

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

**Especificação e Validação dos Mecanismos de Sincronização em  
Sistemas Multimídia Distribuídos, Utilizando a Linguagem Esterel**

Dissertação a ser submetida à Universidade Federal de Santa Catarina

para a obtenção do grau de

**Mestre em Engenharia Elétrica**

*José Miguel Eyzell González*

Florianópolis, 8 de Abril de 1996.

**Especificação e Validação dos Mecanismos de Sincronização  
em Sistemas Multimídia Distribuídos,  
Utilizando a Linguagem Esterel**

**José Miguel Eyzell González**

Esta dissertação foi julgada para a obtenção do título de

**Mestre em Engenharia**

especialidade **Engenharia Elétrica**,

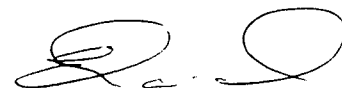
área de concentração **Sistemas de Controle, Automação e Informática Industrial**,

e aprovada em sua forma final pelo Curso de Pós-Graduação.



Prof. Jean Marie Farines, Dr. Ing.

Orientador



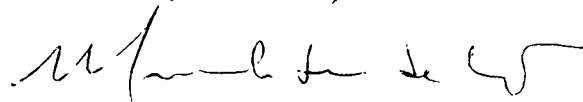
Prof. Enio Valmor Kassick, Dr.

Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

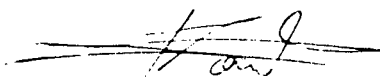
**Banca Examinadora:**



Prof. Jean Marie Farines, Dr. Ing.



Prof. Murilo Silva de Camargo, Dr.



Prof. Wanderley Lopes de Souza, Dr. Ing.

*A mis padres Carmen e Miguel Angel,  
a mi esposa Xiomara,  
a mis hijos Michael y Sailí,  
y a mis hermanos Miguel, Sonia, Carlos y Milagros.*

## Agradecimentos

Ao Prof. Dr. Ing. Jean-Marie Farines pela orientação, estímulo e apoio brindado durante todo o desenvolvimento deste trabalho.

Aos membros da banca examinadora pela aceitação de participação e pelos comentários e críticas construtivas.

Ao mestre Udo Fritzke Jr., por nossa amizade, e pelo enorme apoio, encorajamento e valiosas contribuições brindadas na realização deste trabalho.

Aos colegas do Laboratório de Controle e Microinformática (LCMI) da Universidade Federal de Santa Catarina (UFSC) que espontaneamente me ajudaram nos problemas do dia-a-dia.

À UFSC e à CAPES pelo suporte material e financeiro.

Agradeço especialmente à minha esposa Xiomara, a meu filho Michael e a minha filha Sailí, por todo o amor e alegria que juntos compartilhamos.

# Sumário

<b>Lista de Figuras .....</b>	<b>viii</b>
<b>Lista de Tabelas .....</b>	<b>x</b>
<b>Resumo.....</b>	<b>xi</b>
<b>Abstract.....</b>	<b>xii</b>
<b>Capítulo 1: Introdução Geral.....</b>	<b>1</b>
<b>Capítulo 2: Problemática da Sincronização em Sistemas Multimídia Distribuídos.....</b>	<b>4</b>
2.1 Introdução.....	4
2.2 Sistemas Multimídia .....	4
2.2.1 Mídia, Multimídia e Aplicações Multimídia .....	4
2.2.2 Requisitos das Aplicações Multimídia .....	5
2.2.3 Qualidade de Serviço nos Sistemas Multimídia.....	7
2.3 A Sincronização nos Sistemas Multimídia Distribuídos .....	9
2.3.1 Aspectos Gerais.....	9
2.3.2 Representação da Informação Multimídia: Definições Básicas.....	11
2.4 A Sincronização Intra-Fluxo .....	13
2.4.1 Abordagem Temporal .....	13
2.4.1.1 Sincronização de Fluxos Estocásticos .....	14
2.4.1.2 Sincronização de Fluxos Periódicos .....	17
2.4.2 Abordagem Temporal-Causal .....	21
2.5 A Sincronização Inter-Fluxo .....	23
2.5.1 Abordagem de Multiplexação .....	23
2.5.2 Abordagem Causal .....	24
2.5.2.1 Técnica das Marcas de Sincronização.....	24
2.5.2.2 Técnica do Canal de Sincronização .....	25

2.5.2.3 Técnica da Apresentação Condicional .....	25
2.5.3 Abordagem Temporal .....	26
2.5.3.1 Sincronização de Fluxos Estocásticos .....	27
2.5.3.2 Sincronização de Fluxos Periódicos .....	29
2.6 A Sincronização Multidestino .....	30
2.6.1 Sincronização de Fluxos Estocásticos .....	30
2.6.2 Sincronização de Fluxos Periódicos .....	32
2.7 Conclusão .....	33
<b>Capítulo 3: Projeto e Desenvolvimento de Sistemas Multimídia .....</b>	<b>34</b>
3.1 Introdução .....	34
3.2 As Técnicas de Descrição Formal (TDF) .....	34
3.3 O Fator Tempo-Real nos Sistemas Multimídia .....	35
3.4 Os Métodos Formais no Desenvolvimento de Sistemas Multimídia .....	36
3.5 Esterel como Linguagem de Desenvolvimento .....	37
3.5.1 Características Gerais de Esterel .....	37
3.5.2 Esterel como Linguagem de Desenvolvimento: Uma Justificativa da Escolha .....	38
3.5.3 O Ambiente de Desenvolvimento para a Linguagem Esterel .....	39
3.5.4 A Metodologia Proposta .....	40
3.6 Conclusão .....	41
<b>Capítulo 4: Modelagem em Esterel de Técnicas de Sincronização Multimídia .....</b>	<b>42</b>
4.1 Introdução .....	42
4.2 Modelagem de Técnicas de Sincronização Intra-Fluxo .....	42
4.2.1 Características Gerais do Problema: .....	42
4.2.2 Os Cenários de Sincronização Intra-Fluxo .....	43
4.2.2.1 Primeiro Cenário .....	44
4.2.2.2 Segundo Cenário .....	45
4.2.2.3 Terceiro Cenário .....	46
4.2.2.4 Quarto Cenário .....	47
4.2.2.5 Quinto Cenário .....	48
4.2.3 Modelagem dos Módulos Genéricos .....	50
4.2.3.1 O Modelo do Canal de Transmissão .....	50
4.2.3.2 O Modelo do Relógio .....	54
4.2.4 Modelagem dos Cenários de Sincronização Intra-Fluxo .....	54
4.2.4.1 O Modelo do Primeiro Cenário .....	54
4.2.4.2 O Modelo do Segundo Cenário .....	61

4.2.4.3 O Modelo do Terceiro Cenário .....	65
4.2.4.4 O Modelo do Quarto Cenário .....	68
4.2.4.5 O Modelo do Quinto Cenário.....	72
4.3 Modelagem de Técnicas de Sincronização Inter-Fluxo .....	81
4.3.1 O Cenário de Sincronização Inter-Fluxo .....	81
4.3.2 Modelagem do Cenário de Sincronização Inter-Fluxo .....	83
4.4 Modelagem de Técnicas de Sincronização Multidestino .....	85
4.4.1 O Cenário de Sincronização Multidestino .....	85
4.4.2 Modelagem do Cenário de Sincronização Multidestino .....	87
4.5 Conclusão .....	94
<b>Capítulo 5: Validação e Avaliação das Especificações .....</b>	<b>95</b>
5.1 Introdução.....	95
5.2 Simulação dos Cenários de Sincronização Intra-Fluxo .....	95
5.2.1 Aspectos Gerais da Simulação .....	95
5.2.2 Simulação do Primeiro Cenário.....	96
5.2.3 Simulação do Segundo Cenário .....	98
5.2.4 Simulação do Terceiro Cenário.....	100
5.2.5 Simulação do Quarto Cenário .....	102
5.2.6 Simulação do Quinto Cenário .....	106
5.3 Simulação do Cenário de Sincronização Inter-Fluxo .....	108
5.4 Simulação do Cenário de Sincronização Multidestino .....	109
5.5 Conclusão .....	112
<b>Capítulo 6: Considerações Finais .....</b>	<b>113</b>
<b>Referências Bibliográficas .....</b>	<b>115</b>
<b>Apêndice A: Elementos de Base da Linguagem Esterel .....</b>	<b>121</b>
<b>Apêndice B: Módulo de Sincronização “Sinc_Intra”.....</b>	<b>123</b>
<b>Apêndice C: Módulos Observadores.....</b>	<b>125</b>

## Lista de Figuras

Figura 2.1 Exemplos de Sincronização: (a) Intra-Fluxo, (b) Inter-Fluxo e (c) Multidestino. ....	10
Figura 2.2 Assinatura temporal de um fluxo. ....	12
Figura 2.3 Assinatura temporal de um feixe. ....	12
Figura 4.1 Apresentação remota de uma mídia contínua. ....	42
Figura 4.2 Apresentação de uma mídia contínua num sistema de relógios comuns. ....	44
Figura 4.3 Apresentação de uma mídia contínua num sistema de relógios a derivas comuns. ....	45
Figura 4.4 Apresentação de um fluxo periódico num sistema de relógios a derivas comuns. ....	46
Figura 4.5 Apresentação de um fluxo periódico num sistema de relógios a derivas constantes. ....	47
Figura 4.6 Apresentação de um fluxo periódico num sistema de relógios não sincronizados. ....	49
Figura 4.7 Estrutura do modelo do Canal de Transmissão. ....	50
Figura 4.8 Estrutura modular Esterel do primeiro cenário de sincronização intra-fluxo. ....	55
Figura 4.9 Estrutura do modelo do sincronizador do primeiro cenário de sincronização intra-fluxo. ....	57
Figura 4.10 Estrutura modular Esterel do segundo cenário de sincronização intra-fluxo. ....	61
Figura 4.11 Estrutura modular Esterel do terceiro cenário de sincronização intra-fluxo. ....	65
Figura 4.12 Estrutura do modelo do sincronizador do terceiro cenário de sincronização intra- fluxo. ....	66
Figura 4.13 Estrutura modular Esterel do quarto cenário de sincronização intra-fluxo. ....	69
Figura 4.14 Estrutura modular Esterel do quinto cenário de sincronização intra-fluxo. ....	72
Figura 4.15 Estrutura do modelo do servidor multimídia do quinto cenário de sincronização intra- fluxo. ....	73
Figura 4.16 Estrutura do modelo do midiafone do quinto cenário de sincronização intra-fluxo. ....	78
Figura 4.17 Sincronização de lábios num sistema de relógios comuns. ....	82
Figura 4.18 Estrutura modular Esterel do cenário de sincronização inter-fluxo. ....	83
Figura 4.19 Sincronização de lábios num sistema de relógios não sincronizados. ....	85
Figura 4.20 Estrutura modular Esterel do cenário de sincronização multidestino. ....	88
Figura 4.21 Estrutura do servidor multimídia do cenário de sincronização multidestino. ....	89
Figura 5.1 Dilatação de um fluxo sob o esquema do emulador de período. ....	104



Figura 5.2. Exemplo de compressão de um fluxo sob o esquema do emulador de período.....	105
Figura 5.3 Variação da assincronia de apresentação entre fluxos áudio e vídeo, em unidades de vídeo.....	111

## Lista de Tabelas

Tabela 2.1 Alguns valores de parâmetros de QoS [Dairaine94].....	9
Tabela 5.1 Resultados da simulação do esquema de sincronização intra-fluxo baseado em time-stamping, num sistema de relógios comuns. ....	97
Tabela 5.2 Resultados da primeira simulação do esquema de sincronização intra-fluxo baseado em time-stamping, num sistema de relógios a derivas comuns.....	99
Tabela 5.3 Resultados da segunda simulação do esquema de sincronização intra-fluxo baseado em time-stamping, num sistema de relógios a derivas comuns.....	100
Tabela 5.4 Resultados da simulação do esquema de sincronização intra-fluxo baseado na técnica do prefetch, num sistema de relógios a derivas comuns.....	101
Tabela 5.5 Resultados da primeira simulação da técnica do emulador de período, num sistema de relógios a derivas constantes.....	103
Tabela 5.6 Resultados da segunda simulação da técnica do emulador de período, num sistema de relógios a derivas constantes.....	105
Tabela 5.7 Períodos de realimentação correspondentes a distintos valores de capacidade de armazenamento. ....	107
Tabela 5.8 Respostas do simulador do esquema de sincronização intra-fluxo baseado em mensagens de realimentação. ....	107
Tabela 5.9 Valores de assincronia para diferentes períodos de realimentação, no esquema de sincronização multideestino baseado em mensagens de realimentação.....	111

## Resumo

Os Sistemas Multimídia são sistemas que permitem ao usuário gerar, tratar, armazenar, restituir, e apresentar de maneira integrada mídias tais como texto, gráficos, imagens, voz, áudio e vídeo. A sincronização destas atividades, respondendo a uma certa qualidade de serviço esperada pelo usuário, é uma necessidade primordial destes sistemas. São várias as propostas de sincronização multimídia encontradas na literatura, e cabe ressaltar que não existe uma solução genérica para todos os problemas de sincronização.

Este trabalho apresenta um esforço orientado à concepção formal de sistemas multimídia distribuídos utilizando a linguagem de especificação Esterel. Em particular, utiliza-se uma metodologia de desenvolvimento baseada em Esterel, seguindo o lema WYPIWYE (*What You Prove Is What You Execute*) [Berry88]. Vários cenários de sincronização são estabelecidos, reunindo diferentes requisitos e restrições de sincronização multimídia. A modelagem de cada cenário é conseguida utilizando algumas das principais técnicas de sincronização multimídia propostas na literatura. A validação e avaliação destes modelos é realizada através do processo de simulação. Finalmente são discutidas as vantagens e limitações da metodologia utilizada neste trabalho.

O trabalho permite um entendimento mais preciso das questões de sincronização nos sistemas multimídia, e oferece um ganho de experiência fundamental no desenvolvimento formal de sistemas multimídia.

## Abstract

*A Multimedia System is one which allows end users to create, manipulate, store, restore and present different media such as text, graphics, images, audio and video, in an integrated manner. The synchronization of these activities according to certain quality of service imposed by the user is one of the main tasks of these systems. There are several multimedia synchronization techniques proposed in the literature and it should be noted that no one synchronization technique can be chosen to handle every application's requirements.*

*The main purpose of this work is to show how a formal description technique, i.e. Esterel, can be used in the formal development of multimedia systems. Particularly, we adopted a development methodology based on Esterel's WYPIWYE (What You Prove Is What You Execute) principle [Berry88]. Several synchronization scenarios are established, corresponding to different sets of requirements and restrictions of multimedia synchronization. Modeling of each scenario is achieved by using some of the main synchronization techniques proposed in the literature. Verification and assessment of these models is accomplished by simulation. Finally, we discuss the advantages and limitations of the methodology used in this work.*

*This work allows a more precisely knowledge about multimedia synchronization issues and offers a fundamental experience gain towards the formal development of multimedia systems.*

# Capítulo 1

## *Introdução Geral*

Os recentes progressos, em matéria de tecnologia e técnicas de comunicação nos sistemas computacionais, tem possibilitado a integração de novos tipos de informações tais como áudio e vídeo, resultando no surgimento dos chamados *Sistemas Multimídia*. Estes sistemas permitem ao usuário gerar, tratar, armazenar, restituir, e apresentar de maneira integrada mídias tais como texto, gráficos, imagens, voz, áudio e vídeo. Aplicações tais como sistemas de conferência multimídia, correio eletrônico multimídia, sistemas de aprendizagem a distância e em geral sistemas de informação multimídia, são alguns exemplos.

O tempo torna-se um fator preponderante nos Sistemas Multimídia, em particular, quando estão presentes diversas *mídias contínuas* como áudio e vídeo, cuja apresentação exige fortes requisitos temporais. Em decorrência, as atividades de geração, armazenamento, restituição, distribuição e apresentação da informação multimídia, devem respeitar estritas restrições temporais. Assim, a sincronização temporal nos Sistemas Multimídia é um problema da maior importância, a ser tratado.

