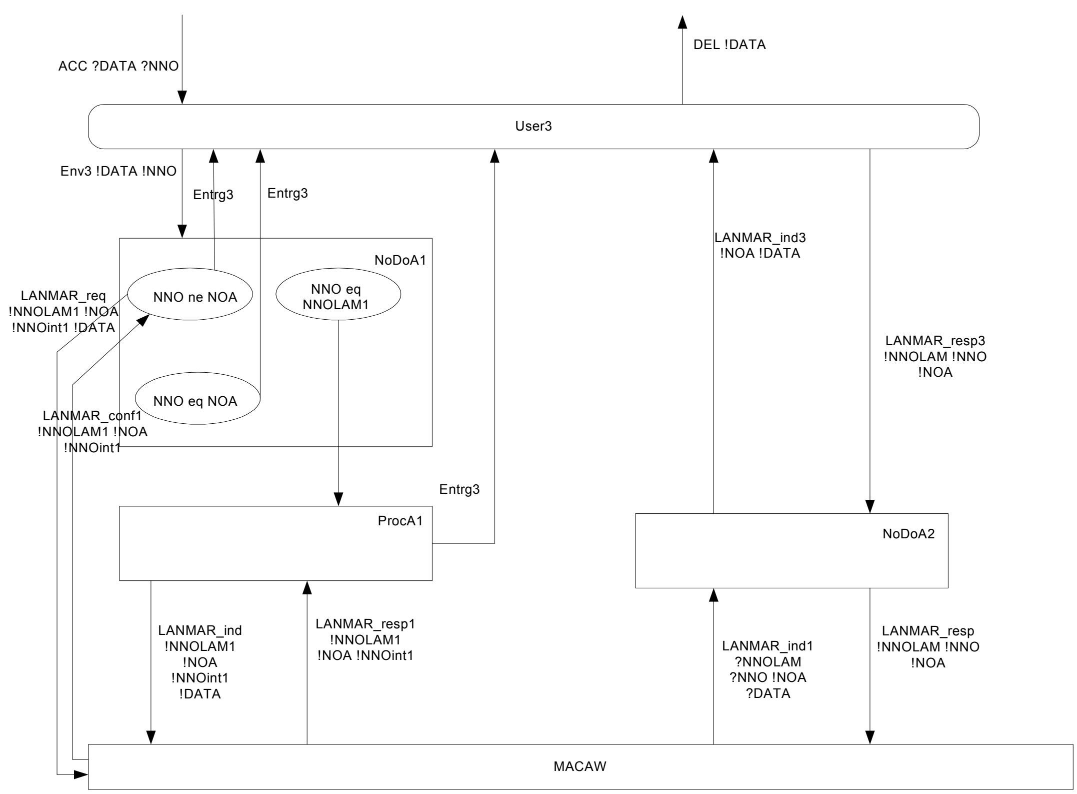

Figura 19 - Representação dos Processos de Uma Estação Subordinada

No sétimo experimento, as duas células A1 e A2, onde LAM1 e LAM2 são líderes, emissores e receptores entre si e as estações A e D tem todas as estruturas de transmissão e recepção, transmitindo e recebendo apenas para seus líderes.

Foi inserido o processo de sincronismo para ocupação do meio, para democratizar o acesso ao meio e visualizar, passo a passo, as interações entre processos, células e estações dentro e fora das células.

```
process sincronizador [ACC, sinc1,  ... , jaera] : noexit :=
        choice NNO:NNO_TYPE[]
                sinc2 !NNO;   jaera;
                        sincronizador [ACC,  ...  , jaera]
endproc
```

Esta estrutura de sincronismo se faz necessária para possibilitar uma visualização dos gráficos e permitir uma análise mais qualitativa dos resultados.

Na figura 20 é interessante observar o sincronismo entre as estações, estado 40, representando a troca de permissão para o acesso ao meio.

Seguindo a linha de raciocínio das simulações anteriores, o oitavo experimento se refere a duas células A1 e A2, onde LAM1 e LAM2 são líderes, emissores e receptores entre si e as estações A e D tem todas as estruturas de transmissão e recepção, mas se comunicam apenas com seus líderes de célula. As estações líderes transmitem e recebem para todas estações líderes da rede e para suas subordinadas.

Na figura 20, a comunicação da estação líder da célula A1, LAM1, com sua estação subordinada A, está representada pelo primeiro ramo da esquerda; a comunicação de LAM1 com a líder da outra célula, LAM2, está representada pelo segundo ramo da esquerda e a comunicação de A com LAM1 está representada pelo terceiro ramo da esquerda.

Na célula A2: a comunicação da estação líder da célula A2, LAM2, com sua estação subordinada D, está representada pelo primeiro ramo da direita; a comunicação de LAM2 com a líder da outra célula, LAM1, está representada pelo segundo ramo da direita; a comunicação de D com LAM2 está representada pelo terceiro ramo da direita.

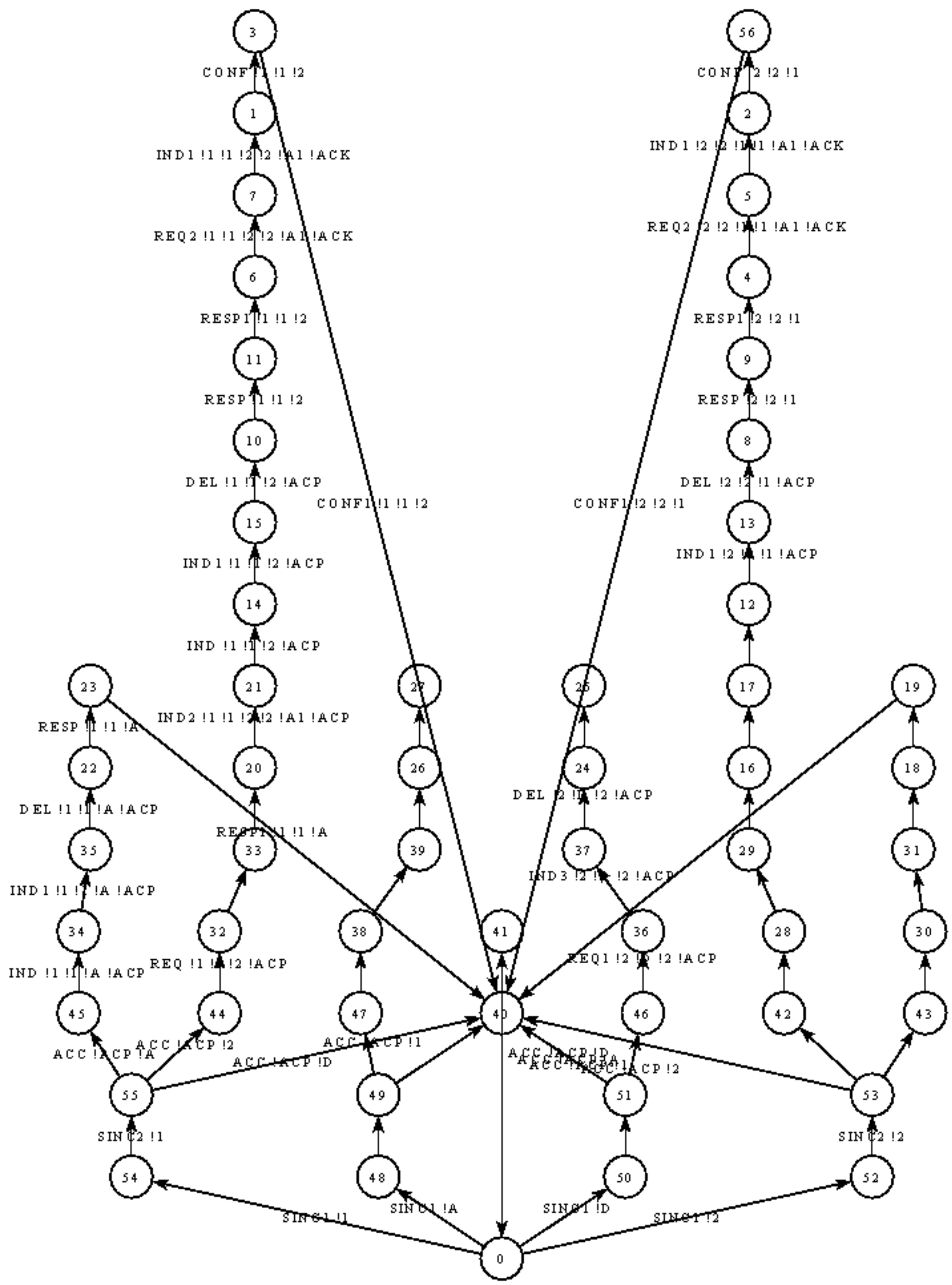

Figura 20 - Estados e Transições Convergentes para Duas Células e Quatro Estações com Processo de Sincronismo Inserido

4.4.1.3. Estações Líderes se Comunicando com qualquer Estação da Rede

Nesta configuração as duas células A1 e A2 tem LAM1 e LAM2 como suas estações líderes, respectivamente. Sendo emissoras e receptoras entre si, as estações A e D estão locadas uma em cada célula e possuem todas as estruturas de transmissão e recepção.

As estações líderes transmitem e recebem para todas estações líderes da rede, para suas próprias subordinadas diretamente e se comunicam com as subordinadas de outras células através de seus respectivos líderes de células. Ou seja, a comunicação com as estações subordinadas de outras células sempre será feita através das líderes destas.

As estações subordinadas apenas se comunicam com seus líderes de células, não podendo ainda, iniciar uma comunicação com as demais estações da rede.

A figura 21 mostra a representação da forma de comunicação, por meio dos líderes de células, entre estações de células distintas.