Para realizar a sincronização, diferentes componentes do sistema multimídia devem intervir. Por exemplo, a aplicação deve apresentar as informações respeitando seus requisitos temporais e semânticos; o sistema operacional deve oferecer os meios de escalonamento de tarefas e alocação de recursos; o sistema de comunicação deve garantir a transmissão de objetos multimídia segundo restrições temporais e requisitos específicos de *qualidade de serviço* estabelecidos pela aplicação.

Do ponto de vista dos sistemas de comunicação multimídia, os *fluxos de dados* (um fluxo para cada mídia) podem ser considerados como os componentes básicos de informação multimídia. O problema da sincronização neste nível pode ser decomposto assim, em três aspectos complementares [Dairaine92]:

- A *sincronização intra-fluxo*: trata da sincronização de um único fluxo gerado desde um emissor até um receptor remoto.

- *A sincronização inter-fluxo*: concerne à sincronização de vários fluxos entre si visando a apresentação num só receptor.
- *A sincronização multideestino*: refere-se à sincronização de vários fluxos vindo desde um ou vários emissores até um ou vários receptores remotos.

O trabalho aqui desenvolvido aborda inicialmente um estudo da problemática da sincronização nos sistemas informáticos distribuídos, ressaltando particularmente os aspectos de comunicação destes sistemas. Uma vez caracterizados e definidos alguns dos principais mecanismos de sincronização multimídia, uma segunda fase do trabalho é dedicada a sua especificação formal utilizando a linguagem Esterel. Numa última fase do trabalho é abordada a validação por simulação destes mecanismos.

A motivação principal deste trabalho consiste em mostrar a utilização da linguagem Esterel nas diferentes fases do desenvolvimento formal de sistemas multimídia e ao mesmo tempo obter um entendimento mais preciso e ganho de experiência nos diferentes aspectos da sincronização multimídia.

Este trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta um estudo da problemática da sincronização de Sistemas Multimídia Distribuídos, ressaltando aspectos relativos à comunicação. Inicialmente são apresentadas algumas definições básicas de Sistemas Multimídia e suas principais aplicações. Em seguida, os principais requisitos são expostos, introduzindo a noção de Qualidade de Serviço e ressaltando a exigência primordial de sincronização nestes sistemas. Na seqüência, são apresentadas e discutidas algumas das principais propostas de sincronização multimídia encontradas na literatura.

O Capítulo 3 é dedicado a aspectos ligados ao projeto e desenvolvimento formal de Sistemas Multimídia Distribuídos. Em primeiro lugar, é apresentada uma visão geral das técnicas de descrição formal como elementos fundamentais no desenvolvimento formal de *software*. A seguir são consideradas as implicações do fator tempo-real, presente nos sistemas multimídia, do ponto de vista da sua especificação. Em seguida, são expostos alguns exemplos dos métodos formais atualmente utilizados no desenvolvimento de Sistemas Multimídia. Posteriormente, são expostos argumentos para a escolha de Esterel como linguagem de especificação utilizada neste trabalho,

sendo que é apresentada uma breve descrição desta ferramenta. Este capítulo é encerrado com a descrição de uma metodologia baseada em Esterel para a concepção de Sistemas Multimídia.

O Capítulo 4 mostra a utilização da linguagem Esterel na modelagem de problemas de sincronização em sistemas multimídia distribuídos. Os problemas de sincronização são apresentados na forma de cenários que reúnem requisitos e restrições particulares de apresentação da informação multimídia em cada um dos domínios de sincronização: intra-fluxo, inter-fluxo e multidestino.

O Capítulo 5 apresenta o processo de validação, através de simulação, das especificações dos problemas de sincronização multimídia introduzidos no capítulo anterior. É descrita a simulação de cada um dos cenários anteriormente modelizados, apresentando critérios para a escolha de casos de teste, o atendimento esperado de qualidade de serviço e a identificação de condições de erro.

Finalmente, são apresentadas as considerações finais deste trabalho na qual se faz uma análise crítica da metodologia de concepção de Sistemas Multimídia baseada em Esterel.

## Capítulo 2

### *Problemática da Sincronização em Sistemas Multimídia Distribuídos*

#### **2.1 Introdução**

Neste capítulo, é apresentada uma visão geral da problemática da sincronização multimídia. O interesse principal concerne às propostas de sincronização que permitem a troca de informações multimídia entre participantes distribuídos geograficamente. Inicialmente são apresentadas algumas definições básicas de sistemas multimídia e suas principais aplicações. Em seguida, são expostos os requisitos destas aplicações, e sua correspondente expressão em termos de exigências de Qualidade de Serviço. Finalmente, são apresentadas e caracterizadas algumas das principais propostas de sincronização multimídia encontradas na literatura.

#### **2.2 Sistemas Multimídia**

##### **2.2.1 Mídia, Multimídia e Aplicações Multimídia**

O termo *mídia* é comumente utilizado para definir informações tais como texto, gráficos, imagens, voz, áudio e vídeo, presentes nos sistemas multimídia. As mídias podem ser classificadas em dois grupos principais [Blair93]:

1. *Mídias discretas*: mídias que não tem uma dimensão temporal. Texto, gráficos e imagens fixas são alguns exemplos de mídias discretas.
2. *Mídias contínuas*: mídias que exibem uma forte dependência temporal. Exemplos de tais mídias são voz, áudio, imagens animadas, vídeo e dados interativos.

Os sistemas que permitem ao usuário gerar, tratar, armazenar, restituir, e apresentar de maneira integrada diferentes mídias são conhecidos como *sistemas multimídia* [Blair93]. Nos últimos anos, o potencial das chamadas *aplicações multimídia* foi enormemente ampliado com a introdução das mídias contínuas. Destacam-se particularmente os campos de automação de escritórios, indústria de serviços, aplicações científicas, culturais e domésticas [Williams94], onde



numerosos sistemas integrando voz, imagens e outras mídias, tem sido desenvolvidos. A título de exemplo citamos: tratamento de documentos multimídia (criação, edição, armazenamento e restituição), correio eletrônico multimídia, sistemas de conferências, ambientes cooperativos, sistemas de aprendizagem a distância e os diversos sistemas de informação multimídia.

### 2.2.2 Requisitos das Aplicações Multimídia

O agrupamento de aplicações multimídia conforme suas necessidades, tem sido abordado pela ITU [Tawbi92]. Nesta classificação identificam-se duas grandes categorias de serviços de aplicações multimídia que podem ser utilizados em redes digitais de banda larga (*B-ISDN Broadband Integrated Service Digital Network*): os serviços interativos e os serviços de distribuição.

A categoria de serviços interativos é subdividida em três classes distintas:

1. *Serviços de conversação*: correspondem às aplicações que exibem necessidades de comunicação bidirecional e de tempo real, para a troca de informações desde uma ou vários emissores, até um ou vários receptores. Exemplo: as vídeo-conferências
2. *Serviços de mensagens*: fornecem as funcionalidades para a troca indireta de informações entre usuários. Os documentos são armazenados e posteriormente apresentados conforme solicitado pelo usuário. Exemplo: correio eletrônico multimídia.
3. *Serviços de consulta*: abrangem os serviços que permitem a recuperação de informação de bases de dados multimídia conforme desejado pelo usuário. Exemplo: televisão por demanda.

A categoria de serviços de distribuição é por sua vez, subdivida em duas classes:

1. *Serviços de distribuição sem controle de apresentação pelo usuário*: incluem-se aqui os serviços de distribuição de informação de forma contínua, desde um emissor central até um ou vários receptores. Os usuários não tem controle sobre o começo, ou a ordem da apresentação das informações, como por exemplo no caso da televisão.
2. *Serviços de distribuição com controle de apresentação pelo usuário*: Esta classe de serviços diferencia-se da classe anterior por integrar informações de controle que

permitem ao usuário um certo controle sobre a apresentação, como por exemplo no caso do cinema ou televisão interativa.

Assim, conforme esta classificação, as aplicações multimídia apresentam requisitos diferentes e até antagônicos, segundo o tipo de serviço utilizado. Assim p. ex., aplicações de serviços de mensagens, como correio eletrônico multimídia, requerem grandes espaços de armazenamentos mas não impõem requisitos temporais estritos durante o transporte das mídias. Por outro lado, serviços de conversação, como as vídeo-conferências, exigem estritos requisitos temporais enquanto não precisam de grandes espaços de armazenamento.

Um outro aspecto a ser analisado concerne as mídias manipuladas pela aplicação. Cada mídia apresenta a suas próprias características e determina diferentes necessidades em termos de transmissão, armazenamento, apresentação, etc.

Por outro lado, os seguintes requisitos podem ser considerados como comuns à maioria de aplicações multimídia [Dairaine94]:

- *Distribuição*: as aplicações multimídia são por natureza distribuídas pois a produção, armazenamento e apresentação das mídias podem ocorrer em lugares distantes geograficamente.
- *Tempo real*: na geração, recuperação, transporte, processamento ou apresentação da informação multimídia o fator tempo é determinante.
- *Difusão de informação*: muitas aplicações multimídia permitem a interação de grupos de participantes. O subsistema de comunicação é diretamente implicado: os protocolos e a rede subjacente devem suportar a difusão de informação; mecanismos de suporte para a coordenação de grupos de participantes devem também ser oferecidos.
- *Sincronização*: as aplicações manipulam informações ligadas a restrições temporais. A sincronização das mídias relacionadas temporalmente com um certo nível de qualidade, é fundamentalmente uma meta para estes sistemas, e em consequência é necessário se preocupar com a sincronização a todos os níveis do sistema.

### 2.2.3 Qualidade de Serviço nos Sistemas Multimídia

A noção de qualidade de serviço (*QoS*) é fundamental para os sistemas multimídia. As necessidades ligadas a cada tipo de aplicação podem ser especificadas em termos de um certo conjunto de parâmetros de *QoS*, entretanto não existe ainda uma definição padrão de um conjunto específico [Koegel94].

Conforme o nível do sistema multimídia analisado tem-se diferentes conjuntos de parâmetros de *QoS*. Assim, do ponto de vista da aplicação, dimensão da imagem, resolução e número de imagens por segundo, poderiam constituir os parâmetros de *QoS*. Enquanto, do ponto de vista do subsistema de comunicação, manipulam-se tipicamente como parâmetros: atraso fim-a-fim, vazão e taxa de erro, os quais permitem configurar convenientemente os protocolos de tempo-real do subsistema de comunicação multimídia. No nível do sistema operacional, as características de *QoS* podem ser expressas em termos de exigências de garantias de processamento ou necessidades de escalonamento em tempo-real das tarefas que processam a informação multimídia.

A negociação, a configuração e o suporte dinâmico dos parâmetros de *QoS* constituem importantes extensões das exigências do serviço baseado na noção de qualidade. Em seguida apresentam-se a título de exemplo os parâmetros de *QoS* típicos do subsistema de comunicação definidos em [Ferrari90]:

- **O Atraso:** o atraso fim-a-fim, ainda chamado de *latência* corresponde ao tempo entre a aquisição de uma informação (no emissor), e sua correspondente apresentação (no receptor). O atraso fim-a-fim pode ser decomposto em três componentes: um *atraso de coleção* no emissor (incluindo a *aquisição*, *codificação* e *empacotamento* da informação), um *atraso de transmissão* na rede e um *atraso de restituição* no receptor (incluindo os atrasos de *desempacotamento*, *decodificação* e *apresentação*).

O termo *jitter* é utilizado para definir a variação instantânea do atraso fim-a-fim. O *jitter* é ocasionado principalmente por atrasos estatísticos em filas no emissor e no receptor, por erros (perdas ou corrupção de pacotes) na rede e por diferenças entre os relógios do emissor e do receptor. A partir de um valor limite do *jitter*, é possível dimensionar um *buffer* no receptor que permite filtrar estas variações de atraso. Um pacote que ultrapasse o valor do *jitter* tolerado, leva à certa degradação da qualidade da informação, sendo que este pode receber o mesmo tratamento que um pacote perdido.

*Requisitos de atraso:* o valor máximo do atraso ( $D_{max}$ ) e do seu jitter ( $J_{max}$ ) em relação a um atraso ideal ( $D$ ) são definidos pelo usuário; cada pacote deve respeitar limites superior ( $D+J_{max}$ ) e inferior ( $D-J_{max}$ ) para seu atraso.

- **A Vazão:** a vazão indica a taxa de transferência de pacotes entre o emissor e o receptor. A vazão pode ser expressa em termos de kilobits por segundo ou em número de pacotes entregues por segundo.

*Requisitos de vazão:* O limite superior da vazão é sempre fornecido pela taxa de envio no emissor; a possibilidade de perda de pacotes ou de ocorrência de erros abaixa o valor da vazão. Por outro lado, a vazão máxima é limitada ainda pela carga da rede, o que pode levar a uma saturação da vazão se o emissor aumentar sua taxa de envio. O usuário define um limite inferior da vazão ( $V_{min}$ ) que deve ser fornecido pelo sistema. Quando a faixa passante alocada para a transferência não é constante, mas varia dinamicamente segundo a demanda, é interessante utilizar um valor de vazão média ( $V_{med}$ ).

- **A Confiabilidade:** a confiabilidade em um sistema multimídia é dimensionada em termos da taxa de erro por bit ( $BER$ ) e taxa de erro por pacote ( $PER$ ) que representam o número de erros por unidade de tempo, por bit e por pacote respectivamente.

*Requisitos de confiabilidade:* por diversas razões as mensagens podem ser corrompidas ou perdidas durante a transmissão. Do ponto de vista do receptor, considera-se perda de mensagem nos dois casos, sem distinguir as razões. Em consequência, basta este expressar o limite de confiabilidade como o limite inferior da probabilidade ( $C_{min}$ ) da mensagem ser entregue corretamente:  $Prob(mensagem\ entregue\ correta) \geq C_{min}$ . Isto implica que a probabilidade de perda é limitada por  $(1-C_{min})$ .

A título de exemplo, na Tabela 2.1 mostram-se alguns valores de parâmetros de  $QoS$  para voz, vídeo e vídeo comprimido.

<i>Mídia</i>	<i>Atraso máx. (s)</i>	<i>Jitter máx. (ms)</i>	<i>Vazão média V<sub>med</sub> (Mb/s)</i>	<i>Taxa de erro/ bit BER</i>	<i>Taxa de erro/pacote PER</i>
Voz	0,25	10	0,064	< 10 <sup>-1</sup>	10 <sup>-1</sup>
Vídeo	0,25	10	100	10 <sup>-2</sup>	10 <sup>-3</sup>
Vídeo comprimido	0,25	1	[2,10]	10 <sup>-6</sup>	10 <sup>-9</sup>

Tabela 2.1 Alguns valores de parâmetros de QoS [Dairaine94].

## 2.3 A Sincronização nos Sistemas Multimídia Distribuídos

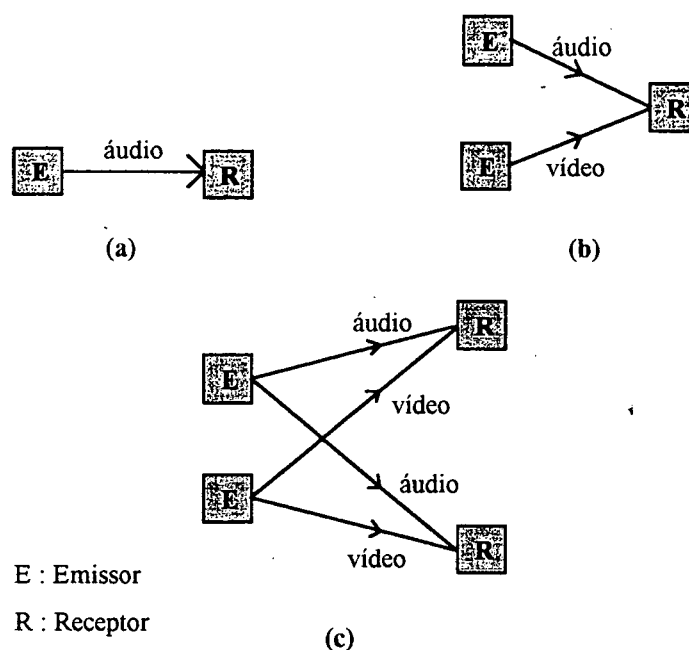
### 2.3.1 Aspectos Gerais

Vimos que a sincronização é um requisito fundamental nos sistemas multimídia pois os serviços de geração, recuperação, transporte, processamento e apresentação de mídias, oferecidos pelo sistemas multimídia, estão fortemente relacionados ao tempo. A sincronização multimídia é então vista como a tarefa de coordenar, ordenar e organizar a apresentação dos objetos multimídia.

Para realizar a sincronização, diferentes componentes do sistema multimídia devem intervir. Por exemplo, a aplicação deve apresentar as informações respeitando seus requisitos temporais e semânticos; o sistema operacional deve oferecer os meios de escalonamento de tarefas e alocação de recursos; o sistema de comunicação deve garantir a transmissão de objetos multimídia segundo restrições temporais e requisitos específicos de qualidade de serviço estabelecidos pela aplicação.

Do ponto de vista do sistema de comunicação multimídia, os *fluxos de dados* (um fluxo para cada mídia) são considerados como as abstrações básicas de informação multimídia. O problema da sincronização neste nível pode ser decomposto assim, em três aspectos complementares [Dairaine92]:

- *A sincronização intra-fluxo*: trata da sincronização de um único fluxo gerado desde um emissor até um receptor remoto (Figura 2.1a).
- *A sincronização inter-fluxo*: concerne à sincronização de vários fluxos entre si visando a apresentação num só receptor (Figura 2.1b).
- *A sincronização multideestino*: refere-se à sincronização de vários fluxos vindo desde um ou vários emissores até um ou vários receptores remotos (Figura 2.1c).



**Figura 2.1 Exemplos de Sincronização: (a) Intra-Fluxo, (b) Inter-Fluxo e (c) Multidestino.**

1. A *sincronização intra-fluxo* visa a conservação das relações temporais existentes entre as unidades de informação constituintes de um fluxo, p. ex. necessárias para a restituição de uma seqüência de amostras de voz. Em particular, procura-se basicamente compensar as variações de atrasos fim-a-fim (*jitter*) respeitando um certo nível de latência (interatividade), conforme a qualidade de serviço especificada pela aplicação. Observe-se que, em muitas aplicações multimídia, um pacote atrasado é tão inútil como outro que nunca chegou, pois já não corresponde com seu instante desejado de apresentação. O requisito de qualidade de serviço da aplicação reflete-se geralmente neste caso em exigências de garantia de mínimo/máximo atraso fim-a-fim do fluxo, limite de perda de pacotes e exigências de largura de banda.
2. A *sincronização inter-fluxo* visa garantir as relações temporais entre as unidades de informação dos diferentes fluxos a sincronizar, sendo que cada fluxo precisa da sincronização intra-fluxo independentemente. A sincronização de voz e vídeo numa aplicação de sincronização de lábios (*lip-sync*) é um exemplo típico deste tipo de sincronização. A soma de novas fontes de variação de atrasos fim-a-fim, tal como diferentes tempos de processamento dos fluxos, (p. ex. no caso de um fluxo de vídeo comprimido e outro de áudio não comprimido), as diferentes distâncias entre emissores e receptor, assim como os diferentes requisitos dos fluxos fazem com que o problema da sincronização inter-fluxo seja mais

complexo. Além dos parâmetros de qualidade de serviço do domínio intra-fluxo, é necessário especificar a deriva máxima aceitável entre os fluxos envolvidos.

3. A *sincronização multideestino* é responsável por garantir as propriedades temporais entre diferentes receptores. Deve permitir a apresentação simultânea dos fluxos em todos os receptores. Entretanto, as diferentes distâncias entre receptores fazem com que os fluxos sofram derivas inter-destinos, dependendo do método de difusão seletiva (*multicast*) utilizado, incrementando-se assim as variações de atrasos fim-a-fim a compensar.

### 2.3.2 Representação da Informação Multimídia: Definições Básicas

A modo de formalizar, neste documento serão utilizadas as seguintes definições relacionadas à sincronização multimídia já adotadas na literatura em [Santoso93], [Carmo94a] e [Dairaine94]:

- **Unidades de Informação (IUs - Information Units):** constituem abstrações de informações elementares associadas a um tipo de mídia particular, p. ex. amostras de áudio ou quadros de vídeo.
- **Fluxo (Stream):** é entendido como uma seqüência de IUs relacionadas temporalmente, como por exemplo um fluxo de vídeo, constituído por uma seqüência de quadros de vídeo.
- **Feixe (Bundle):** refere-se a uma coleção de fluxos inter-relacionados temporalmente, como por exemplo os fluxos de áudio e vídeo numa aplicação de tele-conferência.
- **Assinatura Temporal de um Fluxo (Stream Temporal Signature):** refere-se à caracterização da seqüência de IUs de um fluxo, por suas posições temporais relativas. A Figura 2.2 ilustra a assinatura temporal ( $t_1, t_2, t_3, t_4, t_5, t_6$ ) de um fluxo de unidades de informação IU1, IU2, IU3, IU4, IU5 e IU6.

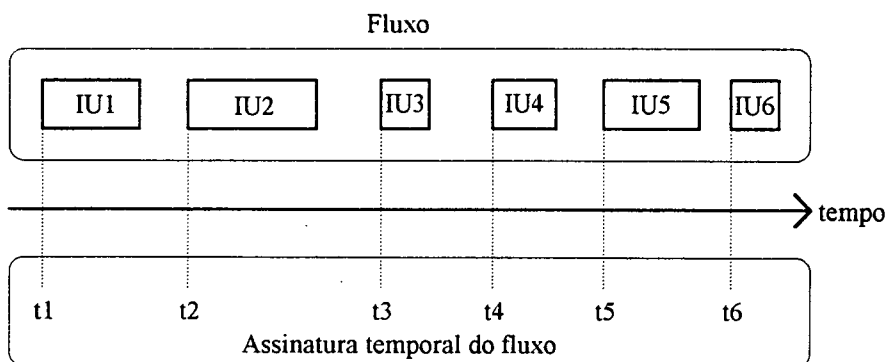


Figura 2.2 Assinatura temporal de um fluxo.

- **Assinatura Temporal de um Feixe** (*Bundle Temporal Signature*): refere-se à caracterização das posições temporais relativas da seqüência de IUs de cada fluxo constituinte do feixe. A Figura 2.3 ilustra a assinatura temporal de um feixe ( $t_{1b}$ ,  $t_{1a}$ ,  $t_{2b}$ ,  $t_{2a}$ , ...,  $t_{5a}$ ) constituído pelo fluxos *A* e *B*.

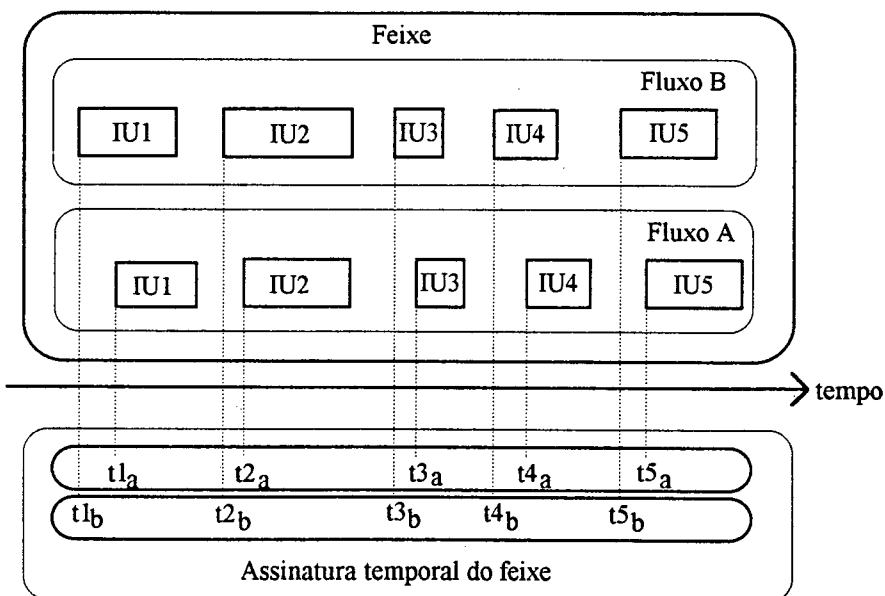


Figura 2.3 Assinatura temporal de um feixe.

Na seqüência é apresentada uma discussão sobre algumas das principais propostas de sincronização em cada um dos aspectos citados anteriormente.



## 2.4 A Sincronização Intra-Fluxo

Nesta seção são apresentadas algumas das principais propostas de sincronização intra-fluxo das abordagens: Temporal e Temporal-Causal, sendo que as propostas da abordagem Temporal são de especial interesse para este trabalho.

### 2.4.1 Abordagem Temporal

As técnicas propostas nesta abordagem, resolvem o problema da sincronização intra-fluxo utilizando basicamente diferentes estratégias de compensação do *jitter* introduzido pelo subsistema de comunicação. Estas técnicas assumem a existência de um *jitter* de comunicação limitado e suficiente espaço de armazenamento. Isto permite implementar um esquema de armazenamento temporário que compensa o *jitter* de comunicação para qualquer fluxo [Nicolaou90].

A qualidade do serviço de sincronização oferecida, depende diretamente da hipótese de sincronização de relógios adotada. Quatro hipóteses são consideradas [Dairaine94]:

1. *Sistemas de relógios comuns*: Os relógios do sistema tem a mesma percepção do tempo. Esta hipótese pode ser formulada da maneira seguinte:  $\forall i, j \in S, t_i(e) = t_j(e)$ . Onde  $i$  e  $j$  são dois relógios quaisquer do sistema;  $S$  representa o conjunto de todos os relógios do sistema; e a notação  $t_h(e)$  indica o tempo de percepção do evento  $e$  no relógio  $h$ .
2. *Sistemas de relógios a derivas comuns*: Os relógios do sistema tem a mesma frequência entretanto possuem bases de tempo diferentes. Esta hipótese pode ser formulada da maneira seguinte:  $\forall i, j \in S, t_i(e) = t_j(e) + Cste$ . Onde  $Cste$  é uma constante que representa a diferença de bases entre os relógios  $i$  e  $j$ .
3. *Sistemas de relógios derivas constantes*: Os relógios do sistema possuem frequências e bases de tempo diferentes. Entretanto, suas frequências são constantes. Esta hipótese pode ser formulada da maneira seguinte:  $\forall i, j \in S, t_i(e) = k.t_j(e) + Cste$ . Onde  $k$  representa a razão entre as derivas (constantes) dos relógios  $i$  e  $j$ ; e a notação  $Cste$  indica um valor constante.
4. *Sistemas de relógios não sincronizados*: Os relógios do sistema possuem bases e frequências quaisquer.

Por outro lado, as técnicas variam conforme o tipo de fluxo suportado [Dairaine94]:

- *Fluxo determinista*: neste tipo de fluxo, as relações temporais de suas unidades de informação constituintes podem ser expressas por uma função determinística. Os *fluxos periódicos* são um subtipo particular deste tipo de fluxos.
- *Fluxos estocásticos*: neste tipo de fluxo, as relações temporais de suas unidades de informação constituintes não podem ser expressas por uma função determinística. O fluxo da mídia texto utilizado como subtítulos de vídeo é um exemplo deste tipo de fluxos.

#### **2.4.1.1 Sincronização de Fluxos Estocásticos**

Neste caso não é conhecida a priori a assinatura temporal do fluxo, e é necessário fazer com que todos os atrasos das IUs sejam os mesmos de maneira a preservar as relações temporais. Para a sincronização deste tipo de fluxos é adotada uma estratégia baseada em rótulos de tempo (*timestamping*), cujo fundamento pode ser descrito da maneira seguinte:

O emissor rotula as IUs com o tempo corrente segundo seu relógio local, no instante de seu envio. A compensação do *jitter* é realizada pelo armazenamento temporário das IUs no receptor até o instante de liberação previsto. Os rótulos de tempo são utilizados no cálculo do instante de liberação das IUs, conforme a hipótese de relógios assumida.

Nesta estratégia, os rótulos de tempo permitem descrever a assinatura temporal do fluxo. Esta informação é utilizada pelo receptor para reconstruir o fluxo uma vez que o *jitter* introduzido pelo subsistema de comunicação destruiu sua assinatura temporal.

### **Técnicas Baseadas na Hipótese de Sistemas de Relógios Comuns**

#### **Proposta 1**

Para a compensação do *jitter*, Ferrari, em [Ferrari92], propõe uma abordagem distribuída no nível de rede de uma rede de comutação de pacotes, sob a suposição da disponibilidade de uma referência de tempo comum (sistema de relógios comuns). Nesta técnica, cada nó entre emissor e receptor fornece a compensação do *jitter* até aí introduzido, através do armazenamento e posterior liberação das IUs. Dada uma referência de tempo global, o instante de liberação de uma IU, em cada nó, vem sendo dado pela soma de seu rótulo de tempo e o máximo atraso de

comunicação entre o nó em questão e seu imediato anterior. Esta estratégia requer que os rótulos de tempo sejam atualizados em cada nó conforme a compensação fornecida.

As características principais desta técnica são as seguintes:

- Distribui uniformemente os requisitos de armazenamento das IUs entre os nós intermediários, sendo que consegue diminuir seu requisito total. O que se caracteriza na maior vantagem desta técnica.
- Complica o tratamento de informação a nível do subsistema de comunicação e requer a sincronização global de relógios.
- A compensação de *jitter* a nível de rede, assume atrasos fixos ou desprezíveis nas camadas superiores, para preservar a assinatura temporal do fluxo. Entretanto, geralmente é necessário a realização deste mesmo tratamento a nível de receptor [Partridge91]. Desta forma, a compensação de *jitter* a nível de rede pode-se considerar redundante.

## Proposta 2

Em [Santoso93] propõe-se uma técnica de sincronização intra-fluxo, baseada em *time-stamping*, assumindo a sincronização de relógios do sistema. Neste esquema, a entidade de sincronização receptora armazena e libera cada unidade de informação, só após um tempo definido relativo a seu rótulo de tempo. Este tempo, aqui chamado de *tempo de apresentação alvo* (*TPT -Target Presentation Time*) é fixado ao valor de atraso máximo do fluxo. O instante de liberação de cada unidade de informação é determinado diretamente pela soma de seu rótulo de tempo e o valor de *TPT*.

As características principais desta técnica são as seguintes:

- Este esquema garante a preservação da assinatura temporal do fluxo.
- O atraso de comunicação compensado é fixo e igual ao tempo de apresentação alvo, *TPT*.
- Pressupõe a sincronização de relógios do sistema.

- Opera numa camada de sincronização acima da camada de transporte de uma rede de banda larga tal como *B-ISDN*, que oferece extensões de qualidade de serviço tais como garantias de largura de banda e perda de pacotes limitada, e atrasos mínimos e máximos.

Enquanto a escolha do tempo de apresentação alvo, *TPT*, igual ao atraso máximo de comunicação é adequada para aplicações que não podem tolerar certas discontinuidades (*gaps*) na apresentação da mídia, esta pode não ser apropriada para aplicações interativas onde o tempo de resposta entre emissor e receptor é crítico.

### Proposta 3

Considerando que as aplicações interativas podem tolerar um certo número de discontinuidades (p. ex. no caso de conversação), em [Dairaine93] propõe-se uma escolha do valor do *TPT*, baseado na determinação estatística de um limite superior do número de discontinuidades e de seu tempo máximo de ocorrência, durante a conversação. O valor do *TPT* é fixado assim num valor inferior ao máximo atraso fim-a-fim, conforme a tolerância imposta pela aplicação, resultando numa melhora da interatividade. O mecanismo consegue assim a conservação quase ótima da assinatura temporal e garante um atraso de comunicação compensado igual ao valor do *TPT*.