A figura 22 mostra a distribuição de estados e transições para um protocolo, cuja única diferença para o experimento anterior seria a possibilidade de comunicação entre a estação líder de uma célula com a estação subordinada de outra, através da líder de célula desta.

Comparando ao experimento anterior, esta modificação tem um impacto visual que pode ser observado através dos novos ramos do gráfico da figura 22. Este passou a ter oito ramos, dois além dos descritos no oitavo experimento.

São eles:

Na célula A2: a comunicação da estação líder A2, LAM2, com a estação subordinada A da célula A1, através de seu líder de célula LAM1, está representada pelo primeiro ramo da esquerda; a comunicação de LAM2 com a líder da outra célula, LAM1, está representada pelo segundo ramo da esquerda; a comunicação de LAM2 com sua subordinada D está representada pelo terceiro ramo da esquerda; a comunicação de D com sua líder de célula LAM2 está representada pelo quinto ramo da esquerda.

Na célula A1: a comunicação da estação líder da célula A1, LAM1, com sua subordinada A está representada pelo primeiro ramo da direita; a comunicação da estação líder da célula A1, LAM1, com a estação subordinada D da célula A2, através de seu líder de célula LAM2, está representada pelo segundo ramo da direita; a comunicação de LAM1 com a líder da outra célula, LAM2, está representada pelo terceiro ramo da direita; a comunicação de A com sua líder de célula LAM1 está representada pelo quinto ramo da direita.

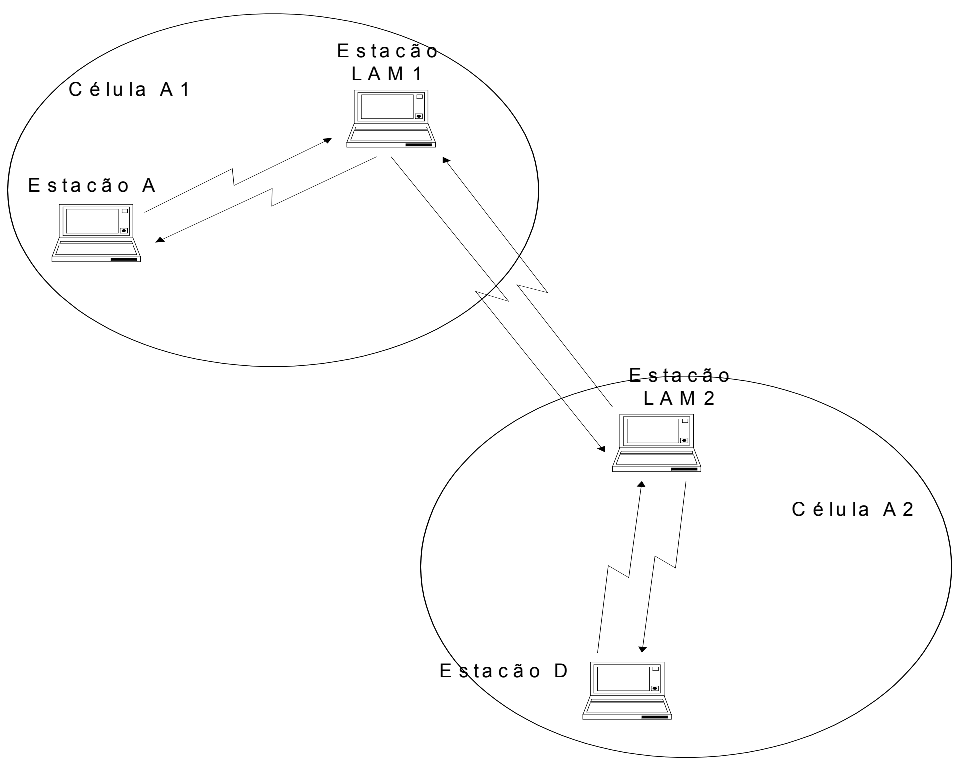

Figura 21 - Representação da Rede com Duas Células e Quatro Estações

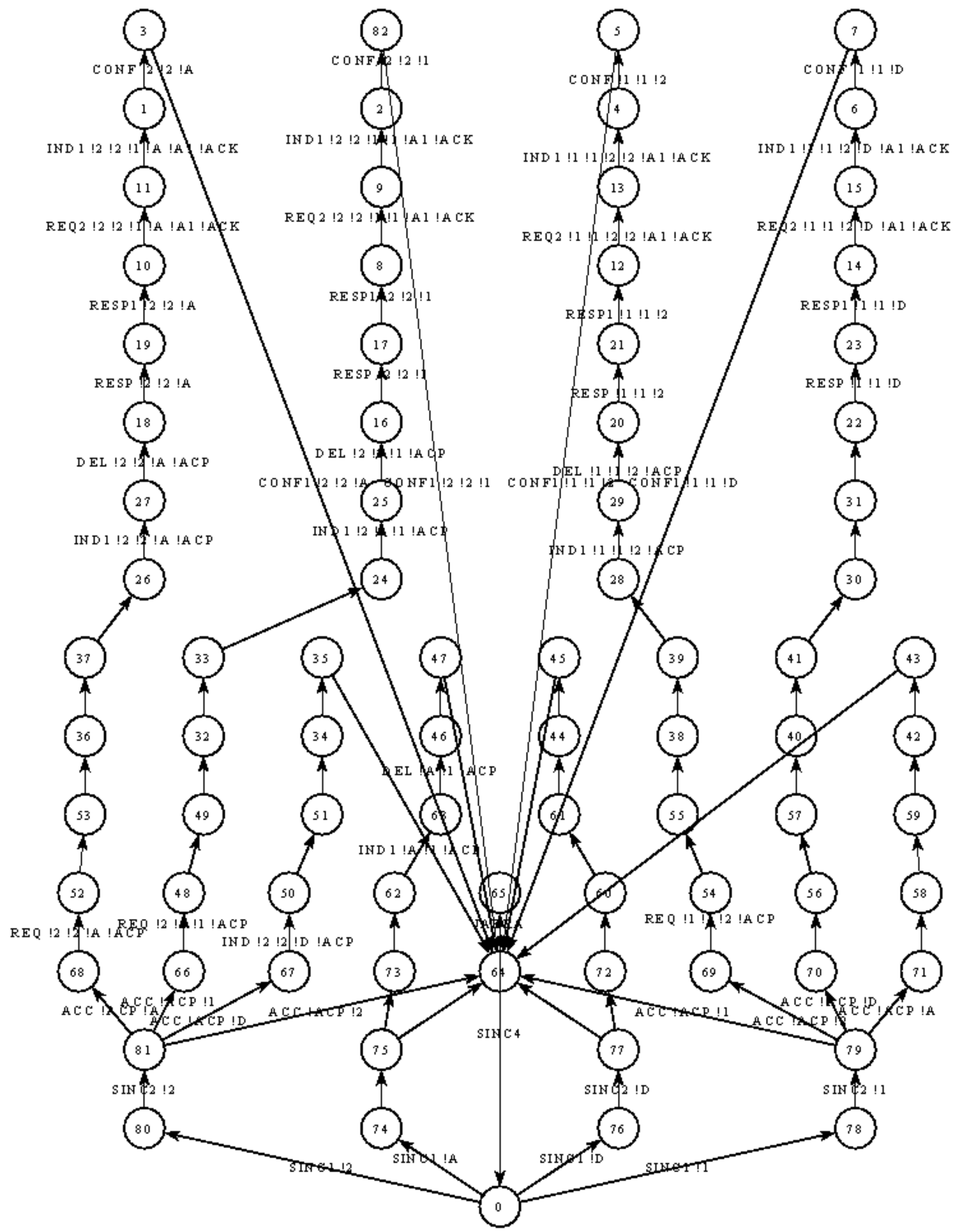

Figura 22 - Estados e Transições Convergentes para Duas Células, cujas Estações Líderes Podem se Comunicar com as Demais Estações da Rede

No décimo experimento, além de todas as possibilidades mencionadas nas simulações anteriores, as estações subordinadas podem iniciar a comunicação com as demais estações da rede por intermédio de seus respectivos líderes de célula.

Como resultado, o protocolo convergiu, porém a capacidade de tratamento de dados do software CADP não permitiu que o protocolo fosse minimizado para apresentação gráfica do resultado, os números de estados e transições foram de 4971199 e 7533706, respectivamente.

O CADP trabalha com estados e transições na ordem de dezenas de milhões, como a ferramenta de minimização gera estados e conseqüentemente transições redundantes, a partir dos existentes, para análise dos mesmos e minimizações por critérios de comparação, estes se tornaram aquém da capacidade do software. Contudo, todas as propriedades do item 3.6.3.1 foram testadas e aprovadas, assim o protocolo era convergente, vivo, reinicializável e isento de *deadlocks*.

4.4.2. Resumo dos Experimentos e Análise dos Resultados Obtidos

A tabela 05 apresenta as diversas fases do desenvolvimento do protocolo LANMAR em especificação formal, através da linguagem LOTOS, cujos dados estão consolidados resumidamente para um entendimento geral da evolução do experimento.

Segue uma sucinta descrição desta tabela para o melhor entendimento do leitor:

-**Simulação** se refere ao protocolo ou ao serviço simulado.

-**Número de nós** se refere ao número total de nós desta, quer sejam emissores, ou receptores, ou ambos, ao mesmo tempo.

-**Quantidade de nós emissores** contabiliza aqueles que tem como propriedade iniciar uma troca de mensagens;

-**Quantidade de nós receptores** contabiliza aqueles que respondem a uma solicitação de troca de mensagens;

-**Número de Estados** faz menção ao número de estados gerado pela ferramenta responsável pela obtenção do grafo LTS;

-**Número de Transições** faz menção ao número de transições gerado pela ferramenta responsável pela obtenção do grafo LTS;

-O valor **minimizado** indica que o grafo LTS passou por um filtro que retirou seus estados e transições redundantes.