Uma consequência importante da escolha de um valor inferior de *TPT* é a redução dos requisitos de espaço de armazenamento no *buffer* do sincronizador. Nas estratégias estatísticas, porém, surge o problema de decidir sobre o tratamento das IUs atrasadas. Nesta proposta, o valor de tempo máximo de ocorrências de discontinuidades, oferece à aplicação um critério para decidir, na fase de negociação de parâmetros, acerca da recepção ou omissão de IUs atrasadas.

### Técnicas Baseadas na Hipótese de Sistemas de Relógios a Derivas Comuns

Em [Santoso93] apresenta-se uma técnica de sincronização intra-fluxo, baseada em *time-stamping*, assumindo que os relógios do sistema possuem derivas comuns. O tempo de apresentação alvo (*TPT - Target Presentation Time*) é fixado ao valor de atraso máximo do fluxo. Devido à hipótese de relógios do sistema, a percepção do tempo no emissor e receptor difere numa constante e em consequência não é possível determinar diretamente os instantes de liberação. Entretanto, é possível deduzir o instante de liberação de uma IU<sub>n</sub>, em relação a sua

imediate anterior,  $IUn-1$ . Este tempo vem sendo dado pela diferença entre os rótulos de tempo da  $IUn$  e  $IUn-1$ , acrescentada pelo instante de apresentação da  $IUn-1$ .

Considerando a impossibilidade do cálculo do instante de liberação da primeira IU, o mecanismo assume seu atraso mínimo de forma a maximizar seu tempo de armazenamento e assegurar a compensação do *jitter*. Informalmente, o instante de liberação da primeira IU, em relação ao relógio do receptor, vem sendo dado por:

$$\text{instante de liberação} = \text{tempo de chegada ao receptor} + (TPT - \text{atraso mínimo})$$

Para as outras IUs o instante de liberação vem sendo dado por:

$$\text{instante de liberação da } IUn = \text{instante de liberação da } IUn-1 + TGR$$

Onde  $TGR$  representa o tempo de geração da IU atual em relação à IU anterior, e vem sendo dado por:

$$TGR = \text{rótulo de tempo da } IUn - \text{rótulo de tempo da } IUn-1$$

Desta forma o mecanismo garante a preservação da assinatura temporal do fluxo mas introduz uma incerteza no atraso de comunicação compensado.

As características principais desta técnica são as seguintes:

- Garante a preservação da assinatura temporal do fluxo.
- O atraso de comunicação compensado apresenta uma incerteza, sendo que seu valor é compreendido no intervalo  $[TPT, TPT + \Delta_{\max} - \Delta_{\min}]$ , onde  $\Delta_{\min}$  e  $\Delta_{\max}$  representam os atrasos mínimo e máximo de comunicação.
- Pressupõe um sistema de relógios a derivas comuns.
- Opera numa camada de sincronização acima da camada de transporte de uma rede de banda larga tal como *B-ISDN*, que oferece extensões de qualidade de serviço tais como garantias de largura de banda e perda de pacotes limitada, e atrasos mínimos e máximos.

#### **2.4.1.2 Sincronização de Fluxos Periódicos**

As técnicas da abordagem temporal que fornecem a sincronização de fluxos periódicos exploram uma característica fundamental deste tipo de fluxos: a assinatura temporal de um fluxo periódico é

definida por seu período nominal. Assim, uma vez que o receptor conhece o período nominal do fluxo, pode prescindir dos rótulos de tempos basicamente usados para descrever a assinatura temporal do fluxo. Como consequência, a técnica evita o *overhead* causado pelo processo de leituras e escritas associadas ao rótulo temporal. Adicionalmente, reduzem-se os requisitos de largura de banda pois não é mais necessário o transporte dos rótulos de tempo.

### **Técnicas Baseadas na Hipótese de Sistemas de Relógios a Derivas Comuns**

A técnica de pre-busca ou *prefetch* [Dairaine94] fornece a sincronização intra-fluxo de fluxos periódicos num sistema de relógios a derivas comuns. Nesta técnica é apenas necessário que a entidade de sincronização receptora conserve armazenadas um certo número de IUs, justo no início da sua entrega no período nominal do fluxo. Desta forma, introduz-se um atraso inicial que compensa o *jitter* de rede.

O valor do *prefetch* ( $P$ ) corresponde a este número de IUs que permanecem armazenadas justo no início da entrega, e vem sendo dado por:

$$P = \lceil (\Delta_{\max} - \Delta_{\min}) / \theta_{\text{nom}} \rceil$$

Onde  $\theta_{\text{nom}}$  denota o período nominal do fluxo.  $\Delta_{\min}$  e  $\Delta_{\max}$  representam os atrasos mínimo e máximo de comunicação. A notação  $\lceil x \rceil$  indica o valor inteiro superior ou igual a  $x$ .

As características principais desta técnica são as seguintes:

- Simples e de pouco custo computacional.
- Fornece a preservação da assinatura temporal do fluxo.
- O atraso de comunicação compensado apresenta uma incerteza, e só pode ser garantido que ele esteja compreendido no intervalo  $[\theta_{\text{nom}} * P + \Delta_{\min}, \theta_{\text{nom}} * P + \Delta_{\max}]$ .

### **Técnicas Baseadas na Hipótese de Sistemas de Relógios a Derivas Constantes**

#### **Proposta 1**

A técnica do emulador de período proposta em [Dairaine94], fornece a sincronização de fluxos periódicos onde os períodos de produção e consumo são diferentes mas constantes (sistema de

relógios a derivas constantes). O mecanismo resolve particularmente o problema de fome ou saturação do *buffer* receptor com a introdução dos seguintes elementos:

- Espera por um certo número de IUs antes de começar a apresentação (técnica do *prefetch*).
- Operações periódicas para eliminar ou duplicar IUs (*delete/duplicate*), realizadas por um emulador de período.

A técnica do *prefetch* é utilizada para compensar o *jitter* de rede através da introdução de um atraso de compensação equivalente a um certo número de IUs que permanecem retidas justo no início da apresentação. Este valor vem sendo dado por:

$$P = \lceil (\Delta_{\max} - \Delta_{\min}) / \theta_{\text{in}} \rceil$$

Onde  $\theta_{\text{in}}$  denota o período de geração do fluxo.  $\Delta_{\min}$  e  $\Delta_{\max}$  representam os atrasos mínimo e máximo de comunicação. A notação  $\lceil x \rceil$  indica o valor inteiro superior ou igual a  $x$ .

O emulador de período evita a saturação ou fome do *buffer* receptor, com operações periódicas de delete ou *duplicate* sobre as IUs recebidas segundo seja o caso da diferença de períodos entre emissor e receptor. No caso de um período de produção,  $\theta_{\text{in}}$ , menor que o período de consumo,  $\theta_{\text{out}}$ , ou seja, a situação onde a saturação do *buffer* é iminente, o emulador elimina do *buffer* uma IU por vez a intervalos de  $\theta_{\text{emu}} = (\theta_{\text{out}} * \theta_{\text{in}}) / (\theta_{\text{out}} - \theta_{\text{in}})$ . No caso de um período de produção maior que o período de consumo, i.e., na situação de potencial fome no receptor, o emulador duplica uma IU por vez a intervalos dados por  $\theta_{\text{emu}} = (\theta_{\text{in}} * \theta_{\text{out}}) / (\theta_{\text{in}} - \theta_{\text{out}})$ . Estas operações *delete* ou *duplicate* conseguem respectivamente os efeitos de dilatação ou compressão do fluxo [Carmo93].

As características principais desta técnica são as seguintes:

- Fornece a sincronização intra-fluxo de mídias periódicas onde os períodos de produção e consumo são diferentes mas constantes.
- Pressupõe conhecidos os períodos de produção e consumo do fluxo.

- A compensação da diferença de períodos não exige nenhuma modificação dos relógios local ou remoto do sistema, estabelecendo assim certa independência entre emissor e receptor.
- Fornece a compensação da diferença de períodos através de operações periódicas *delete* ou *duplicate* conseguindo-se o efeito respectivo de dilatação ou compressão do fluxo.
- O valor atual do atraso de comunicação compensado não é conhecido, e só pode ser garantida sua permanência no intervalo  $[\theta_{in} * P + \Delta_{min}, \theta_{in} * P + \Delta_{max}]$ .
- O emissor não precisa alterar o fluxo utilizando *time-stamps* ou marcas de sincronização, e o receptor apenas realiza operações simples no *buffer*, o que faz com que a técnica seja de pouco custo computacional.

## Proposta 2

A técnica anterior assume como conhecidos os períodos de produção e consumo do fluxo. Para os casos onde estes valores sejam desconhecidos, Dairaine, em [Dairaine94], propõe uma *estratégia de estimação dicotômica*. Esta estratégia fundamenta-se na contínua confrontação da evolução da carga do *buffer* receptor com valores teóricos de mínima e máxima carga. Uma carga do *buffer* fora deste limites indica que os períodos de entrada e saída são diferentes. Segundo a ultrapassagem, pode-se deduzir se o período de consumo é inferior ou superior àquele de produção. Sob a hipótese de relógios a derivas constantes, é possível fazer com que o ajuste do período de consumo, para cada ultrapassagem dos limites teóricos, convirja para o valor do período de produção. Para isto utiliza-se um algoritmo dito dicotômico.

## Técnicas Baseadas na Hipótese de Sistemas de Relógios não Sincronizados

O mecanismo apresentado em [Ramanathan93], fornece a sincronização de fluxos periódicos na presença do jitter de rede e de variações não determinísticas do período de apresentação do receptor. O mecanismo é basicamente utilizado para fluxos previamente armazenados. Assume-se na proposta, que o receptor, (neste artigo chamado de *midiafone*), é um dispositivo com a mínima capacidade de apresentação e sem a sofisticação para implementar algum mecanismo de sincronização. Neste esquema, o emissor (Servidor Multimídia) fornece a sincronização utilizando pequenas mensagens de realimentação, enviados desde o receptor, que lhe permitem estimar os



tempos atuais de apresentação de cada unidade de informação. Baseado nesta informação, o emissor determina os tempos mínimo e máximo de envio de IUs posteriores visando evitar saturação ou fome no *buffer* do receptor. Assim, este esquema garante a continuidade do fluxo mas precisa de um canal adicional para o transporte de mensagens de realimentação.

As principais características desta técnica são as seguintes:

- Trata-se de um esquema de sincronização de fluxos periódicos enviados a receptores remotos que apresentam variações nas taxas de apresentação.
- O emissor fornece um controle de fluxo que garante a continuidade de apresentação da mídia no receptor. Com este propósito, usa mensagens de realimentação desde o receptor.
- Permite determinar a quantidade mínima de mensagens de realimentação necessária para garantir a continuidade, baseado no tamanho do *buffer* do receptor.
- Permite a definição de uma política flexível de escalonamento de envios, conforme a estimação de intervalos válidos de transmissão de IUs.
- Pressupõe conhecidos os limites de variação da taxa de apresentação do receptor.
- Não pressupõe sincronização de relógios de emissor e receptor.
- Pode-se adaptar dinamicamente às variações de carga da rede através de estimativas periódicas dos limites de atrasos atuais.
- As mensagens de realimentação precisam de um canal adicional e introduzem certa carga adicional na rede.

### 2.4.2 Abordagem Temporal-Causal

Em [Carmo94a] propõem-se extensões dos requisitos geralmente atendidos nas técnicas de sincronização intra-fluxo e associadas principalmente à preservação da assinatura temporal de um fluxo:

1. Uma extensão temporal que se refere à possibilidade de alteração da assinatura temporal do fluxo gerado pelo emissor, em termos de sua dilatação ou compressão; pode-se

considerar desta forma requisitos de aplicações onde, por exemplo, seqüências de imagens e/ou fluxos de vídeo devem ser entregues a velocidades diferentes a aquela nominal na qual forem originalmente geradas.

2. Uma extensão funcional que diz respeito ao atendimento de restrições adicionais, sobre a entrega final da informação ao usuário receptor, envolvendo a avaliação de dependências condicionais, na forma de expressões lógicas (*boolean*), entre as unidades de informação (IUs) de um fluxo. Estas restrições são particularmente proveitosas em aplicações cujos fluxos contenham IUs relacionadas semanticamente, e onde quer que não sejam utilizados serviços de transporte que garantem seqüenciamento e/ou confiabilidade. Assim, por exemplo, sob esquemas de compressão diferencial de fluxos de vídeo é possível *filtrar* o despacho de cada quadro de vídeo cuja codificação depende de um ou mais quadros eventualmente perdidos. A filtragem é considerada: *ativa* no caso de operar uma reordenação de IUs com o objetivo de tentar satisfazer suas relações de dependência; ou *passiva*, se apenas verifica a validade destas relações antes do despacho de cada IU.

As características principais desta técnica são as seguintes:

- Garante a preservação da assinatura temporal do fluxo e suporta adicionalmente sua alteração sistemática por compressão ou dilatação.
- Suporta a entrega condicional das IUs baseado em expressões lógicas, associadas a cada uma destas, que especificam suas relações de dependência.
- Opera sob as hipóteses de sistemas de relógios comuns ou a derivas comuns.
- Opera numa camada de sincronização acima da camada de transporte de uma rede de banda larga tal como *B-ISDN*, que oferece serviços orientados a *datagrama* (seqüenciamento não garantido), e parâmetros de qualidade de serviço tais como garantias de largura de banda e perda de pacotes limitada, e atrasos mínimos e máximos.
- O armazenamento e escalonamento das IUs são fornecidos na entidade de sincronização receptora, e eventualmente na entidade de sincronização emissora, no caso do requisito da alteração da assinatura temporal.

## 2.5 A Sincronização Inter-Fluxo

Nesta seção são apresentadas algumas das principais propostas de sincronização inter-fluxo das abordagens: Multiplexação, Causal e Temporal, sendo que as propostas da abordagem Temporal são de especial interesse para este trabalho.

### 2.5.1 Abordagem de Multiplexação

Esta técnica consiste em transmitir as diferentes mídias a sincronizar através de um canal de comunicação único (p. ex. uma conexão da camada de transporte). Uma multiplexação de mídias é realizada a nível do emissor com sua correspondente demultiplexação a nível de receptor. O agrupamento das diferentes mídias a sincronizar (p. ex. voz e imagem) permite fixar as relações temporais inter-fluxo durante a transmissão, devido a que as mídias multiplexadas estão sujeitas ao mesmo atraso de comunicação. Um exemplo de tal mecanismo é proposto em [Leung90].

As características principais desta técnica são as seguintes:

- Oferece uma solução simples cuja implementação requer poucas modificações de protocolos existentes. Não requer canais de controle adicionais nem sincronização dos relógios do sistema.
- Permite conservar as relações temporais inter-fluxo, entretanto a preservação das relações intra-fluxo, depende da qualidade de serviço oferecida pelo canal de transmissão utilizado. Assim, esta técnica só resolve parcialmente o problema da sincronização.
- O esquema pode ser utilizado em serviços de difusão seletiva (*multicast*), entretanto sem sincronizar os receptores entre si.
- Uma vez que os fluxos multiplexados apresentam diferentes características, o canal de comunicação é obrigado a oferecer uma qualidade de serviço equivalente à máxima exigência de qualidade de serviço entre os fluxos. Assim, as características individuais das mídias não são aproveitadas, resultando isto numa perda da eficiência do serviço de comunicação e de um aumento global do tempo de transmissão, o que se configura na maior desvantagem do esquema.

## 2.5.2 Abordagem Causal

### 2.5.2.1 Técnica das Marcas de Sincronização

A técnica de marcas de sincronização (*SM - Synchronization Marker*), introduzida em [Shepherd90] é um dos esquemas clássicos para prover serviços de comunicação que fornecem a sincronização inter-fluxo quando os fluxos são transmitidos por diferentes canais de comunicação. Neste esquema o emissor insere marcas de sincronização, de forma periódica e simultânea, em todos os fluxos a ser sincronizados. No caso destes fluxos sofrerem diferentes atrasos, o receptor não recebe as marcas de sincronização simultaneamente. Para sincronizar os fluxos, a entidade de sincronização receptora deve reter (armazenar) os fluxos cujas *SM* já tenham chegado. Os fluxos cujas *SM* não tenham chegado (fluxos atrasados) continuam sendo apresentados. No momento de chegarem as correspondentes *SM* de todos os fluxos atrasados, todas as IUs são liberadas simultaneamente.

As características principais desta técnica são as seguintes:

- Sua implementação é simples sendo que exige poucas modificações a nível de protocolos, unicamente a introdução de marcas de sincronização, o que se configura na maior vantagem do esquema.
- Devido a sincronização ser fornecida a nível de receptor, o esquema pode ser utilizado em serviços de difusão seletiva (*multicast*). Entretanto, o esquema só fornece a sincronização inter-fluxo em configurações de um emissor e um receptor.
- Não integra nenhuma noção explícita do tempo, pois apesar das IUs serem apresentadas simultaneamente, o tempo atual desta apresentação é indefinido.
- Requer grandes recursos de armazenamento no receptor dependendo das variações de atrasos entre os fluxos e das taxas de produção destes.
- Modifica fortemente os fluxos de informação, operação difícil no caso de certos tipos de compressão de dados.

Estas três últimas características constituem as maiores desvantagens do esquema e restringem seu campo de aplicação.

### 2.5.2.2 Técnica do Canal de Sincronização

A técnica do canal de sincronização, também introduzida em [Shepherd90], utiliza um canal de comunicação adicional chamado *canal de sincronização*. Este canal é utilizado pelo emissor para transmitir ao receptor informações de sincronização especificadas no momento da transmissão. Estas informações são compostas de um parâmetro de controle de apresentação (serial, paralela ou independente) e de pontos de referências dos diferentes fluxos. O receptor retém e libera os fluxos conforme os parâmetros de controle de apresentação recebido pelo canal de sincronização.

As características principais desta técnica são as seguintes:

- Permite regular a granularidade da sincronização ajustando a frequência de inserção de pontos de referência no canal de sincronização.
- Devido a sincronização ser fornecida a nível de receptor, a técnica pode ser utilizada em serviços de difusão seletiva (*multicast*).
- O esquema é utilizável para a sincronização inter-fluxo em configurações de um emissor e de um receptor e alternativamente em configurações de mais de um emissor no caso de uma especificação prévia das informações de controle.
- Neste esquema, como no anterior, não é integrada nenhuma noção explícita do tempo. Entretanto, isto pode ser resolvido pela inserção adicional de informação temporal no canal de sincronização [Saparamadu93].
- No caso de vários emissores, o esquema requer um canal de sincronização para cada emissor.

### 2.5.2.3 Técnica da Apresentação Condicional

Em [Courtiat94] apresenta-se uma extensão, para o caso de sincronização inter-fluxo, do conceito de dependências condicionais apresentado em [Carmo94a] (Veja na seção 2.4.2). As relações de dependência condicional inter-fluxo apresentam-se de duas formas:

1. *Dependências condicionais simples*. Corresponde a uma generalização direta das dependências intra-fluxo. A representação destas relações através de um grafo dirigido, resulta num grafo sem circuitos, onde um arco desde  $x$  até  $y$ , indica que a apresentação da

IU  $x$  depende da apresentação da IU  $y$ . Desta forma, uma IU será entregue imediatamente só após a entrega de todas as IUs das quais depende. A entrega de uma IU está igualmente condicionada pelas restrições temporais impostas no fluxo ao qual pertence.

2. *Dependências condicionais acopladas.* Sua representação por um grafo dirigido, determina a existência de circuitos que definem os chamados *pontos de sincronização*. Por exemplo, o circuito formado por uma seta desde  $x$  até  $y$  e outra de  $y$  a  $x$ , indica que a apresentação da IU  $x$  está condicionada à apresentação da IU  $y$ , e vice-versa. Tal condicionamento recíproco, entre IUs de diferentes fluxos especifica a sua apresentação simultânea. No caso de não estar disponível alguma das IUs envolvidas, nenhuma delas é entregue.

Assim o esquema, não impõe restrições temporais adicionais nos fluxos para especificar a sincronização inter-fluxo desejada. No lugar destas, utilizam-se as dependências condicionais para expressar relações de dependência entre fluxos relacionados temporalmente. A apresentação de *slides* com comentários sonoros é um tipo de aplicação que ilustra a vantagem desta característica. Neste caso, o mais importante não é preservar a assinatura temporal de ambos fluxos, mas sim, garantir a sincronização entre cada *slide* e seu correspondente comentário de áudio. A definição de pontos de sincronização entre estes últimos, expressa precisamente esta necessidade.

As características principais desta técnica são as seguintes:

- Proveitosa em aplicações cujos fluxos contenham IUs relacionadas semanticamente, e quando não sejam utilizados serviços de transporte que garantem seqüenciamento e/ou confiabilidade.
- Os requisitos condicionais inter-fluxo são satisfeitos só no caso de que os requisitos temporais de cada fluxo (se existem) estejam satisfeitos.
- Opera sob as hipóteses de sistemas de relógios comuns ou a derivas comuns.

### 2.5.3 Abordagem Temporal

As técnicas de sincronização intra-fluxo da abordagem temporal podem ser estendidas ao domínio da sincronização inter-fluxo. Para isto, duas condições devem ser satisfeitas:

1. Os emissores devem sincronizar o começo do envio dos fluxos, através da definição de uma referência de início de sessão comum.
2. Os atrasos de comunicação dos fluxos a sincronizar devem ser equalizados no receptor. Isto é, deve ser definido um mesmo atraso para todos os fluxos.

### **2.5.3.1 Sincronização de Fluxos Estocásticos**

Neste caso uma estratégia baseada em *time-stamping* é utilizada. No domínio de sincronização inter-fluxo, esta estratégia pode-se descrever assim:

Os emissores rotulam as IUs com o tempo corrente, segundo seus relógios locais, no instante de seu envio. A compensação do *jitter* é realizada pelo armazenamento temporário das IUs do feixe num único receptor até um instante definido de liberação comum. O cálculo do instante de liberação depende da hipótese de relógios assumida, e deve considerar os atrasos de coleção, e restituição próprios de cada fluxo.

A definição do valor do atraso comum é um aspecto chave neste contexto. Em geral corresponde ao máximo valor de atraso entre os fluxos do feixe. Como consequência, os fluxos mais rápidos devem esperar pelos mais lentos incrementando-se assim, seus respectivos requisitos de armazenamento no receptor.

### **Técnicas Baseadas na Hipótese de Sistemas de Relógios Comuns**

Dada uma referência de tempo global, as técnicas de sincronização inter-fluxo da abordagem temporal não apresentam dificuldade para a sincronização de emissores. A definição de uma referência temporal absoluta é suficiente neste caso.

Escobar, em [Escobar94], apresenta uma proposta de um *protocolo de sincronização (Flow Synchronization Protocol)* baseado em *time-stamping*, que preserva a assinatura temporal do feixe num sistema de relógios sincronizados. Um aspecto principal da proposta é a determinação do atraso comum, no artigo citado chamado de *atraso de sincronização (Synchronization Delay)*. Este valor é calculado periodicamente por um protocolo distribuído segundo as estimações dos atrasos máximos atuais dos diferentes fluxos, e corresponde em geral ao valor do fluxo mais lento. Desta forma, o protocolo pode-se adaptar a mudanças de atrasos de rede.

Devido à hipótese de relógios, os instantes de liberação das IUs do feixe são diretamente calculados:

$$\text{instante de liberação} = \text{rótulo de tempo} + \text{atraso de sincronização} - \text{atraso de restituição}$$

Algumas características desta técnica são as seguintes:

- Adapta-se às mudanças de atrasos de rede através do cálculo periódico do atraso de sincronização.
- Pressupõe a sincronização dos relógios do sistema. Entretanto, esta hipótese parece menos realista em configurações que envolvem mais de um emissor e um receptor [Carmo94b].
- Pode ser utilizada em serviços de difusão seletiva.

### **Técnicas Baseadas na Hipótese de Sistemas de Relógios a Derivas Comuns**

O relaxamento da hipótese de relógios faz com que o problema da sincronização se torne mais difícil neste caso.

Um primeiro problema refere-se à sincronização dos instantes de liberação das IUs dos diferentes fluxos. Lembremos que sob esta hipótese de relógios, a técnica de sincronização intra-fluxo só garante a liberação da primeira IU com uma margem de incerteza igual a  $(\Delta_{\max} - \Delta_{\min})$  (Veja na seção 2.4.1.1). A extensão desta técnica ao domínio inter-fluxo, com a simples definição de um atraso comum, produz uma defasagem das assinaturas temporais dos fluxos.

Esta situação pode ser remediada com a sincronização dos instantes de liberação das primeiras IUs. Uma possibilidade, é utilizar uma extensão da técnica de sincronização intra-fluxo como mecanismo subjacente para fornecer a sincronização de cada fluxo em separado [Dairaine94]. Neste caso, é necessário fazer com que só no primeiro instante de liberação, no lugar de entregar a primeira IU, o mecanismo intra-fluxo informe ao mecanismo superior (inter-fluxo) sobre seu estado de pronto e permaneça esperando por uma ordem de início de apresentação do fluxo. O mecanismo inter-fluxo deve esperar pela condição de pronto de todos os fluxos, para então ordenar simultaneamente o início de apresentação (*rendez-vous*). O mecanismo intra-fluxo deve considerar este tempo de espera adicional para o cálculo dos posteriores instantes de liberação.



Um outro problema concerne à sincronização do começo de envios dos fluxos pelos emissores, sob a suposição de relógios a derivas comuns. A definição de um tempo de início de sessão absoluto só faz sentido num sistema de relógios comuns, onde todos os emissores tem a mesma percepção do tempo. Uma solução proposta em [Carmo94b] fundamenta-se na existência de um mecanismo de *rendez-vous* gerado por uma camada superior, chamada *camada de orquestração*, destinada a coordenar a fase de introdução da aplicação distribuída.

A proposta de *Apresentação Condicional* [Carmo94a], introduzida na seção 2.5.2.3, corresponde a um exemplo de técnica de sincronização inter-fluxo que opera sob esta hipótese de relógios.

### **2.5.3.2 Sincronização de Fluxos Periódicos**

As técnicas de sincronização de fluxos periódicos no domínio inter-fluxo, também prescindem dos rótulos de tempo. A estratégia adotada aqui consiste geralmente na sincronização de cada fluxo em separado sendo que instante de liberação da primeira IU de cada fluxo deve ser comum. A técnica do *prefetch* é geralmente utilizada para a compensação do *jitter* de cada fluxo. O instante de liberação comum corresponde ao momento em que o *prefetch* de todos os fluxos tenha sido completado.

Esta estratégia pode ser adotada diretamente no caso de relógios a derivas comuns. Sob a hipótese de relógios a derivas constantes pode ser utilizado adicionalmente um emulador de período para cada fluxo.

Em [Courtiat95] apresenta-se um esquema de sincronização de lábios num sistema de relógios não sincronizados. O mecanismo intra-fluxo, utilizado para cada mídia, pressupõe conhecido o período nominal do fluxo e utiliza a técnica do *prefetch* para a compensação do *jitter*. Para a determinação do período do emissor é utilizada a estratégia de estimação convergente sugerida em [Dairaine94] (Veja na seção 2.4.1.2). Os ajustes de período no receptor são realizados através de ajustes de relógio do dispositivo de apresentação. A sincronização inter-fluxo é conseguida assim, com a sincronização do começo da apresentação dos fluxos (no momento em que ambos tenham completado seu correspondente *prefetch*) e com a sincronização dos fluxos de áudio e vídeo em separado pelo mecanismo de sincronização intra-fluxo mencionado. O esquema é especificado em *RT-LOTOS* através de módulos de sincronização intra-fluxo instanciados em paralelo, cujo começo de apresentação é sincronizado por uma ação *start*, indicando que o *prefetch* de cada fluxo já foi concluído.

## 2.6 A Sincronização Multidestino

A complexidade do problema faz com que, em geral, técnicas da abordagem temporal sejam adotadas neste caso. As condições de extensão desta abordagem ao caso multidestino são similares com aquelas estabelecidas para sua extensão ao caso inter-fluxo e podem ser formuladas assim:

1. Os emissores devem sincronizar o começo do envio dos fluxos, através da definição de uma referência de início de sessão comum.
2. Os atrasos de comunicação dos fluxos a sincronizar devem ser equalizados nos receptores. Isto é, deve ser definido um mesmo atraso para todos os fluxos em todos os receptores.

A condição (1) corresponde com aquela formulada para o domínio inter-fluxo, e já foi discutida. A condição (2) é acrescentada aqui para os diversos receptores. Entretanto é admitido freqüentemente, que a sincronização multidestino, pelo tipo de aplicação no qual é necessária (tele-conferências), requer menos exatidão que a sincronização inter-fluxo [Dairaine94]. Assim por exemplo, numa aplicação de sincronização de lábios é possível tolerar certa diferença nos atrasos de apresentação nos diferentes receptores. Entretanto a exatidão requerida para a sincronização dos fluxos de áudio e vídeo em cada destino em separado (requisitos de sincronização inter-fluxo) é mais restritiva.

### 2.6.1 Sincronização de Fluxos Estocásticos

Para este tipo de fluxos as técnicas de sincronização multidestino adotam uma estratégia baseada em *time-stamping*, com a definição de um atraso comum tal qual no domínio inter-fluxo. Entretanto, no caso da sincronização de fluxos estocásticos, a condição (2), dificilmente pode ser satisfeita em sistemas de relógios não sincronizados globalmente. Um problema adicional refere-se ao incremento dos requisitos de armazenamento de cada fluxo, como consequência da definição de um atraso comum, onde os fluxos mais rápidos devem aguardar pelos mais lentos em todos os receptores.

#### Protocolo de sincronização adaptativo

A proposta do protocolo de sincronização adaptativo [Escobar94], (introduzida na seção 2.5.3.1) baseada na sincronização global de relógios, corresponde ao tipo de mecanismo diretamente

aplicável neste domínio. Entretanto, nesta técnica não é resolvido o problema da sobrecarga dos receptores. Nas propostas de [Dairaine94] e [Carmo94b] aborda-se o problema da sobrecarga coordenando um certo tempo de armazenamento nos emissores para atrasar o envio dos fluxos mais rápidos até os instantes previstos. Desta maneira, a carga é repartida entre emissores e receptores, resultando isto num armazenamento mais eficiente, particularmente em esquemas de difusão seletiva. No caso de fluxos previamente armazenados, são gerenciados os tempos de aquisição dos fluxos, de maneira a evitar um tempo de armazenamento inútil no emissor.

### **Protocolo de Conversação Multi-fluxo**

Uma outra proposta [Yavatkar92] apresenta um protocolo chamado MCP (*Multiflow Conversation Protocol*) baseado na sincronização global de relógios, que provê a coordenação e sincronização temporal e causal necessárias em aplicações multimídia cooperativas, como é o caso dos sistemas de conferência.

Nesta proposta, um *fluxo (flow)* é entendido como uma conexão bidirecional entre dois ou mais participantes, sendo que a cada fluxo é associado o transporte de uma mídia. Assim, p. ex. um fluxo de voz ligando três participantes num sistema de conferência, permite a intervenção passiva e/ou ativa (ouvir e/ou falar) de quaisquer dos três participantes. Uma *conversação multi-fluxo*, resulta da composição de distintos fluxos relacionados entre si.