Item	Simulação	Número de Nós	Quantidade de nós emissores	Quantidade de nós receptores	Número de Estados	Número de Transições	Minimi-zado
01	Primeira	2	2	2	1060	3530	Não
02	Primeira	2	2	2	58	134	Sim
03	Segunda	2	2	2	5824	16316	Não
04	Segunda	2	2	2	133	245	Sim
05	Terceira	3	2	3	15808	41065	Não
06	Terceira	3	2	3	186	338	Sim
07	Quarta	3	2	3	72548	402296	Não
08	Quarta	3	2	3	218	1008	Sim
09	Quinta	4	2	4	115075	1348800	Não
10	Quinta	4	2	4	114	650	Sim
11	Sexta	4	3	4	369550	3996215	Não
12	Sexta	4	3	4	259	1624	Sim
13	Sétima	4	4	4	322449	578232	Não
14	Sétima	4	4	4	52	67	Sim
15	Oitava	4	4	4	497477	887912	Não
16	Oitava	4	4	4	57	72	Sim
17	Nona	4	4	4	1924238	3040551	Não
18	Nona	4	4	4	83	98	Sim

19	Décima	4	4	4	4971199	7533706	Não
20	Décima	4	4	4	(1)	(1)	Sim
21	Serviço	X	X	X	6	12	Sim

Tabela 05 - Evolução do Experimento com o Protocolo LANMAR

(1) Estouro da capacidade de trabalho do software CADP.

Um breve histórico com os experimentos do protocolo LANMAR pode ser descrito da seguinte forma:

No primeiro experimento foram descritas apenas duas estações cujas estruturas de recepção e transmissão, dentro e fora das futuras células, foram implementadas, porém a maioria dos processos foi bloqueada para que houvesse uma observação inicial da mecânica de funcionamento das estações com o protocolo em questão. Havia uma estação apenas transmissora e a outra apenas receptora.

No experimento seguinte, as duas estações passaram a ser transmissoras e receptoras entre si e trocaram mensagens, validando esta parte inicial do protocolo.

No terceiro experimento foram descritas as estruturas de uma célula com uma estação subordinada e esta possuía apenas os processos de recepção de mensagens para o seu líder de célula.

No quarto experimento já existem duas células A1 e A2, onde LAM1 e LAM2 são seus líderes, respectivamente, com plenas possibilidades de transmissão e recepção, além disso, existe uma estação subordinada, A, que tem todas as estruturas de transmissão, mas ainda não transmite, é apenas receptora da líder da célula A1.

No quinto experimento existem duas células com seus respectivos líderes, além de duas estações subordinadas A e D, uma para cada célula, que têm todas as estruturas de transmissão e recepção, mas ainda não transmitem. A estação A é receptora da líder da célula A1 e a estação D é receptora da líder da célula A2.

No sexto experimento as duas células A1 e A2, onde LAM1 e LAM2 são líderes, as estações A e D tem todas as estruturas de transmissão e recepção. Apenas D ainda não transmite. A já transmite e recebe de seu líder de célula LAM1.

No sétimo experimento, para possibilitar a visualização do gráfico que até então era ininteligível, foi inserido o processo de sincronismo, o qual funcionou a contento e possibilitou uma análise mais qualitativa dos gráficos de resultados dos estados e transições.

No oitavo experimento o processo de sincronismo foi retirado do corpo do protocolo e inserido ao meio, como deveria de ser o controle de acesso ao meio, permitindo que as estações se comunicassem duas a duas.

No nono experimento as estações líderes de células transmitem e recebem para todas estações líderes da rede, para suas próprias estações subordinadas, diretamente, e se comunicam com as subordinadas de outras células por meio de seus respectivos líderes de células. As estações subordinadas apenas se comunicam com seus líderes de células, não podendo iniciar uma comunicação com as demais estações da rede.

No décimo experimento, além de todas as possibilidades mencionadas nas simulações anteriores, as estações subordinadas podem iniciar a comunicação com as demais estações da rede, através de seus respectivos líderes de célula.

Até este ponto todos os experimentos convergiram e foram comprovadas todas as propriedades descritas no item 3.6.3.1, provando que em todas as etapas do experimento, o protocolo era convergente, vivo, reinicializável e isento de *deadlock*s.

Houve ainda um décimo primeiro experimento, uma última simulação, com sete estações dentro de três células, distribuídas da seguinte forma:

-três líderes de células, um em cada célula;

-quatro estações subordinadas, duas em uma célula e as outras duas células com uma cada;

-todas as estações teriam todas as possibilidades mencionadas nas simulações anteriores.

Nesta fase do projeto o pacote de software CADP ficou simulando as condições acima descritas por oito dias, não houve capacidade de convergência porque os números de estados e transições superariam a ordem de dezenas de milhões, limite da ferramenta. Apesar disto, a convergência estaria garantida porque todas as características do protocolo já tinham sido simuladas em conjunto.

4.4.3. Conclusões sobre os Experimentos com o LANMAR

Cabe aqui uma análise dos resultados obtidos, com a especificação do protocolo LANMAR em LOTOS, onde os experimentos testaram várias possibilidades para rede em células, sempre com um incremento em complexidade desta rede, quer seja nas estações, quer seja nas células, variando entre cada experimento. Como resultado destes experimentos foi possível concluir que:

-o *draft* do IETF que descreve o protocolo LANMAR define a filosofia para a estruturação de uma rede em células com seus respectivos líderes;

-as estações residentes dentro destas células deveriam ter instalado protocolos pró-ativos modificados, tal como o FSR; para a troca de mensagens entre estações fora de uma mesma célula o *draft* sugere a utilização de protocolos pró-ativos modificados como o DSDV, OLSR e TBRPF;

-o *draft* não define a interligação destes protocolos e nem os pontos de interação entre estes.

Nesta pesquisa, através da especificação formal, utilizando a linguagem LOTOS, foi possível definir os pontos de instalação e interligação dos diversos protocolos, como foi mostrado pela figura 17.

Foi formalmente especificada a estrutura de formação da rede, utilizando o protocolo LANMAR. Com a utilização do protocolo reativo DSR para descoberta de rotas entre células, o qual já havia sido simulado, avaliado, modificado e validado.

O DSR permite ao LANMAR trabalhar com escalabilidade, porque propicia a inserção de novas estações e células à rede, o que não era permitido com protocolos puramente pró-ativos. Além disso, o LANMAR se torna um híbrido, aproveitando as melhores características de pró-ativo dentro das células, já que na arquitetura a qual ele se propõe, o movimento relativo entre os nós dentro desta é muito baixo, ou se pode reduzir as células a tamanhos que isto ocorra; e reativo entre as células, ou seja, apenas os líderes destas podem iniciar o protocolo DSR, o que faz com que a rede pareça ser bem menor.

Na descrição em LOTOS, o FSR e o DSR são processos dentro do LANMAR, que acionava um ou outro para atuar dentro ou fora da célula, respectivamente.

4.5. Comentários sobre os Experimentos

Neste capítulo foram mostrados todos os experimentos feitos nesta pesquisa, vale ressaltar que os resultados obtidos foram conseguidos passo a passo e muitas vezes eram necessárias várias simulações dentro de cada experimento, assim se chegou as modificações descritas em cada protocolo. Estas foram abordadas em vários experimentos, onde cada um deles simulava parte do mesmo problema e modificações eram refeitas diversas vezes, em várias versões, até se chegar a uma sólida abordagem daquele conhecimento, geralmente concretizado nos últimos experimentos.

As alterações descritas aqui são apenas frutos destes experimentos e poderão, a partir de novas experimentações e implementações, sofrerem outras modificações.

A evolução do experimento foi sempre em função de uma representação mais geral possível da rede, ou melhor, que representava a rede com mais realismo. As alterações dos protocolos sempre estiveram atreladas a esta evolução, exemplo disto, é a estrutura de sincronismo, que apenas se tornou necessária com o aumento da complexidade da rede, conseqüentemente um número muito elevado de estados e transições, irrepresentáveis mesmo após a aplicação da propriedade de minimização.

As estruturas de comunicação por inundação também só faziam sentido quando o número de estações excedia de duas.

Como mencionado anteriormente, os experimentos sempre conduziram a modificações após o crescimento em generalidade da representação da rede e por isto estão “pulverizados” ao longo dos vários experimentos, onde cada um deles representa partes da abordagem de um mesmo problema.

5. Conclusão

Os protocolos considerados foram analisados através de especificação formal para verificar e validar as exigências de funcionalidade para uma rede *ad-hoc*.

Foram salientados os aspectos e dificuldades para um bom desempenho dos protocolos de acesso ao meio e de roteamento para este tipo de rede. Nesta, as estações funcionam totalmente independentes umas das outras, sem gerenciamento centralizado, ou um banco de dados comum, não existem mais terminais fixos ou geograficamente localizados em uma única posição, o que existe é o conceito de estações móveis, dinamicamente mutáveis, com alto nível de deslocamentos para posições completamente aleatórias em intervalos de tempos indefiníveis.

As redes *ad-hoc* tiveram descrições genéricas para uma abordagem mais geral quanto possível. Sua estrutura foi mostrada fisicamente e sua arquitetura em camadas mostrou as ligações dos protocolos por meio de PDUs e SDUs.

Vários protocolos de acesso ao meio e de roteamento foram estudados, observando as diversas características de funcionamento existentes, inclusive os protocolos com abordagem probabilística, cujos resultados não garantiam sucesso na comunicação quando se levava em conta o problema do terminal escondido. Este problema está sempre presente em redes móveis, principalmente para aquelas sem previsibilidade na movimentação e no aparecimento de novas estações. Para solucionar este tipo de problema, utilizou-se o protocolo MACAW, com algumas modificações.

Os dez protocolos de roteamento existentes, atualmente, no IETF, foram estudados. A maioria pró-ativos, alguns reativos, uns com abordagens de roteamento por zona, outros por células, todos dependendo da característica predominante de cada rede.

Por adoção de características mais gerais possíveis para redes *ad-hoc*, optou-se por protocolos reativos, o DSR, e por divisão em células, abrangendo a maioria dos casos reais existentes.

Para arquiteturas em células foi definido o protocolo LANMAR, este tem sua eficiência ligada à topologia da rede, onde o movimento entre grupos é primordial para o seu melhor funcionamento em relação aos demais protocolos, ou seja, a célula representa um grupo onde a velocidade relativa entre as estações componentes deve ser baixa.

Em protocolos para redes *ad-hoc*, onde não há um responsável pela rede, um gerente de rede, o algoritmo é distribuído, o conjunto de estados e soluções tende para infinito. Utilizaram-se métodos formais porque estes são baseadas em princípios matemáticos, permitindo uma boa modelagem e posteriormente verificação, análise e validação de forma genérica, precisa e sem ambigüidades de todos os protocolos e das interligações existentes entre eles.

Apresentou-se uma proposta de analisar alguns dos mais conhecidos e utilizados protocolos de acesso ao meio e roteamento em redes móveis *ad-hoc*, com objetivo de futura implementação compondo uma arquitetura completa, em multi-camadas, desde o meio físico até o nível de aplicação. Em especial, apresentam-se as especificações formais em LOTOS e verificação de propriedades essenciais de funcionamento para os protocolos MACAW, DSR e LANMAR, o que mostrou as potencialidades da ferramenta utilizada, a viabilidade do experimento e a correção da especificação do protocolo frente ao modelo requerido para o serviço.

Como conclusões e contribuições podem ser destacadas as inclusões de processos como os de sincronismo, que auxiliam na diminuição dos números de estados e transições dos grafos do experimento em questão. Esta diminuição do volume de estados e transições auxilia numa análise qualitativa e quantitativa dos protocolos, facilitando a tomada de decisão sobre qual parte do protocolo se deve dar atenção e a necessidade de testar determinadas características que se queira colocar em evidência, quando da construção do protocolo.

Nos protocolos de acesso ao meio estudados foi escolhido o protocolo MACAW, através especificação formal, além das simulações feitas, foi possível observar que uma pequena alteração para possibilitar a transmissão por inundação seria necessária para agilizar e melhorar o desempenho deste protocolo. Esta modificação possibilitaria um melhor suporte a protocolos de roteamento do tipo reativo, que por sua característica intrínseca de fazer busca de rotas, *Route Request*, necessita fazer várias trocas de mensagens por inundação na rede.

Dentre os protocolos de roteamento pró-ativos se destaca o DSR, através de sua especificação formal foi possível concluir que o identificador utilizado por ele, para identificação das buscas de rotas, não é exatamente necessário nas mesmas. Isto porque toda estação ao receber o vetor de rotas, antes de qualquer atitude, deverá analisar se seu endereço já está contido neste, assim o identificador de rotas não é necessário na busca destas, devendo ser empregado pelas estações para indexar rotas redundantes em seu *Route Cache*.

Destaca-se também a implementação de estruturas de testes e decisões para implementação dos processos de busca, *Route Request*, e resposta, *Route Reply*, que foram construídas tendo como referência descrições sucintas dos *drafts* do IETF.

Ainda como contribuição, as ferramentas de especificação formal possibilitaram uma análise qualitativa, com indicativos do que se pode implementar em *software* e o que se deve implementar em *hardware* como, por exemplo, estruturas de comparação, acumulação e preenchimento de vetores de rotas que tem sua implementação em *software* facilitada. Porém estruturas de minimização, comparação, armazenamento indexado de rotas, bem como descarte de rotas com erros; devem ser, exclusivamente, implementadas em *hardware*, para diminuir o processamento dos nós nos instantes de buscas e trocas de mensagens, minimizando o trabalho das estações com roteamento em tempo real.

Para o protocolo LANMAR, foi feita uma substituição do protocolo pró-ativo entre células pelo DSR, afim de torná-lo mais eficiente para redes em células com uma maior dinâmica, como descritas no capítulo dois.

Quando a rede começa a receber novas estações, novas células e/ou os nós começam a se “desgarrar” das células, o LANMAR passa a ter desempenho inferior dentre todos os protocolos [54]. Além disso, os protocolos reativos são bem superiores aos pró-ativos nestas situações. Por estes motivos foi utilizado o protocolo DSR para fazer o roteamento entre as células, como o AODV, ele é essencialmente reativo, porém de mais simples implementação.

O DSR permite ao LANMAR trabalhar com escalabilidade, porque propicia a inserção de novas estações e células à rede, o que não era permitido com protocolos puramente pró-ativos. Além disso, o LANMAR se torna um híbrido, aproveitando as melhores características de pró-ativo dentro das células, já que na arquitetura a qual ele se propõe, o movimento relativo entre os nós dentro desta é muito baixo, ou se pode reduzir as células a tamanhos que isto ocorra; e reativo entre as células, ou seja, apenas os líderes destas podem iniciar o protocolo DSR, o que faz com que a rede pareça ser bem menor.

Todos os testes foram feitos levando em consideração a estrutura do DSR e a arquitetura em camadas das ligações dos protocolos por meio de PDUs e SDUs, contudo o AODV também poderá ser utilizado para testes, desde que a arquitetura em camadas seja devidamente manipulada para se adequar às características deste último protocolo.

Como passos futuros, seria necessária a especificação dos demais protocolos das camadas superiores a de roteamento, sendo descritas e submetidas à mesma abordagem empregada aos protocolos estudados, não só reavaliando a forma de verificação, mas em especial a própria modelagem da rede. Outras ferramentas de verificação de propriedades, relacionadas à lógica temporal podem ser utilizadas. Assim, será possível avaliar melhor a correção destes protocolos, ou mesmo sugerir alterações nos protocolos apresentados na literatura.

O CADP, em uma de suas ferramentas gera um arquivo em linguagem C, obtida por intermédio do código feito da especificação formal em LOTOS, seria interessante verificar se este arquivo tem sua implementação viável, ou poderá ser usado em alguma outra ferramenta de simulação como, por exemplo, o ns-2.

Dois trabalhos nesta mesma linha de pesquisa já estão sendo feitos por membros da equipe desta instituição, um deles será a especificação formal com testes de verificação e simulação com protocolos das camadas superiores, para validar protocolos atendendo a critérios de uma arquitetura para descoberta de serviços em redes *ad-hoc*. Este trabalho usará como base os protocolos validados aqui.

O outro trabalho será a implementação destes protocolos, principalmente das partes apontadas pelo método de especificação formal como sendo próprias para *hardware*, num projeto em micro-eletrônica.

Referências Bibliográficas

1. CORSON, S. and MACKER, J., "Mobile Ad-hoc Networking (MANET): Routing Protocol Performance and Issues and Evalation Considerations", *Internet RFC 2501,* (1999).

2. MCDONALD, A., ZNATI, T., "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks", *in Wireless Ad Hoc Networks. IEEE JSAC,* (Agosto 1999).

3. OBRACZKA, K., TSUDIK, G., "Multicast routing issues in ad hoc networks", *IEEE ICUPC,* October (1998).

4. PARK, V. D., CORSON M. S., "A Highly Adaptative Distributed Routing Algorithm for Mobile Wireless Networks", *INFOCOM (3*), páginas 1405-1413, (1997).

5. MÄÄTTÄ, R., "Wireless Ad Hoc Routing Protocols, a Taxonomy", *Defence Forces Research Institute of Technology Electronics and Information Technology Section,* (Maio 2000).

6. GRAY, B., "Soldiers, Agents and Wireless Networks: A Report on a Miliary Application", Dartmouth College, Hanover, (2000).

7. BLUETOOTH SIG, "Service discovery protocol (SDP)", (2000).

8. MALTZ, D. M., "Resource Management in Multi-hop Ad-Hoc Networks", *CMU-CS-00-150,* (Novembro 1999).

9. JOHANSSON, P., LARSSON, T., HEDMAN, N., et al., "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks", In Proceedings of ACM/IEEE MOBICOM'98, PP 195-206 , (1999).

10. MALTZ, D. A., BROCH, J., JONHSON, D. B., et al., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Networking Routing Protocols", In Proceedings of ACM/IEEE MOBICOM'98, dallas, TX, 1998, pp 85-97

11. ROZOVSKY, R., KUMAR, P. R., "Seedex: A MAC protocol for ad hoc networks", (2001).

12. BHARGHAVAN, V., SHENKER, S., ZHANG L., et al., "MACAW: A Media Access Protocol for Wireless {LAN}'s", *SIGCOMM*, páginas 212- 225, (1994).

13. MOURA, D. F. C., BAGATELLI, R., PEDROZA, A. DE C. P., "Uma Arquitetura de Suporte a Descoberta de Serviços em Redes Móveis Ad Hoc", Worksho de Métodos Formais, (Outubro 2002).

14. SOARES, L. F. G., LEMOS, G., COLCHER, S., "Rede de Computadores das LANs, MANS e WANs ás Redes ATM", 7ª *edição, editora CAMPUS*, (1995).

15. PEDROZA, A. de C. P., "Apostila Redes de Computadores, Arquiteturas e Projeto de Protocolos", *Dep. de Eletrônica, Prog. de Eng. Elétricas, COPPE , Universidade Federal do Rio de Janeiro,* (2000).