Para a obtenção da sincronização temporal e causal numa conversação multi-fluxo, é proposto o uso da noção de  $\Delta$ -causalidade, que associa à relação de causalidade as restrições temporais de atrasos das mídias transportadas pelos fluxos que constituem a conversação. Neste esquema cada mensagem tem associada uma informação de contexto e um rótulo de tempo. A informação de contexto contém a informação necessária para o fornecimento da sincronização causal enquanto os rótulos de tempos são utilizados para a obtenção da sincronização temporal.

O protocolo MCP oferece também mecanismos de coordenação dos participantes baseado em passagem de bastão, que permitem diferentes níveis de concorrência. Em um extremo, o controle de concorrência estrito (*strict floor concurrency control*) onde só existe um bastão na conversação sendo que a intervenção ativa de um participante é possível unicamente quando em posse do bastão. Em outro extremo, não é fornecido controle de concorrência algum, o qual corresponde ao nível de coordenação utilizado em aplicações cooperativas tipo *brainstorming*.

Pode-se afirmar finalmente que a sincronização global de relógios constitui a desvantagem principal de todos os esquemas apresentados nesta seção.

## 2.6.2 Sincronização de Fluxos Periódicos

Neste domínio a revisão bibliográfica permitiu apenas encontrar uma proposta de sincronização de fluxos periódicos (*Multiple Feedback Techniques*) [Ramanathan93], que opera em sistemas de relógios não sincronizados. A técnica corresponde a uma extensão da proposta intra-fluxo (introduzida na seção 2.4.1.2) a qual é utilizada aqui para fornecer a 'continuidade de cada fluxo por separado.

Para o fornecimento da sincronização inter-fluxo é definido um fluxo *master*, geralmente aquele com requisitos de sincronização intra-fluxo mais estritos. Adicionalmente as IUs possuem rótulos de tempo relativos (*RTS - Relative Time Stamps*) que representam seu tempo de apresentação em relação ao início da apresentação do fluxo. As mensagens de realimentação enviadas desde todos os receptores (midiafones) permitem estimar a dessincronização entre os fluxos. A ressincronização é então conseguida por operações *delete/duplicate* de eliminação e duplicação aplicadas oportunamente pelo emissor (servidor) nas IUs do fluxo *slave*.

Devido ao fato desta técnica fazer uso de um emissor centralizado não existe o problema da sincronização de emissores distintos. Por outro lado a sincronização de receptores é conseguida de maneira indireta através da ressincronização dos fluxos. Isto é, uma dessincronização inicial é inevitável, entretanto os fluxos podem ser sincronizados posteriormente.

As principais características desta técnica são as seguintes:

- Garante a continuidade de todos os fluxos porém, o fluxo *slave*, está sujeito a operações *delete/duplicate* para compensar possíveis perdas de sincronização inter-fluxo.
- A máxima assincronia inter-fluxo depende da exatidão de sua estimação, a qual por sua vez, depende do período de mensagens de realimentação.
- O início da apresentação de cada fluxo varia conforme o atraso atual da IU de *prefetch*. De maneira que o esquema não garante a sincronização do início da apresentação de cada fluxo.

- Não é aplicável em serviços de *multicast*, pois o ajuste de períodos é feito no emissor e depende da variação de apresentação do receptor em questão.
- Não suporta aplicações multimídia distribuídas de configurações N:1.

## 2.7 Conclusão

A introdução de mídias contínuas nos sistemas multimídia tem originado recentemente uma explosão de novas aplicações em diversos campos, destacando-se particularmente os campos de automação de escritórios, indústria de serviços, aplicações científicas, culturais e domésticas. Aplicações tais como tratamento de documentos multimídia, correio eletrônico multimídia, sistemas de conferências, ambientes cooperativos, sistemas de aprendizagem a distância e em geral, sistemas de informação multimídia são alguns exemplos.

A sincronização de informações ligadas a restrições temporais é um aspecto de grande importância em sistemas multimídia. Para tratar este aspecto, é necessária a participação integral de todos os componentes do sistema multimídia, assim como o desenvolvimento de novas técnicas e paradigmas computacionais.

Do ponto de vista da comunicação, o problema da sincronização pode ser abordado segundo três aspectos onde o fluxo de dados é considerado um componente básico: a sincronização intra-fluxo, a sincronização inter-fluxo e a sincronização multidestino, sendo que na literatura existem várias técnicas de sincronização propostas.

Assim, neste capítulo, foi apresentada uma visão geral da problemática da sincronização multimídia e de algumas das principais propostas. No seguinte capítulo, será abordada a problemática do desenvolvimento de sistemas multimídia segundo uma abordagem formal.

## Capítulo 3

### *Projeto e Desenvolvimento de Sistemas Multimídia*

#### **3.1 Introdução**

O processo de desenvolvimento de *software* confronta-se com a necessidade primordial de garantir uma implementação correta e eficiente dos requisitos fixados pelo usuário. Entretanto, a complexidade encontrada em certos sistemas, p. ex. distribuídos ou tempo-real, faz com que o fornecimento deste nível de qualidade só seja possível através do uso de métodos formais nas diversas etapas do desenvolvimento do *software*. Mais além, o uso destes métodos é o modo mais eficiente de garantir que certos fatores de qualidade tais como segurança, consistência e correção temporal, sejam de fato atingidos [Bucci95]. Este capítulo apresenta aspectos gerais do desenvolvimento de sistemas multimídia e justifica a escolha de Esterel como linguagem de especificação adotada nesta dissertação.

#### **3.2 As Técnicas de Descrição Formal (TDF)**

Nos últimos anos, os métodos formais ou *Técnicas de Descrição Formal (TDF)*, tem sido reconhecidos como cada vez mais importantes no desenvolvimento de sistemas complexos, a tal ponto que algumas dessas técnicas passaram a ser padronizadas pela ISO e pela ITU, como é o caso das linguagens Estelle, LOTOS e SDL.

A motivação principal para o uso das TDFs consiste em permitir uma especificação completa, clara e sem ambigüidades dos sistemas, e servir de base na construção de ferramentas que permitem automatizar atividades tais como: análise de comportamento lógico e do desempenho, verificação das especificações, implementação destas, e determinação da conformidade do *software* resultante em relação aos requisitos iniciais [Farines89].

As TDFs permitem assim, a detecção de erros em etapas iniciais do projeto, servem de base para a implementação facilitando sua modificação, manutenção e reusabilidade, e em geral, melhoram a qualidade do *software* resultante.

Existe hoje um grande número de abordagens diferentes para as TDFs, baseadas em formalismos diversos entre as quais podem-se destacar a Máquina de Estado Finita, Rede de Petri, Álgebras de Processos (p. ex. CCS), Lógica Temporal e formalismos híbridos. Todas estas abordagens procuram satisfazer através do formalismo escolhido e em níveis diversos, aspectos importantes associados a: *funcionalidade* (relações de causalidade, comunicação e sincronização, restrições de tempo real e representação de dados); *nível de abstração* (i. e. independência da implementação); *grau de formalismo* (o que resulta em níveis diferentes de validação, avaliação, prototipagem e compilação das especificações e possibilita a construção de ferramentas apropriadas); *características de estruturação* (que permitem a decomposição do sistemas em subsistemas).

Alguns dos aspectos mencionados são conflitantes e as TDFs hoje existentes acabam por priorizar um ou vários destes aspectos. As TDFs baseadas na Álgebra de Processos (CCS), por exemplo, favorecem o aspecto da verificação enquanto aquelas baseadas em modelos de transição e estados aproximam-se mais da implementação.

### 3.3 O Fator Tempo-Real nos Sistemas Multimídia

Como visto no capítulo anterior, uma das características mais relevantes dos sistemas multimídia é a introdução das mídias contínuas, e em decorrência, do fator *tempo-real*. Nos sistemas que exibem comportamento tempo-real, a correção do sistema depende tanto dos resultados lógicos produzidos pelo sistema, como do tempo (real) nos quais estes resultados são produzidos. Desta forma, é necessário preservar certas propriedades lógicas tais como ausência de *deadlocks* e *livelocks*, enquanto se mantém a correção temporal do sistema.

Diferentes, dos chamados *sistemas tempo-real crítico* (*Hard Real Time Systems*) onde o atraso de uma mensagem pode ser julgado de condição catastrófica, os sistemas multimídia podem ser classificados dentro dos *sistemas tempo-real não críticos* (*Soft Real Time Systems*) pois apresentam certo grau de flexibilidade no seu comportamento temporal. Neste tipo de sistemas uma perda dos requisitos temporais, como por exemplo numa aplicação de sincronização de lábios, não representa uma situação catastrófica, mas sim uma degradação na qualidade de serviço que pode ou não ser aceitável para o usuário.

A introdução do fator tempo faz com que a maior parte das FDTs, e entre elas as padronizadas, não sejam apropriadas para o desenvolvimento de sistemas multimídia, pois não possuem as

capacidades necessárias para a expressão de restrições temporais. Na sequência são apresentados alguns dos métodos formais atualmente utilizados no desenvolvimento de sistemas multimídia.

### **3.4 Os Métodos Formais no Desenvolvimento de Sistemas Multimídia**

As abordagens formais utilizadas no desenvolvimento de sistemas multimídia podem ser classificadas em duas categorias [Carmo94b]:

- As abordagens baseadas em extensões temporais do formalismo Rede de Petri que visam basicamente, a representação efetiva da informação multimídia com suas restrições de sincronização associadas.
- As abordagens baseadas em linguagens que além de representar a informação multimídia facilitam a descrição de procedimentos e algoritmos de tratamento desta.

Na primeira categoria, pode-se citar as seguintes técnicas:

1. Em [Little90] é proposto um modelo de Rede de Petri temporizado chamado *Object Composition Petri Net (OCPN)*, que permite a descrição de cenários de restituição e apresentação de objetos multimídia. A proposta parte da representação do tempo através de intervalos temporais [Allen83] que permitem especificar sete relações temporais fundamentais que caracterizam as distintas possibilidades de sincronização entre objetos multimídia. Neste modelo de Rede de Petri, cada lugar descreve um processo de apresentação de um objeto multimídia com sua duração de apresentação associada. Uma das principais limitações do modelo OCPN é sua incapacidade de representação das relações de sincronização não ideais inerentes aos objetos multimídia, tais como seu *jitter* de apresentação permissível.
2. Em [Diaz94] é proposto um modelo chamado *Time Stream Petri Net (TSPN)* que leva em conta as restrições do modelo OCPN, permitindo a representação das sete relações de sincronização fundamentais entre objetos multimídia considerando a especificação do *jitter* permissível de cada informação multimídia. Este modelo corresponde a uma extensão e integração do modelo OCPN e do modelo *Arc Time Petri Nets* [Walter83].

Na segunda categoria, se destacam as técnicas:



1. LOTOS [Bolognesi87] é uma álgebra de processos que permite a especificação de sistemas distribuídos com um alto grau de abstração. Entretanto, o modelo básico de álgebra de processos não é suficientemente adequado para modelar comportamentos tempo-real por não representar explicitamente o tempo. A problemática da especificação da sincronização multimídia assim tem sido abordada através de extensões temporizadas de LOTOS. Tal é o caso da linguagem RT-LOTOS utilizada em [Carmo94b] para a especificação do mecanismo de apresentação condicional intra-fluxo e inter-fluxo introduzidos nas seções 2.4.2 e 2.5.2.3, e em [Camargo95] para a especificação do algoritmo de sincronização de lábios. Uma outra abordagem dita LOTOS/QTL [Blair93] constitui uma abordagem dual que utiliza a linguagem LOTOS para a especificação do comportamento funcional, e a *lógica temporal quantitativa* QTL para a expressão de requisitos temporais, associados à informação multimídia.
2. Esterel [Berry88] é uma linguagem síncrona basicamente utilizada para a programação de sistemas reativos, cujas características serão apresentadas no item seguinte. Uma aplicação interessante de Esterel na problemática de sincronização multimídia é apresentada em [Stefani92], onde esta linguagem é utilizada na programação de *objetos reativos* que encapsulam o comportamento tempo-real necessário na sincronização de objetos multimídia, sendo adotado um modelo computacional distribuído que mistura as abordagens síncrona e assíncrona.

## **3.5 Esterel como Linguagem de Desenvolvimento**

### **3.5.1 Características Gerais de Esterel**

A linguagem Esterel é uma linguagem imperativa baseada na hipótese síncrona que supõe que um sistema reage de forma instantânea a eventos externos. A representação semântica de Esterel é simples, determinística e independente da plataforma, o que facilita a verificação e a prova deste tipo de sistemas. A satisfação desta hipótese em tempo de execução é possível se o tempo de reação do sistema é sempre inferior ao menor tempo entre eventos sucessivos [Berry88]. Os princípios de base da abordagem síncrona são os seguintes:

- **reatividade:** o modelo é reativo no sentido que se aplica a sistemas reativos, i.e. sistemas que mantêm interação contínua com seu ambiente externo; o sistema reativo entra em ação nos instantes onde recebe um estímulo externo.
- **sincronia:** a hipótese de sincronia significa que as reações são instantâneas, como se os programas fossem executados numa máquina de velocidade infinita; as reações são consideradas atômicas e em consequência não interferem entre si. Nesta abordagem não há concorrência entre as reações, o que elimina uma fonte de não-determinismo.
- **comunicação por difusão instantânea:** a comunicação entre componentes (ou módulos) segue o princípio da difusão instantânea; um sinal emitido é visto no mesmo instante da sua emissão por todos os receptores.
- **determinismo:** das características anteriores decorre que este modelo é determinista, o que tem como resultado a simplificação das programações reativas.

### 3.5.2 Esterel como Linguagem de Desenvolvimento: Uma Justificativa da Escolha

A linguagem Esterel (descrita mais em detalhe no Apêndice A) é adotada para a especificação de mecanismos de sincronização multimídia neste documento. As principais motivações para a escolha de Esterel são as seguintes:

- É uma linguagem de alto nível, definida com uma semântica formal precisa, apropriada para as fases de especificação e desenvolvimento de sistemas reativos tempo-real [Jagadeesan95].
- A abordagem síncrona permite uma abstração simples e elegante na modelagem dos problemas de concorrência e tempo-real nos sistemas multimídia. Assim por exemplo, os fluxos de dados podem ser modelados como sinais e eficientemente manuseados pelas construções temporais oferecidas pela linguagem. O tempo pode ser tratado como mais um sinal do sistema, e manuseado com as mesmas construções temporais dos outros sinais resultando isto num tratamento homogêneo dos aspectos temporais da especificação.

- As técnicas de compilação de especificações Esterel geram autômatos de estado finito deterministas nos quais o paralelismo e a comunicação, expressos no formalismo, desaparecem; em decorrência disto, estes autômatos apresentam um grau elevado de eficiência e previsibilidade temporal.
- Permite provas e verificação das especificações em Esterel sobre o autômato que é gerado pela especificação e que será executado, e não simplesmente sobre a especificação, como em outras técnicas, seguindo o lema WYPIWYE (*What You Prove Is What You Execute*) [Berry88].
- A forte orientação de Esterel à implementação faz com que a especificação de sistemas requer a identificação e definição de todos os elementos necessários para a geração de um código executável. Esta exigência pode representar um enorme ganho em experiência e conhecimento das questões ligadas à implementação de técnicas de sincronização multimídia.

Cabe mencionar que as implementações geradas a partir de especificações que seguem a abordagem síncrona apresentam as mesmas características de eficiência e previsibilidade anteriormente citadas, desde que a hipótese de aplicação desta abordagem continua verificada em tempo de execução.

Finalmente, encontram-se dificuldades em aplicar um modelo estritamente síncrono em caso de aplicações distribuídas. O modelo computacional apresentado em [Stefani92], que mistura elementos das abordagens síncronas e assíncronas, representa uma alternativa para a aplicação do modelo síncrono ao caso de sistemas distribuídos.

### 3.5.3 O Ambiente de Desenvolvimento para a Linguagem Esterel

O compilador **Esterel versão 4.41** foi utilizado para a compilação das especificações neste trabalho. A compilação produz a implementação em linguagem C do autômato que representa a especificação. O código gerado requer um certo código auxiliar, ou **interface de tratamento de dados**, que define os tipos, constantes, funções e procedimentos declarados na especificação. Para a execução direta, requer-se adicionalmente do código de **interface de execução**, que

detecta sinais de entrada, ativa o autômato e realiza ações de saída. O código da interface de manuseio de dados e de execução deve ser escrito pelo usuário.

A versão 4.41 de Esterel também permite a simulação interativa do autômato gerado pelo compilador, através da sua extensão com um certo código de simulação. Este código estendido é também gerado pelo compilador, e deve ser ligado (*linked*) com um código adicional de simulação, contido numa biblioteca, para gerar um arquivo executável ou simulador. Desta forma, o código simulado é igual ao código executável, seguindo o lema WYPIWYE (*What You Prove Is What You Execute*) [Berry88].

No processo de simulação o usuário gera sinais de entrada e o simulador responde apresentando sinais de saída e informação opcional sobre o estado interno do autômato, sinais internos e valor de variáveis, conforme solicitado pelo comando *trace*.

Alternativamente, é possível utilizar para a simulação o autômato gerado pelo compilador (em código C), acrescentado com uma interface de execução escrita pelo usuário, p. ex., encarregada de gerar estatísticas.

### 3.5.4 A Metodologia Proposta

A metodologia proposta é baseada no princípio WYPIWYE (*What You Prove Is What You Execute*) e compreende três atividades básicas: modelagem, validação e avaliação de desempenho.

A modelagem consiste na especificação formal do comportamento do sistema multimídia. Esta atividade preocupa-se inicialmente pela partição funcional do sistema. Para isto é fundamental, a identificação de comportamentos genéricos na forma de componentes básicos, que estimulem a reutilização e facilitem a composição correta e completa do sistema global. Deve ser definida uma interface destes componentes, que permita refletir diferentes condições de trabalho, p. ex. através da sua parametrização. O interesse principal da modelagem é centrado na concepção de módulos sincronizadores que encapsulem o comportamento tempo-real necessário para esta função.

A validação do comportamento consiste em provar a correção da especificação e da conseqüente futura implementação. Propõe-se, neste trabalho realizar a validação através do processo de simulação, dada a forte orientação da linguagem Esterel à implementação e a equivalência entre código gerado e especificação Esterel. A simulação deve se preocupar inicialmente com a prova individual de componentes básicos, para posteriormente abordar a prova do sistema global,

visando especialmente a validação do comportamento dos módulos sincronizadores. Nesta atividade devem ser definidos todos os componentes necessários para a geração de um código simulador, tais como funções, tipos de dados, etc., conforme declarados na especificação Esterel.

A avaliação de desempenho consiste na determinação da performance do sistema sob diferentes condições de trabalho. Esta atividade também é realizada por simulação e deve preocupar-se principalmente pela avaliação da qualidade de serviço resultante, gerando estatísticas que permitam uma análise apropriada.

A integração destas três atividades compreende um processo iterativo que leva à autorização da implementação uma vez verificados a satisfação os requisitos e restrições de concepção iniciais.

### **3.6 Conclusão**

Neste capítulo foi apresentada uma visão geral do problema do desenvolvimento formal de sistemas multimídia. Num primeiro lugar foram mostradas as principais motivações para a utilização de TDFs no desenvolvimento formal de *software*. No caso de sistemas multimídia, a presença das mídias contínuas nos levou a considerar um subconjunto de TDFs capazes de expressar as restrições de tempo-real inerentes à representação e tratamento destas mídias. São apresentadas algumas considerações que nos levaram finalmente a adotar Esterel como linguagem de especificação no desenvolvimento de sistemas multimídia baseados numa metodologia proposta neste documento.

## 4. Capítulo 4

### Modelagem em Esterel de Técnicas de Sincronização Multimídia

#### 4.1 Introdução

Neste capítulo, é mostrada a utilização da linguagem Esterel na especificação de problemas de sincronização em sistemas multimídia distribuídos. Inicialmente é abordado o problema da sincronização intra-fluxo. Uma primeira especificação de requisitos gerais permite a definição de vários cenários particulares de problemas de sincronização neste domínio. Em seguida é apresentada a modelagem de cada um dos cenários de sincronização intra-fluxo. A seguir, apresentam-se de forma similar, cenários e modelos respectivos, nos domínios de sincronização inter-fluxo e multideestino. Finalmente, são apresentadas algumas conclusões sobre a representação em Esterel.

#### 4.2 Modelagem de Técnicas de Sincronização Intra-Fluxo

##### 4.2.1 Características Gerais do Problema:

Conforme uma visão mais abstrata do problema de sincronização intra-fluxo, podem-se formular os seguintes requisitos gerais do sistema.

Requer-se a sincronização da apresentação de uma mídia contínua gerada num emissor remoto. A sincronização deve respeitar certas exigências de qualidade de serviço de apresentação, isto é, certos valores de máximo *jitter* e máxima latência (atraso fim-a-fim). A seguir são apresentadas as hipóteses do problema e os elementos do sistema (Figura 4.1):

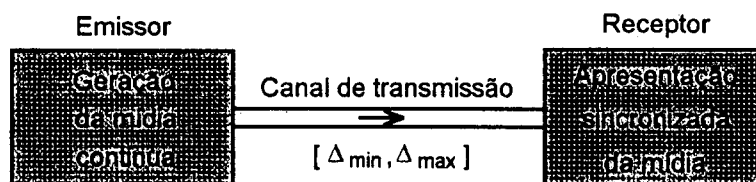


Figura 4.1 Apresentação remota de uma mídia contínua.

- **O Emissor**, gera uma mídia contínua na forma de um fluxo de dados interrelacionados temporalmente.
- **O Canal de Transmissão**, oferece garantias de qualidade de serviço tais como: garantia de largura de banda, limites inferior e superior de atrasos (na Figura 4.1, definidos como:  $\Delta_{\min}$  e  $\Delta_{\max}$ ), perda de pacotes limitada, e seqüenciamento. Outrossim, o canal introduz um *jitter*, na transmissão de dados.
- **O Receptor**, visa a apresentação sincronizada da mídia contínua, quer dizer, a conservação das relações temporais dos dados gerados pelo emissor remoto, respeitando os requisitos de qualidade de serviço impostos pela aplicação. Em particular, faz-se necessário compensar o *jitter*, introduzido pelo canal de transmissão, e adicionalmente respeitar a qualidade de interatividade definida pela latência máxima de apresentação permitida.

Uma especificação mais concreta do sistema refere-se à hipótese de sincronização de seus relógios, elemento que em particular, determina as diferentes soluções do problema. Os requisitos atendidos assim variam, sendo que uma distinção especial é dada àquelas soluções que suportam exclusivamente fluxos periódicos e outras que, em geral, suportam qualquer tipo de fluxo.

#### 4.2.2 Os Cenários de Sincronização Intra-Fluxo

Nesta seção são apresentados alguns dos cenários correspondentes aos principais problemas de sincronização intra-fluxo. São apresentados cinco cenários distintos que reúnem requisitos sobre a qualidade de serviço e o tipo de mídia a sincronizar, assim como algumas restrições sobre o tipo de relógios do sistema. Paralelamente, são associadas certas soluções do problema de sincronização intra-fluxo para os cinco cenários. As soluções do problema de sincronização multimídia apresentadas, variam conforme a hipótese de sincronização de relógios do sistema; a medida que a hipótese de relógios é relaxada, os requisitos atendidos são cada vez mais restritivos.

### 4.2.2.1 Primeiro Cenário

#### Especificação e Requisitos do Sistema

Requer-se a sincronização da apresentação de uma mídia contínua gerada remotamente. O fluxo gerado não é necessariamente periódico. O canal de transmissão é considerado como totalmente confiável, e só garante um atraso de transmissão no intervalo  $[\Delta_{\min}, \Delta_{\max}]$ . Não é desejada a ocorrência de *jitter* na comunicação, e deve-se respeitar certa latência máxima de apresentação. Trata-se de um sistema de relógios comuns, i.e. relógios com mesma base  $b$  e deriva  $d$ . A estrutura deste sistema é apresentada na Figura 4.2.

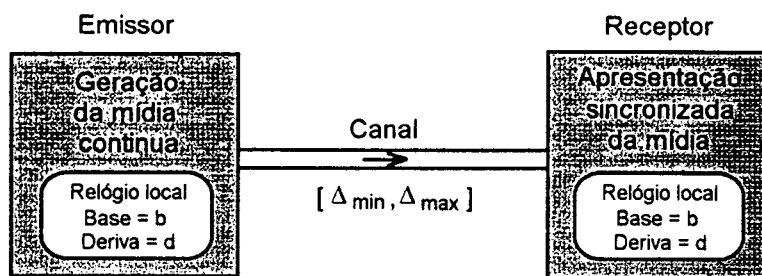


Figura 4.2 Apresentação de uma mídia contínua num sistema de relógios comuns.

#### Solução do Problema

O mecanismo de sincronização baseado em *time-stamping*, proposto em [Santoso93] e introduzido na seção 2.4.1.1, foi escolhido para a solução deste problema.

Uma vez que o sistema possui um tempo global comum, a sincronização neste esquema é conseguida a partir de uma idéia simples. A entidade de sincronização emissora rotula cada unidade de informação com o tempo corrente no instante de seu envio. A entidade receptora, responsável da sincronização, armazena e entrega cada unidade de informação para apresentação, só após um tempo de apresentação alvo (*TPT - Target Presentation Time*), relativo a seu rótulo de tempo. O tempo de apresentação alvo, *TPT*, é fixo e igual ao máximo atraso fim-a-fim,  $\Delta_{\max}$ .

#### Atendimento dos Requisitos:

- Neste esquema, nenhuma suposição é feita sobre a natureza das relações temporais entre os dados do fluxo. Isto é, o fluxo a sincronizar pode ser estocástico ou determinista.
- Este esquema garante a não ocorrência de *jitter* na transmissão do fluxo.



- Neste esquema, a latência de apresentação é definida, e corresponde ao valor do tempo de apresentação alvo, aqui fixado ao valor do máximo atraso fim-a-fim. Assim, o requisito de latência máxima permitida é satisfeito sob a condição do canal garantir um atraso máximo, que seja menor ou igual à máxima latência permitida pela aplicação.

#### 4.2.2.2 Segundo Cenário

##### Especificação e Requisitos do Sistema

Este cenário difere do anterior pelo fato de se tratar de um sistema de relógios a derivas comuns. A estrutura deste sistema é apresentada na Figura 4.3.

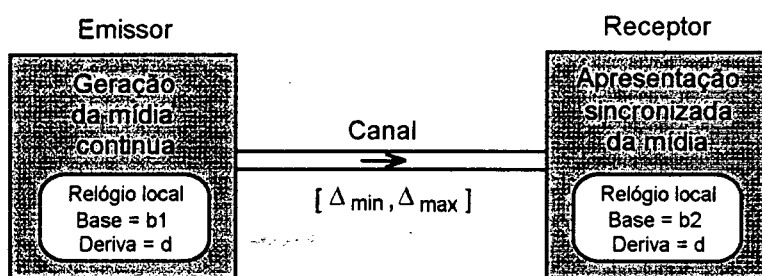


Figura 4.3 Apresentação de uma mídia contínua num sistema de relógios a derivas comuns.

##### Solução do Problema

O mecanismo de sincronização baseado em *time-stamping* proposto em [Santoso93] e introduzido na seção 2.4.1.1, foi escolhido para a solução deste problema. Neste esquema, a entidade de sincronização emissora rotula as IUs com o tempo corrente, segundo seu relógio local, no instante de seu envio. Entretanto, devido à hipótese de relógios a derivas comuns onde o tempo de emissor e receptor difere numa constante, a entidade de sincronização só pode calcular o tempo de apresentação de uma  $IUn$ , segundo o sua imediata anterior,  $IUn-1$ . Este tempo vem sendo dado pela diferença dos rótulos de tempo da  $IUn$  e  $IUn-1$ , mais o tempo de apresentação da  $IUn-1$ . Devido à impossibilidade do cálculo do tempo de apresentação da primeira IU, isto é, de seu atraso atual, o mecanismo assume um atraso atual igual a  $\Delta_{\min}$ , de maneira a introduzir um atraso inicial de compensação máximo ( $TPT - \Delta_{\min}$ ) e assegurar a compensação do *jitter*. O tempo de apresentação alvo  $TPT$  é fixo e igual ao máximo atraso fim-a-fim,  $\Delta_{\max}$ .

##### Atendimento dos Requisitos:

- Suporta-se a sincronização de fluxos estocásticos ou deterministas.

- O mecanismo garante a não ocorrência de *jitter* na transmissão do fluxo.
- Como consequência do relaxamento da hipótese de relógios, o mecanismo introduz uma imprecisão na latência de apresentação. Esta incerteza está compreendida no intervalo  $[\Delta_{\max}, 2*\Delta_{\max}-\Delta_{\min}]$ . Assim, o requisito de máxima latência permitida é só estritamente satisfeito no caso do canal garantir atrasos máximo e mínimo, para os quais o limite superior do intervalo de incerteza, seja ainda menor ou igual à máxima latência permitida pela aplicação. Por exemplo, os valores de atrasos mínimo e máximo 10 e 15 ms, implicam o intervalo de incerteza de latência [15, 20] ms. Neste caso só os requisitos de latências iguais o superiores a 20 ms seriam estritamente satisfeitos pelo mecanismo.

#### 4.2.2.3 Terceiro Cenário

##### Especificação e Requisitos do Sistema

Este cenário difere do anterior (segundo cenário) apenas por se tratar da apresentação de um fluxo periódico. A estrutura deste sistema é apresentada na Figura 4.4.

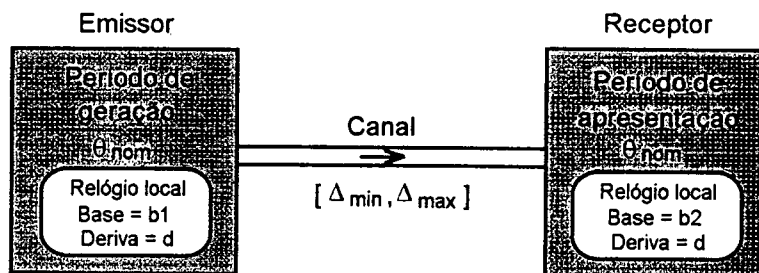


Figura 4.4 Apresentação de um fluxo periódico num sistema de relógios a derivas comuns.

##### Solução do Problema

A técnica do *prefetch*, apresentada em [Dairaine94] e introduzida na seção 2.4.1.2, oferece uma solução a este problema. Neste esquema, é apenas necessário que o receptor ou entidade de sincronização receptora, armazene um certo número de IUs, antes da sua entrega a intervalos constantes e iguais ao período nominal do fluxo. Desta forma, introduz-se um atraso inicial que compensa o *jitter* de rede, para evitar a possibilidade de fome no receptor.

O valor do *prefetch* (P) corresponde ao número de IUs que permanecem armazenadas exatamente no início da apresentação e vem sendo dado por:

$$P = \lceil (\Delta_{\max} - \Delta_{\min}) / \theta_{\text{nom}} \rceil$$

Onde  $\theta_{\text{nom}}$  denota o período nominal do fluxo e a notação  $\lceil x \rceil$  indica o valor inteiro superior ou igual a  $x$ .

#### Atendimento dos Requisitos:

- Neste esquema, o fluxo a sincronizar deve ser periódico.
- Garante-se a continuidade do fluxo, isto é, a não ocorrência de *jitter* de comunicação.
- Neste esquema, a latência de apresentação varia segundo o atraso atual da unidade de informação correspondente ao valor do *prefetch*. Assim, a latência de apresentação está compreendida no intervalo  $[\theta_{\text{nom}} * P + \Delta_{\min}, \theta_{\text{nom}} * P + \Delta_{\max}]$ .