16. JOHNSON, D. B., MALTZ, D. A., HU Y.C., et al., "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", (Novembro 2001), http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-06.txt, (trabalho em andamento).

17. OGIER, R. G., TEMPLIM, F. L., BHARGAV, B., et al., "Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)", (Março 2002), http://www.ietf.org/internet-drafts/draft-ietf-manet-tbrpf-05.txt, (trabalho em andamento).

18. GERLA, M, HONG, X., PEI, G., "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks", (Junho 2002), http://www.ietf.org/internet-drafts/draft-ietf-manet-fsr-02.txt, (trabalho em andamento).

19. PERKINS, C. E., BELDING-ROYER, E. M., SAMIR R. D., "Ad hoc On-Demand Distance Vector (AODV) Routing", (19 de Janeiro 2002), http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-10.txt, (trabalho em andamento).

20. PERKINS, C. E., BHAGWAT, P., "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", (1995), http://www.cise.ufl.edu/~helal/6930F01/papers/DSDV.pdf

21. CLAUSEN T., JACQUET., P., LAOUITI, A., ET AL., " Optimized Link State Routing Protocol (OLSR)", (Setembro 2002), http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-06.txt, (trabalho em andamento).

22. HASS, Z. J., PEARLMAN, M. R., SAMAR, P., "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", (Janeiro 2003), http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-zrp-04.txt, (trabalho em andamento).

23. HASS, Z. J., PEARLMAN, M. R., SAMAR, P., "The Intrazone Routing Protocol (IARP) for Ad Hoc Networks", (Janeiro 2003), http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-iarp-02.txt, (trabalho em andamento).

24. HASS, Z. J., PEARLMAN, M. R., SAMAR, P., "The Interzone Routing Protocol (IERP) for Ad Hoc Networks", (Janeiro 2003), http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-ierp-02.txt, (trabalho em andamento).

25. HASS, Z. J., PEARLMAN, M. R., SAMAR, P., "The Bordercast Resolution Protocol (BRP) for Ad Hoc Networks", (Janeiro 2003), http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-brp-02.txt, (trabalho em andamento).

26. GERLA, M, HONG, X., PEI, G., et al., "Landmark Routing Protocol (LANMAR) for Large Scale Ad Hoc Networks", (Junho 2002), http://www.ietf.org/internet-drafts/draft-ietf-manet-lanmar-03.txt, (trabalho em andamento).

27. BROCH, J., MALTZ, D. A., JOHNSON, D. B., "Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks", Computer Science Department, Carnegie Mellon University, (1999), http://www.monarch.cs.cmu.edu/ .

28. MALTZ, D. A., BROCH, J., "Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Networks Tested", *CMU-CS-99-LL6*, (Março 1999).

29. JOHNSON, D. B., MALTZ, D. A., "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing, volume 353, publicado no Kluwer Academic Publishers*, (2001).

30. GUPTA, P., KUMAR, P., "The capacity of wireless network", *IEEE Transactions on Information Theory, IT-46:388-404, editor Imielinski and Korth,* (Março 2000).

31. KARN, P., "MACA - A New Channel Access Method for Packet Radio", *ARRL/CRRL Amateur Radio 9th Computer Network Conference",* (1990).

32. TANG, Z., GARCIA-L.-A. J. J., ”Hop Reservation Multiple Acess (HRMA) for Multichannel Packet Radio Networks”, *Computer Engineering Department School of Engeering,* Santa Cruz, CA, (1998).

33. TANG, Z., GARCIA-L.-A. J. J., ”Hop Reservation Multiple Acess (HRMA) for Ad-Hoc Networks”, *Computer Engineering Department School of Engeering, Santa Cruz, CA,* (1998).

34. DOS SANTOS, C. F., CÂMARA, D., LOUREIRO, A. A. F., "Uma Metodologia para Verificação Formal de Protocolos de Roteamento para Redes Ad hoc", *XIX Simpósio Brasileiro de Redes de Computadores SBRC'2001,* Florianópolis, SC, (2001).

35. THE INTERNET ENGINEERING TASK FORCE in group MOBILE AD-HOC NETWORKS (MANET), http://www.ietf.org/ids.by.wg/manet.html.

36. BRINKSMA, E., BOLOGNESI, T., "Introduction to the {ISO} Specification Language {LOTOS}", *Computer Networks and ISDN Systems,* volume 14, número 01, (1987).

37. ISO/IEC, "IS 8807: Information Processing Systems -- Open Systems Interconnection -- LOTOS -- A Formal Description Technique based on the Temporal Ordering of Observational Behaviour", Geneva, Switzerland, (1988).

38. EHRIG, H., MAHR, B., "Fundamentals of Algebraic Specifications", *livro Monographs on Theoretical Computer Science 1 (2),* volume 6 (21), editora Springer-Verlag, 1985 (1990).

39. MILNER, R, "Communication and Concurrency", *Prentice Hall,* (1989).

40. VASY - INRIA Rhône-Alpes. Case Studies Achieved using the CADP Toolset, Siège de l'INRIA (moyens d'accès), Domaine de Voluceau, Rocquencourt - B.P. 105, 78153 Le Chesnay Cedex - France, http://www.inrialpes.fr/vasy/cadp/case-studies/, (2001).

41. TANENBAUN, A. S., "Computer Networks", *3rd. ed. Pretince Hall*, New Jersey, (1996).

42. GERMEAU, F., LEDUC, G., "Model-based Design and Verification of Security Protocols using LOTOS", (1997).

43. GERMEAU, F., LEDUC, G., "Verification of Security Protocols Using LOTOS-method and Appplication", *Computer Communication 23*, páginas 1089-1103, Elsevier Science B. V. (2000), "http://www.elsevier.com/locate/comcom.

44. GARAVEL, H., HERMANNS, H., "On Combining Functional Verification and Performance Evaluation using CADP", *Théme, Rapport de Recherche,* número 4492, (Julho 2002).

45. TURNER, K. J., "The Formal Specification Language LOTOS, A Course for Users", *Department of Computing Science,* University of Striling, Scoland, (16 de abril de 1996).

46. GARAVEL, H., SIFAKIS, J., "Compilation and Verification of LOTOS Specifications", VERILOG Rhône-Alpes, (1990).

47. CADP (Caesar/Aldebaran Development Package): A Software Engineering Toolbox for Protocols and Distributed Systems,Version 1.163 last updated on 03/03/12 20:25:13, http://www.inrialpes.fr/vasy/cadp/.

48. EUCALYPTUS, University of Liège (Sart-Tilman Campus), Institut d'Electricité Montefiore (Parking P 32, Building B 28), B-4000, Liège 1, Belgium http://www.run.montefiore.ulg.ac.be/Projects/Presentation/index.php?project= Eucalyptus.

49. TCL/TK TOOL COMMAND LANGUAGE, INRIA Sophia-Antipolis, B.P. 93, 06902, Sophia-Antipolis CDX, FRANCE, http://www.tcl.tk/software/tcltk/.

50. APERO, Université de Liège, Institut Montefiore (B28), B-4000 Liège, BELGIUM, http://www.run.montefiore.ulg.ac.be/Projects/Presentation/Apero.

51. ELUDO, The UofO LOTOS Research Group, University of Ottawa, Canada, http://lotos.site.uottawa.ca/eludo/.

52. MEIJE PROJECT, INRIA Sophia-Antipolis, B.P. 93, 06902, Sophia-Antipolis CDX, FRANCE, http://www-sop.inria.fr/meije/verification/doc.html.

53. GARAVEL, H., “An Overview of the Eucalyptus Toolbox”, INRIA Rhône-Alpes/ VERIMAG, Zirst, 655, avenue de l’Europe, F-38330, Montbonnot Saint Martin, France, (1997).

54. PEI, G., GERLA, M., HONG, X., “LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility”, *Cumputer Science Department,* University of California, Los Angeles, CA 90095-1596, (2000).

Apêndice A - Especificação Formal em LOTOS do Protocolo DSR

```
specification DSR [ACC, phy_req3,  phy_ind3, phy_rep2,
phy_req1,phy_ind1,phy_req2,phy_ind2,dsr_req, dsr_conf,dsr_ind, dsr_resp,
sincro, jaera, DEL] : noexit

library

      Boolean, NaturalNumber, DSRLIBB


endlib

behaviour




(*hide phy_req3,  phy_ind3, phy_rep2,
phy_req1,phy_ind1,phy_req2,phy_ind2,dsr_req, dsr_conf,dsr_ind, dsr_resp,
sincro, jaera  in *)



      (
            (
            (((
      (*N*) User1[ACC, dsr_req, dsr_conf](ACP of DATA_TYPE, NOC of
NNO_TYPE)
            |||
            User11[dsr_ind, DEL, dsr_resp]
      (*O*) )
            |[dsr_req, dsr_conf, dsr_ind, dsr_resp]|
            (
            DSR1[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](A1 of ID_TYPE)
            |||
      (*1*) DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro, jaera](ESTE_NO_ESTA_DESATIVADO of
AVISO_TYPE)
            ))
            |||
            ((
      (*N*) User2[ACC, dsr_req, dsr_conf](ACP of DATA_TYPE, NOC of
NNO_TYPE)
            |||
            User22[dsr_ind, DEL, dsr_resp]
      (*O*) )
            |[dsr_req, dsr_conf, dsr_ind, dsr_resp]|
            (
            DSR2[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](A1 of ID_TYPE)
            |||
```

```
      (*2*) DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro, jaera](ESTE_NO_ESTA_DESATIVADO of
AVISO_TYPE)
            ))
            |||
            ((
      (*N*) User3[ACC, dsr_req, dsr_conf](ACP of DATA_TYPE, NOC of
NNO_TYPE)
            |||
            User33[dsr_ind, DEL, dsr_resp]
      (*O*) )
            |[dsr_req, dsr_conf, dsr_ind, dsr_resp]|
            (
            DSR3[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](A1 of ID_TYPE)
            |||
      (*3*) DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro, jaera](ESTE_NO_ESTA_DESATIVADO of
AVISO_TYPE)
            ))
            )
            |[sincro, jaera]|
            (Perdendo[sincro, jaera])
            )
            |[phy_req3,  phy_ind3, phy_rep2,phy_req1, phy_ind2, phy_req2,
phy_ind1]|
            MEDIUM[phy_req3,  phy_ind3, phy_rep2,phy_req1, phy_ind2,
phy_req2,  phy_ind1]
      )
 where

(* - Inicio da Descricao Completa do No 1 - *)

            process User1[ACC, dsr_req, dsr_conf](DATA:DATA_TYPE,
NNO:NNO_TYPE) : noexit :=
                        ACC !NOA ?DATA:DATA_TYPE ?NNO:NNO_TYPE;
                        dsr_req !NOA !DATA !NNO;
                        ((
                              dsr_conf !NOA !NNO;
                                    User1[ACC, dsr_req, dsr_conf](DATA,NNO)
                        )
                        []
                        (
                              dsr_conf !NOA !NNO ?AVISO:AVISO_TYPE;
                                    User1[ACC, dsr_req, dsr_conf](DATA,NNO)
                        ))
            endproc
            process User11[dsr_ind, DEL, dsr_resp] : noexit :=
                        dsr_ind !NOA ?DATA:DATA_TYPE;
                        DEL !NOA !DATA;
                        dsr_resp !NOA ;
                        User11[dsr_ind, DEL, dsr_resp]
            endproc
            process DSR1[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1,
phy_ind1, sincro,  jaera, dsr_conf] (ID:ID_TYPE): noexit:=
                        dsr_req !NOA ?DATA:DATA_TYPE ?NNO:NNO_TYPE;
```

```
                        RouteRequest1[phy_req3,  phy_ind3,
phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera, dsr_conf](DATA, NNO,
ID)
                        where
                              process RouteRequest1[phy_req3,  phy_ind3,
phy_rep2, dsr_req, phy_req1, phy_ind1, sincro,  jaera, dsr_conf]
(DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                    ((
                                    [NNO eq NOA]->(dsr_conf !NOA !NNO;
                                                      DSR1[phy_req3,
phy_ind3, phy_rep2, dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ID))
                                    )
                                    []
                                    (
                                    [NNO ne NOA]->(let NNOint1:NNO_TYPE =
NNO  in
                                                      RotReq1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](NNOint1, DATA, NNO, ID))
                                    ))
                                                where
                                                process RotReq1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](NNOint1:NNO_TYPE, DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) :
noexit :=
                                                phy_req3 !NOA !NNOint1 !ID;
                                                (( jaera !NOA !NNOint1 !ID;
                                                      RotReq1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1,phy_ind1, sincro,  jaera,
dsr_conf](NNOint1, DATA, NNO, ID)

                                                )

                                                []
                                                (phy_ind1 !NOA
?NNOint2:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                                                      (let NNO:NNO_TYPE =
NNOint1  in
                                                            dsr_conf !NOA
!NNO !AVISO;

      DSR1[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](ID)
                                                      )
                                                )
                                                []
                                                (phy_ind1 !NOA
?NNOint2:NNO_TYPE !NNOint1 !ID;
                                                      EnviaDados1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](DATA, NNO, NNOint2, NNOint1, ID)
                                                ))


                                    where
                                    process EnviaDados1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
```

```
dsr_conf](DATA:DATA_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, NNOint1:NNO_TYPE,
ID:ID_TYPE) : noexit :=
                                             phy_req1 !NOA !NNOint2 !NNOint1
!ID !DATA;
                                                    RECEBEACK1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ACK, DATA, NNO, NNOint2, NNOint1, ID)
                                                    WHERE
                                                    PROCESS RECEBEACK1[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ACK:ACK_TYPE, DATA:DATA_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE,
NNOint1:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                                           ((phy_ind1 !NOA
!NNOint2 !NNOint1 !ID !ACK;
                                                                  (let
NNO:NNO_TYPE =  NNOint1  in
                                                                        dsr_conf
!NOA !NNO;

      DSR1[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](ID)
                                                                  )
                                                           )
                                                           )
                                                    ENDPROC
                                             endproc
                                      endproc
                               endproc
            endproc
            process DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO:AVISO_TYPE) : noexit :=
                   (phy_ind2 ?NNO:NNO_TYPE !NOA ?NNOint1:NNO_TYPE
?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                         Repita_phy_req11[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](NNO, NNOint1, ID, AVISO)
                    )
                   []
                   DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                   []
                   (phy_ind2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                         (([NNOint1 eq NOA]->
                               RouteReplay1[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](AVISO, NNO, NNOint2, ID)
                         )
                         []
                         ([NNOint1 ne NOA]->     (([NNOint2 eq NOA]->
                                                           (sincro !NNO !NNOint1
!ID;
                                                                  DSR11[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO)
                                                           )
                                                    )
                                                    []
```

```
                                        ([NNOint2 ne NOA]->

    (Repita_phy_req1[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](NNO, NNOint2, NNOint1, ID,
AVISO)
                                                    )
                                              ))
                    ))
              )
              []
              (phy_ind2 ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                    (([NNOint1 ne NOA]->
                          (let NNOint2:NNO_TYPE = NOA in
                                (([NNOint2 eq NNO]->
                                                  (sincro !NNO !NNOint1
!ID;
                                                        DSR11[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO)
                                                  )
                                )
                                []
                                ([NNOint2 ne NNO]->
                                      Repita1 [phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, NNOint1, ID)
                                ))
                          )
                    )
                    []
                    ([NNOint1 eq NOA]->(let NNOint2:NNO_TYPE = NOA in
                          RouteReplay1[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, ID)
                          )
                    ))
              )
              where
              process RouteReplay1[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) :
noexit :=
                    phy_rep2 !NNO !NNOint2 !NOA !ID;
                    RecebeDados1[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, ID)
                          where
                          process RecebeDados1[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                (
                                (
                                phy_ind2 !NNO !NNOint2 !NOA !ID
?DATA:DATA_TYPE;
                                      dsr_ind !NOA !DATA;
                                            dsr_resp !NOA;
```

```
                                                          EnviaAck1[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO, NNO, NNOint2, ID, ACK)
                                    ))
                                                                where
                                                                process
EnviaAck1[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind,
dsr_resp, phy_ind1, sincro,  jaera] (AVISO:AVISO_TYPE, NNO:NNO_TYPE,
NNOint2:NNO_TYPE, ID:ID_TYPE, ACK:ACK_TYPE): noexit :=
                                                                      phy_req2
!NNO !NNOint2 !NOA !ID !ACK;
                                                                      (

      DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind,
dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                                                                      )
                                                                endproc
                              endproc
                  endproc
            endproc
            process Repita_phy_req1[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](NNO:NNO_TYPE,
NNOint2:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE, AVISO:AVISO_TYPE): noexit
:=
                  phy_req2 !NOA !NNOint2 !NNOint1 !ID !AVISO;
                  (
                  (
                        DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO)
                  ))

            endproc
            process Repita_phy_req11[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(NNO:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE, AVISO:AVISO_TYPE): noexit :=
                  phy_req3 !NNO !NOA !NNOint1 !ID !AVISO;
                  (
                  (
                        DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO)
                  ))
            endproc
            process Repita1 [phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO:AVISO_TYPE,
NNO:NNO_TYPE, NNOint2:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE): noexit :=
                  phy_req2 !NNO !NNOint2 !NNOint1 !ID;
                        (
                        (
                              DSR11[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                        ))
            endproc
(* - Fim da Descricao Completa do No 1 - *)


(* - Inicio da Descricao Completa do No 2 - *)
```

```
            process User2[ACC, dsr_req, dsr_conf](DATA:DATA_TYPE,
NNO:NNO_TYPE) : noexit :=
                        ACC !NOB ?DATA:DATA_TYPE ?NNO:NNO_TYPE;
                        dsr_req !NOB !DATA !NNO;
                        ((
                              dsr_conf !NOB !NNO;
                                    User2[ACC, dsr_req, dsr_conf](DATA,NNO)
                        )
                        []
                        (
                              dsr_conf !NOB !NNO ?AVISO:AVISO_TYPE;
                                    User2[ACC, dsr_req, dsr_conf](DATA,NNO)
                        ))
            endproc
            process User22[dsr_ind, DEL, dsr_resp] : noexit :=
                        dsr_ind !NOB ?DATA:DATA_TYPE;
                        DEL !NOB !DATA;
                        dsr_resp !NOB ;
                        User22[dsr_ind, DEL, dsr_resp]
            endproc
            process DSR2[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1,
phy_ind1, sincro,  jaera, dsr_conf] (ID:ID_TYPE): noexit:=
                        dsr_req !NOB ?DATA:DATA_TYPE ?NNO:NNO_TYPE;
                        RouteRequest2[phy_req3,  phy_ind3,
phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera, dsr_conf](DATA, NNO,
ID)
                        where
                              process RouteRequest2[phy_req3,  phy_ind3,
phy_rep2, dsr_req, phy_req1, phy_ind1, sincro,  jaera, dsr_conf]
(DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                    ((
                                    [NNO eq NOB]->(dsr_conf !NOB !NNO;
                                                      DSR2[phy_req3,
phy_ind3, phy_rep2, dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ID))
                                    )
                                    []
                                    (
                                    [NNO ne NOB]->(let NNOint1:NNO_TYPE =
NNO  in
                                                      RotReq2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](NNOint1, DATA, NNO, ID))
                                    ))
                                                where
                                                process RotReq2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](NNOint1:NNO_TYPE, DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) :
noexit :=
                                                phy_req3 !NOB !NNOint1 !ID;
                                                (( jaera !NOB !NNOint1 !ID;
                                                      RotReq2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1,phy_ind1, sincro,  jaera,
dsr_conf](NNOint1, DATA, NNO, ID)

                                                )
```

```
                                                        []
                                                        (phy_ind1 !NOB
?NNOint2:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                                                               (let NNO:NNO_TYPE =
NNOint1  in
                                                                      dsr_conf !NOB
!NNO !AVISO;

      DSR2[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](ID)
                                                               )
                                                        )
                                                        []
                                                        (phy_ind1 !NOB
?NNOint2:NNO_TYPE !NNOint1 !ID;
                                                               EnviaDados2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](DATA, NNO, NNOint2, NNOint1, ID)
                                                        ))


                                          where
                                          process EnviaDados2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](DATA:DATA_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, NNOint1:NNO_TYPE,
ID:ID_TYPE) : noexit :=
                                                 phy_req1 !NOB !NNOint2 !NNOint1
!ID !DATA;
                                                        RECEBEACK2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ACK, DATA, NNO, NNOint2, NNOint1, ID)
                                                        WHERE
                                                        PROCESS RECEBEACK2[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ACK:ACK_TYPE, DATA:DATA_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE,
NNOint1:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                                               ((phy_ind1 !NOB
!NNOint2 !NNOint1 !ID !ACK;
                                                                      (let
NNO:NNO_TYPE =  NNOint1  in
                                                                             dsr_conf
!NOB !NNO;

      DSR2[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](ID)
                                                                      )
                                                               )
                                                               )
                                                        ENDPROC
                                                 endproc
                                          endproc
                                   endproc
             endproc
             process DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO:AVISO_TYPE) : noexit :=
                    (phy_ind2 ?NNO:NNO_TYPE !NOB ?NNOint1:NNO_TYPE
?ID:ID_TYPE ?AVISO:AVISO_TYPE;
```

```
                        Repita_phy_req22[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](NNO, NNOint1, ID, AVISO)
                   )
                  []
                  DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                  []
                  (phy_ind2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                        (([NNOint1 eq NOB]->
                              RouteReplay2[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](AVISO, NNO, NNOint2, ID)
                        )
                        []
                        ([NNOint1 ne NOB]->     (([NNOint2 eq NOB]->
                                                      (sincro !NNO !NNOint1
!ID;
                                                            DSR22[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO)
                                                      )
                                                )
                                                []
                                                ([NNOint2 ne NOB]->

      (Repita_phy_req2[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](NNO, NNOint2, NNOint1, ID,
AVISO)
                                                      )
                                                ))
                        ))
                  )
                  []
                  (phy_ind2 ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                        (([NNOint1 ne NOB]->
                              (let NNOint2:NNO_TYPE = NOB in
                                    (([NNOint2 eq NNO]->
                                                      (sincro !NNO !NNOint1
!ID;
                                                            DSR22[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO)
                                                      )
                                    )
                                    []
                                    ([NNOint2 ne NNO]->
                                          Repita2 [phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, NNOint1, ID)
                                    ))
                              )
                        )
                        []
                        ([NNOint1 eq NOB]->(let NNOint2:NNO_TYPE = NOB in
```

```
                              RouteReplay2[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, ID)
                              )
                        ))
                  )
                  where
                  process RouteReplay2[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) :
noexit :=
                        phy_rep2 !NNO !NNOint2 !NOB !ID;
                        RecebeDados2[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, ID)
                              where
                              process RecebeDados2[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                    (
                                    (
                                    phy_ind2 !NNO !NNOint2 !NOB !ID
?DATA:DATA_TYPE;
                                          dsr_ind !NOB !DATA;
                                                dsr_resp !NOB;
                                                      EnviaAck2[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO, NNO, NNOint2, ID, ACK)
                                    ))
                                                            where
                                                            process
EnviaAck2[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind,
dsr_resp, phy_ind1, sincro,  jaera] (AVISO:AVISO_TYPE, NNO:NNO_TYPE,
NNOint2:NNO_TYPE, ID:ID_TYPE, ACK:ACK_TYPE): noexit :=
                                                                  phy_req2
!NNO !NNOint2 !NOB !ID !ACK;
                                                                  (

      DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind,
dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                                                                  )
                                                            endproc
                              endproc
                  endproc
            endproc
            process Repita_phy_req2[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](NNO:NNO_TYPE,
NNOint2:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE, AVISO:AVISO_TYPE): noexit
:=
                  phy_req2 !NOB !NNOint2 !NNOint1 !ID !AVISO;
                  (
                  (
                        DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO)
                  ))

            endproc
```

```
            process Repita_phy_req22[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(NNO:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE, AVISO:AVISO_TYPE): noexit :=
                  phy_req3 !NNO !NOB !NNOint1 !ID !AVISO;
                  (
                  (
                        DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO)
                  ))
            endproc
            process Repita2 [phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO:AVISO_TYPE,
NNO:NNO_TYPE, NNOint2:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE): noexit :=
                  phy_req2 !NNO !NNOint2 !NNOint1 !ID;
                        (
                        (
                              DSR22[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                        ))
            endproc


(* - Fim da Descricao Completa do No 2 - *)



(* - Inicio da Descricao Completa do No 3 - *)




            process User3[ACC, dsr_req, dsr_conf](DATA:DATA_TYPE,
NNO:NNO_TYPE) : noexit :=
                        ACC !NOC ?DATA:DATA_TYPE ?NNO:NNO_TYPE;
                        dsr_req !NOC !DATA !NNO;
                        ((
                              dsr_conf !NOC !NNO;
                                    User3[ACC, dsr_req, dsr_conf](DATA,NNO)
                        )
                        []
                        (
                              dsr_conf !NOC !NNO ?AVISO:AVISO_TYPE;
                                    User3[ACC, dsr_req, dsr_conf](DATA,NNO)
                        ))
            endproc
            process User33[dsr_ind, DEL, dsr_resp]  : noexit :=
                        dsr_ind !NOC ?DATA:DATA_TYPE;
                        DEL !NOC !DATA;
                        dsr_resp !NOC ;
                        User33[dsr_ind, DEL, dsr_resp]
            endproc
            process DSR3[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1,
phy_ind1, sincro,  jaera, dsr_conf] (ID:ID_TYPE): noexit:=
                        dsr_req !NOC ?DATA:DATA_TYPE ?NNO:NNO_TYPE;
                        RouteRequest3[phy_req3,  phy_ind3,
phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera, dsr_conf](DATA, NNO,
ID)
                        where
```

```
                              process RouteRequest3[phy_req3,  phy_ind3,
phy_rep2, dsr_req, phy_req1, phy_ind1, sincro,  jaera, dsr_conf]
(DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                    ((
                                    [NNO eq NOC]->(dsr_conf !NOC !NNO;
                                                      DSR3[phy_req3,
phy_ind3, phy_rep2, dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ID))
                                    )
                                    []
                                    (
                                    [NNO ne NOC]->(let NNOint1:NNO_TYPE =
NNO  in
                                                      RotReq3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](NNOint1, DATA, NNO, ID))
                                    ))
                                                where
                                                process RotReq3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](NNOint1:NNO_TYPE, DATA:DATA_TYPE, NNO:NNO_TYPE, ID:ID_TYPE) :
noexit :=
                                                phy_req3 !NOC !NNOint1 !ID;
                                                (( jaera !NOC !NNOint1 !ID;
                                                      RotReq3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1,phy_ind1, sincro,  jaera,
dsr_conf](NNOint1, DATA, NNO, ID)

                                                )


                                                []
                                                (phy_ind1 !NOC
?NNOint2:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                                                      (let NNO:NNO_TYPE =
NNOint1  in
                                                            dsr_conf !NOC
!NNO !AVISO;

      DSR3[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](ID)
                                                      )
                                                )
                                                []
                                                (phy_ind1 !NOC
?NNOint2:NNO_TYPE !NNOint1 !ID;
                                                      EnviaDados3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](DATA, NNO, NNOint2, NNOint1, ID)
                                                ))


                                    where
                                    process EnviaDados3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](DATA:DATA_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, NNOint1:NNO_TYPE,
ID:ID_TYPE) : noexit :=
                                          phy_req1 !NOC !NNOint2 !NNOint1
!ID !DATA;
```

```
                                                RECEBEACK3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ACK, DATA, NNO, NNOint2, NNOint1, ID)
                                                WHERE
                                                PROCESS RECEBEACK3[phy_req3,
phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1, sincro,  jaera,
dsr_conf](ACK:ACK_TYPE, DATA:DATA_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE,
NNOint1:NNO_TYPE, ID:ID_TYPE) : noexit :=
                                                      ((phy_ind1 !NOC
!NNOint2 !NNOint1 !ID !ACK;
                                                            (let
NNO:NNO_TYPE =  NNOint1  in
                                                                  dsr_conf
!NOC !NNO;

      DSR3[phy_req3,  phy_ind3, phy_rep2,dsr_req, phy_req1, phy_ind1,
sincro,  jaera, dsr_conf](ID)
                                                            )
                                                      )
                                                      )
                                                ENDPROC
                                          endproc
                                    endproc
                              endproc
            endproc
            process DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO:AVISO_TYPE) : noexit :=
                  (phy_ind2 ?NNO:NNO_TYPE !NOC ?NNOint1:NNO_TYPE
?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                        Repita_phy_req33[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](NNO, NNOint1, ID, AVISO)
                  )
                  []
                  DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                  []
                  (phy_ind2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                        (([NNOint1 eq NOC]->
                              RouteReplay3[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](AVISO, NNO, NNOint2, ID)
                        )
                        []
                        ([NNOint1 ne NOC]->     (([NNOint2 eq NOC]->
                                                      (sincro !NNO !NNOint1
!ID;
                                                            DSR33[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO)
                                                      )
                                                )
                                                []
                                                ([NNOint2 ne NOC]->

      (Repita_phy_req3[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2,
```

```
dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](NNO, NNOint2, NNOint1, ID,
AVISO)
                                                     )
                                              ))
                    ))
               )
               []
               (phy_ind2 ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                    (([NNOint1 ne NOC]->
                         (let NNOint2:NNO_TYPE = NOC in
                              (([NNOint2 eq NNO]->
                                                     (sincro !NNO !NNOint1
!ID;
                                                         DSR33[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO)
                                                     )
                              )
                              []
                              ([NNOint2 ne NNO]->
                                   Repita3 [phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, NNOint1, ID)
                              ))
                         )
                    )
                    []
                    ([NNOint1 eq NOC]->(let NNOint2:NNO_TYPE = NOC in
                         RouteReplay3[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, ID)
                         )
                    ))
               )
               where
               process RouteReplay3[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera](AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) :
noexit :=
                    phy_rep2 !NNO !NNOint2 !NOC !ID;
                    RecebeDados3[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO, NNO, NNOint2, ID)
                         where
                         process RecebeDados3[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(AVISO:AVISO_TYPE, NNO:NNO_TYPE, NNOint2:NNO_TYPE, ID:ID_TYPE) : noexit :=
                              (
                              (
                              phy_ind2 !NNO !NNOint2 !NOC !ID
?DATA:DATA_TYPE;
                                   dsr_ind !NOC !DATA;
                                        dsr_resp !NOC;
                                             EnviaAck3[phy_req3,
phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,
jaera] (AVISO, NNO, NNOint2, ID, ACK)
                              ))
                                                       where
```

```
                                                           process
EnviaAck3[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind,
dsr_resp, phy_ind1, sincro,  jaera] (AVISO:AVISO_TYPE, NNO:NNO_TYPE,
NNOint2:NNO_TYPE, ID:ID_TYPE, ACK:ACK_TYPE): noexit :=
                                                                 phy_req2
!NNO !NNOint2 !NOC !ID !ACK;
                                                                 (

      DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2, phy_req2, dsr_ind,
dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                                                                 )
                                                           endproc
                              endproc
                  endproc
            endproc
            process Repita_phy_req3[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](NNO:NNO_TYPE,
NNOint2:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE, AVISO:AVISO_TYPE): noexit
:=
                  phy_req2 !NOC !NNOint2 !NNOint1 !ID !AVISO;
                  ((
                        DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO)
                  ))

            endproc
            process Repita_phy_req33[phy_req3,  phy_ind3,
phy_rep2,phy_ind2, phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera]
(NNO:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE, AVISO:AVISO_TYPE): noexit :=
                  phy_req3 !NNO !NOC !NNOint1 !ID !AVISO;
                  ((
                        DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera](AVISO)
                  ))
            endproc
            process Repita3 [phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO:AVISO_TYPE,
NNO:NNO_TYPE, NNOint2:NNO_TYPE, NNOint1:NNO_TYPE, ID:ID_TYPE): noexit :=
                  phy_req2 !NNO !NNOint2 !NNOint1 !ID;
                        ((
                              DSR33[phy_req3,  phy_ind3, phy_rep2,phy_ind2,
phy_req2, dsr_ind, dsr_resp, phy_ind1, sincro,  jaera] (AVISO)
                        ))
            endproc

(* - Fim da Descricao Completa do No 3 - *)


            process Perdendo [sincro, jaera]:noexit :=
                  sincro ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                        jaera !NNO !NNOint1 !ID;
                              Perdendo[sincro, jaera]
            endproc
            process Medium[phy_req3,  phy_ind3, phy_rep2, phy_req1,
phy_ind2, phy_req2,  phy_ind1] : noexit :=
                        (
                        phy_req3 ?NNO:NNO_TYPE ?NNOint1:NNO_TYPE
?ID:ID_TYPE;
```

```
                              ((    phy_ind2 !NNO !NNOint1 !ID;
                                    Medium[phy_req3,  phy_ind3,
phy_rep2,phy_req1, phy_ind2, phy_req2,  phy_ind1]

                        )     ))
                        []
                        (phy_req2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                              ((phy_ind2 !NNO !NNOint2 !NNOint1 !ID;
                                    Medium[phy_req3,  phy_ind3, phy_rep2,
phy_req1, phy_ind2, phy_req2,  phy_ind1]
                        )     ))
                        []
                        (phy_rep2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE;
                              ((phy_ind1 !NNO !NNOint2 !NNOint1 !ID;
                                    Medium[phy_req3,  phy_ind3, phy_rep2,
phy_req1, phy_ind2, phy_req2, phy_ind1   ]
                        )     ))
                        []
                        (
                        phy_req1 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE ?DATA:DATA_TYPE;
                              ((phy_ind2 !NNO !NNOint2! NNOint1 !ID !DATA;
                                    Medium[phy_req3,  phy_ind3, phy_rep2,
phy_req1, phy_ind2, phy_req2, phy_ind1   ]
                        )     ))
                        []
                        (phy_req2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE ?ACK:ACK_TYPE;
                              ((phy_ind1 !NNO !NNOint2 !NNOint1 !ID !ACK;
                                    Medium[phy_req3,  phy_ind3, phy_rep2,
phy_req1, phy_ind2, phy_req2, phy_ind1   ]
                        )     ))
                        []
                        Medium[phy_req3,  phy_ind3, phy_rep2, phy_req1,
phy_ind2, phy_req2, phy_ind1   ]
                        []
                        (phy_req2 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                              ((phy_ind2 !NNO !NNOint2 !NNOint1 !ID !AVISO;
                                    Medium[phy_req3,  phy_ind3, phy_rep2,
phy_req1, phy_ind2, phy_req2, phy_ind1   ]
                        )     ))
                        []
                        (phy_req3 ?NNO:NNO_TYPE ?NNOint2:NNO_TYPE
?NNOint1:NNO_TYPE ?ID:ID_TYPE ?AVISO:AVISO_TYPE;
                              ((phy_ind1 !NNO !NNOint2 !NNOint1 !ID !AVISO;
                                    Medium[phy_req3,  phy_ind3, phy_rep2,
phy_req1, phy_ind2, phy_req2, phy_ind1   ]
                              ))
                        )
            endproc
endspec
```

Apêndice B - Biblioteca em LOTOS dos Parâmetros do Protocolo DSR

```
(* Obs.1 : Jamais colocar como saída de um operador um sort que seja
renomeação de outro sort *)

(* Obs.2 : Seja O1 um operador definido em um sort S1. Não se deve inserir
o comentário de construção em O1 se sua saida nao for S1 *)

(* Obs.3 : Seja O1 um operador definido em um sort S1. Deve-se definir os
construtores para operadores que possuírem saída igual a S1, desde que não
sejam da forma S1->S1 *)


  type NNO is Boolean, NaturalNumber
    sorts NNO_TYPE(*! implementedby NNOTYPE comparedby CMP_NNOTYPE
                 enumeratedby ENUM_NNOTYPE printedby PRINT_NNOTYPE *)

    opns

      NOA (*! implementedby NOA constructor *),
      NOZ (*! implementedby NOZ constructor *),
      NOB (*! implementedby NOB constructor *),
      NOC (*! implementedby NOC constructor *):-> NNO_TYPE
      _eq_ (*! implementedby EQ_NNOTYPE *),
      _ne_ (*! implementedby NEQ_NNOTYPE  *): NNO_TYPE, NNO_TYPE -> Bool

      eqns forall a,b : NNO_TYPE

  ofsort Bool
    a=b => a eq b = true;
    a eq b = false; (* dans les autres cas *)

    a=b => a ne b = false;
    a ne b = true;


  endtype

  type ACK is


    sorts  ACK_TYPE  (*! implementedby  ACKTYPE comparedby CMP_ACKTYPE

                 printedby PRINT_ACKTYPE *)

    opns

      ACK (*! implementedby  ACK constructor *): ->  ACK_TYPE


  endtype
```

```
 type PERDA is

    sorts  PERDA_TYPE    (*! implementedby  PERDATYPE   comparedby
CMP_PERDATYPE

                     printedby PRINTPERDA *)

    opns

      perdido2 (*! implementedby  perdido2 constructor *),
      perdido3 (*! implementedby  perdido3 constructor *): ->  PERDA_TYPE

  endtype

  type ID is  Boolean, NaturalNumber

    sorts ID_TYPE  (*! implementedby IDTYPE comparedby CMP_IDTYPE
                     enumeratedby ENUM_IDTYPE printedby PRINT_IDTYPE *)
      opns

      A1 (*! implementedby A1 constructor *):-> ID_TYPE


  endtype

      type AVISO is

    sorts AVISO_TYPE  (*! implementedby AVISOTYPE comparedby CMP_AVISOTYPE

                       printedby PRINT_AVISOTYPE *)

    opns

      ESTE_NO_ESTA_DESATIVADO (*! implementedby ESTE_NO_ESTA_DESATIVADO
constructor *):-> AVISO_TYPE

  endtype

  type DATA is

    sorts DATA_TYPE  (*! implementedby DATATYPE comparedby CMP_DATATYPE

                      enumeratedby ENUM_DATATYPE printedby PRINT_DATATYPE
*)

    opns

      ACP (*! implementedby ACP constructor *):-> DATA_TYPE

endtype
```