#### 4.2.2.4 Quarto Cenário

##### Especificação e Requisitos do Sistema

Este cenário difere do anterior por se tratar de um sistema de relógios a derivas constantes. Em consequência, o período de geração do fluxo ( $\theta_{\text{in}}$ ), difere do período da sua apresentação ( $\theta_{\text{out}}$ ).

A estrutura deste sistema é apresentada na Figura 4.5.

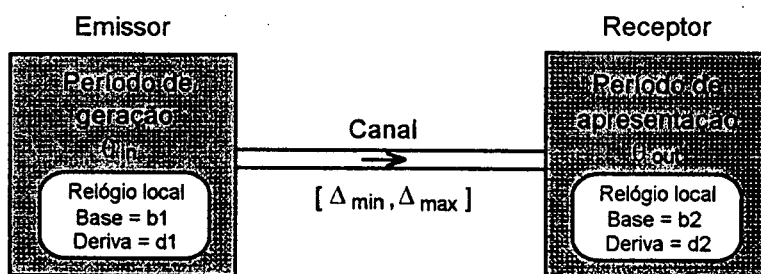


Figura 4.5 Apresentação de um fluxo periódico num sistema de relógios a derivas constantes.

#### Solução do Problema

A técnica do emulador de período proposta em [Dairaine94] e introduzida na seção 2.4.1.2, fornece a sincronização de fluxos periódicos onde os períodos de produção e consumo são diferentes mas constantes (sistema de relógios a derivas constantes).

O *emulador de período* evita a saturação ou fome do *buffer* receptor com operações periódicas: eliminar ou duplicar (*delete/duplicate*). Estas operações são realizadas sobre as IUs recebidas segundo seja o caso da diferença de períodos entre emissor e receptor. No caso de um período de produção,  $\theta_{in}$ , menor que o período de consumo,  $\theta_{out}$ , ou seja, a situação onde a saturação do *buffer* é iminente, o emulador elimina do *buffer* uma IU por vez a intervalos de  $\theta_{emu} = (\theta_{out} * \theta_{in}) / (\theta_{out} - \theta_{in})$ . No caso de um período de produção maior que o período de consumo, i.e. na situação de potencial fome no receptor, o emulador duplica uma IU por vez a intervalos dados por  $\theta_{emu} = (\theta_{in} * \theta_{out}) / (\theta_{in} - \theta_{out})$ . Estas operações *delete* ou *duplicate* conseguem respectivamente os efeitos de dilatação ou compressão do fluxo [Carmo93].

Adicionalmente, é utilizada a técnica do *prefetch* para compensar o *jitter* de rede.

#### Atendimento dos Requisitos:

- Neste esquema, o fluxo a sincronizar deve ser periódico.
- Garante-se a continuidade do fluxo, isto é, a não ocorrência de saturação ou fome no *buffer* receptor.
- Fornece-se a compensação da diferença de períodos através de operações periódicas *delete* ou *duplicate* conseguindo-se o efeito respectivo de dilatação ou compressão do fluxo.
- Neste esquema, a latência de apresentação varia segundo o atraso atual da unidade de informação correspondente ao valor do *prefetch* ( $P$ ). Assim, a latência de apresentação está compreendida no intervalo  $[\theta_{in} * P + \Delta_{min}, \theta_{in} * P + \Delta_{max}]$ .

#### 4.2.2.5 Quinto Cenário

##### Especificação e Requisitos do Sistema

Requer-se a sincronização da apresentação de um fluxo periódico gerado num emissor remoto, sendo que o período de apresentação do dispositivo receptor, varia dentro dos limites  $[\theta_{min}, \theta_{max}]$ . O canal de transmissão é considerado como totalmente confiável, e garante um atraso de transmissão do fluxo no intervalo  $[\Delta_{min}, \Delta_{max}]$ . Não são desejáveis *gaps* na

apresentação, isto é, exige-se a continuidade do fluxo, sendo que é tolerado o *jitter* provocado pela variação de período do dispositivo de apresentação, e adicionalmente, um certo valor máximo da latência na apresentação. Trata-se de um sistema de relógios não sincronizados. A estrutura deste sistema é apresentada na Figura 4.6.

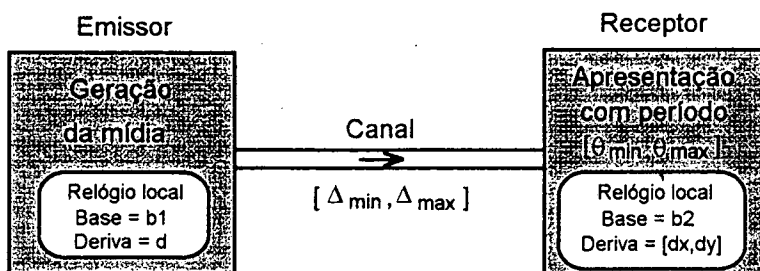


Figura 4.6 Apresentação de um fluxo periódico num sistema de relógios não sincronizados.

### Solução do Problema

O mecanismo apresentado em [Ramanathan93] e introduzido na seção 2.4.1.2, garante a sincronização de um fluxo periódico na presença do *jitter* de rede e de variações não determinísticas do período de apresentação do receptor. Assume-se que o receptor, (neste artigo chamado de *midiafone*), é um dispositivo com a mínima capacidade de apresentação e sem a sofisticação para implementar algum mecanismo de sincronização. Neste esquema, um emissor (Servidor Multimídia) fornece a sincronização utilizando pequenas mensagens de realimentação, enviados desde o receptor, que lhe permitem estimar os tempos atuais de apresentação de cada unidade de informação. Baseado nesta informação o emissor determina os tempos mínimo e máximo de envio de IUs posteriores visando evitar saturação ou fome no *buffer* do receptor. Assim, este esquema garante a continuidade do fluxo mas precisa de um canal adicional para o transporte de mensagens de realimentação.

### Atendimento dos Requisitos:

- Neste esquema, o fluxo a sincronizar deve ser periódico.
- Garante-se a continuidade do fluxo, sendo que a ocorrência de *jitter* na apresentação é inevitável, e depende da variação inerente ao dispositivo de apresentação do fluxo. Entretanto, este *drift* é geralmente desprezível.

- A latência de apresentação varia segundo o atraso atual da unidade de informação correspondente ao valor do *prefetch*.

### 4.2.3 Modelagem dos Módulos Genéricos

Esta seção apresenta a modelagem dos módulos genéricos necessários para todos os cenários.

#### 4.2.3.1 O Modelo do Canal de Transmissão

O canal de transmissão foi concebido como um canal confiável (sem perda, corrupção ou reordenação dos dados). O comportamento deste canal é modelado como uma fila que suporta a ocorrência de entradas e saídas simultâneas. O atraso atual dos dados, segue uma função de distribuição uniforme dentro dos limites mínimo e máximo exigidos. Pressupõe-se que o canal oferece suficiente largura de banda.

#### A. Estrutura do Modelo do Canal de Transmissão

O canal de transmissão foi modelado como um módulo Esterel, composto de quatro componentes paralelos, neste documento chamados de *threads*. A estrutura do modelo do canal de transmissão é mostrada na Figura 4.7.

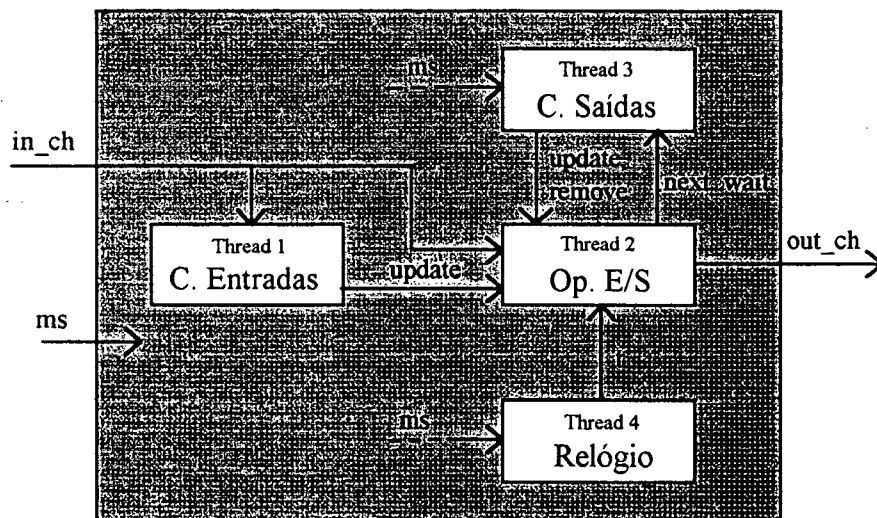


Figura 4.7 Estrutura do modelo do Canal de Transmissão.

A interface externa (entrada e saída) do módulo vem sendo dada pelos sinais:

- *in\_ch*, sinal de entrada que indica a chegada de uma IU ao canal de transmissão.
- *out\_ch*, sinal de saída do tipo *elt\_type*. Indica a saída de uma IU do canal.

- *ms*, sinal periódico de entrada, gerado cada milissegundo por um relógio externo ao sistema.

Para a comunicação entre *threads* são utilizados os seguintes sinais internos:

- *update*, sinal que indica a atualização da fila. Este sinal é emitido para cada saída ou entrada de IUs na fila do canal.
- *remove*, sinal de indicação do instante de saída de uma IU do canal e assim, da fila. Este sinal é emitido simultaneamente com o sinal *update*.
- *next\_wait*, sinal que quantifica o tempo de espera do elemento na frente da fila.
- *t*, sinal que indica o tempo atual do relógio interno.

O comportamento de cada *thread* pode ser descrito assim:

- 1) O primeiro *thread*, encarrega-se de coordenar a entrada das IUs na fila emitindo um sinal (*update*) para cada chegada ao canal, isto é, cada vez que um sinal externo de entrada (*in\_ch*) for emitido.
- 2) O segundo *thread*, tem duas funções: *i*) manusear diretamente a fila (*queue*), operando a entrada e saída de cada IU e *ii*) calcular seu atraso atual. Este *thread* é ativado cada vez que o sinal *update* é emitido.
- 3) O terceiro *thread*, encarrega-se de coordenar a saída de IUs da fila fazendo as vezes de temporizador, isto é, para cada emissão do sinal *next\_wait*, o *thread* aguarda um tempo equivalente ao valor deste sinal, e imediatamente emite os sinais *update* e *remove*, os quais produzem a saída da IU em questão.
- 4) O quarto *thread*, corresponde à instanciação de um relógio local, que fornece o valor do tempo atual utilizado para o cálculo dos tempos de espera. O relógio é reconhecido como mais um módulo de uso geral e será apresentado na seguinte seção.

## B. Especificação Completa do Canal de Transmissão

module Canal:

constant

D\_MIN, D\_MAX, SIZE\_CH: integer,  
TRUE, FALSE: boolean;

```

type
  elt_type, queue_type, queue_type_int;
input
  in_ch(elt_type), ms;
output
  out_ch(elt_type);
function
  new_q(integer): queue_type,
  top_q(queue_type): elt_type,
  empty_q(queue_type): boolean,
  new_q_int(integer): queue_type_int,
  top_q_int(queue_type_int): integer,
  calc_delay(integer, integer, integer, integer): integer;
procedure
  in_q (queue_type) (elt_type),
  out_q (queue_type) (),
  in_q_int (queue_type_int) (integer),
  out_q_int (queue_type_int) ();

signal update, remove, next_wait(integer), t(integer) in
  every in_ch do emit update end          % thread coordenador de entradas no canal

||

var queue: queue_type,                    % thread de operações de entrada e saída
    t_queue: queue_type_int,              % e cálculo de atrasos atuais.
    new_wait: boolean in
queue:= new_q(SIZE_CH+1); t_queue:= new_q_int(SIZE_CH+1);
every update do
  new_wait:= FALSE;
  present in_ch then
    if (empty_q(queue)) then new_wait:= TRUE end;
    call in_q (queue) (?in_ch); call in_q_int (t_queue) (?t)
  end;
  present remove then
    emit out_ch(top_q(queue));
    call out_q (queue) (); call out_q_int (t_queue) ();
    if (not(empty_q(queue))) then new_wait:= TRUE end
  end;
  if (new_wait) then
    emit next_wait(calc_delay(top_q_int(t_queue), ?t, D_MIN, D_MAX))
  end
end every
end var

||

loop                                     % thread coordenador de saídas do canal
  await immediate next_wait;
  await (?next_wait) ms;
  emit update; emit remove
end

||

copymodule Relogio [signal t/curr_time]   % thread relógio

```



```
end signal
end module
```

Na seqüência, de forma a completar a descrição do canal de transmissão, descrevem-se as declarações utilizadas.

### C. Declarações

As constantes utilizadas são as seguintes:

- *D\_MIN* e *D\_MAX*, indicam respectivamente os atrasos mínimo e máximo de transmissão.
- *SIZE\_CH*, indica o comprimento da fila do canal.

Três tipos de dados são utilizados:

- *elt\_type*, representa o tipo de unidade de informação do fluxo a transmitir.
- *queue\_type*, caracteriza o tipo de fila que armazena dados tipo *elt\_type*. Esta fila é definida externamente como uma fila finita, de elementos de tamanho fixo.
- *queue\_type\_int*, é um tipo de fila auxiliar, que armazena os tempos de entrada das IUs na fila, utilizados para o posterior cálculo de seus atrasos.

Finalmente, as seguintes funções e procedimentos são ainda utilizados como suporte para a especificação:

- *new\_q()*, permite criar uma fila de dados tipo *elt\_type*.
- *top\_q()*, consulta o elemento na frente da fila.
- *empty\_q()*, consulta o estado da fila: vazia ?
- *new\_q\_int()*, permite criar uma fila de dados tipo *integer*.
- *top\_q\_int()*, consulta o elemento na frente da fila de inteiros.
- *calc\_delay()*, calcula os atrasos dos dados enviados pelo canal respeitando os limites de atraso mínimo e máximo exigidos, e a correta seqüência dos dados. Os atrasos atuais são calculados conforme uma função de distribuição uniforme.

- *in\_q()*, insere na fila uma unidade de informação tipo *elt\_type*.
- *out\_q()*, extrai da fila uma unidade de informação tipo *elt\_type*.
- *in\_q\_int()*, insere um dado na fila de inteiros.
- *out\_q\_int()*, extrai um dado da fila de inteiros.

#### 4.2.3.2 O Modelo do Relógio

O relógio é modelado, em realidade, apenas como um contador de sinais periódicos fornecidos por um relógio externo ao sistema. O tempo atual fornecido pelo relógio vem sendo dado assim pelo número corrente de *ticks* recebidos mais uma base de tempo própria do relógio.

Este módulo recebe como entrada o sinal periódico *ms*, ou *tick* de milisegundos, emitido pelo relógio externo. O sinal de saída *curr\_time* contém o valor do tempo atual em milisegundos. O relógio utiliza como base de tempo o parâmetro *CLOCK\_SEED*.

#### Especificação do Relógio

```

module Relogio:
constant
  CLOCK_SEED: integer;
input
  ms;
output
  curr_time(integer);
var cur:= CLOCK_SEED: integer in
  every ms do
    cur:= cur + 1;
    emit curr_time(cur)
  end
end var
end module

```

### 4.2.4 Modelagem dos Cenários de Sincronização Intra-Fluxo

#### 4.2.4.1 O Modelo do Primeiro Cenário

##### Estrutura da Especificação

A estrutura modular da especificação em Esterel é mostrada na Figura 4.8. Neste modelo do sistema tem-se os seguintes módulos:

- Emissor, que gera um fluxo, rotulando cada unidade de informação, segundo o tempo fornecido por seu relógio local cuja base de tempo é  $b$ .
- Canal de transmissão, que introduz variações de atraso no intervalo  $[\Delta_{\min}, \Delta_{\max}]$ , e cujo modelo já foi apresentado.
- Sincronizador, que reconstrui o fluxo através da compensação do *jitter* de rede, baseado na percepção de tempo de seu relógio local, cuja base de tempo é  $b$ .

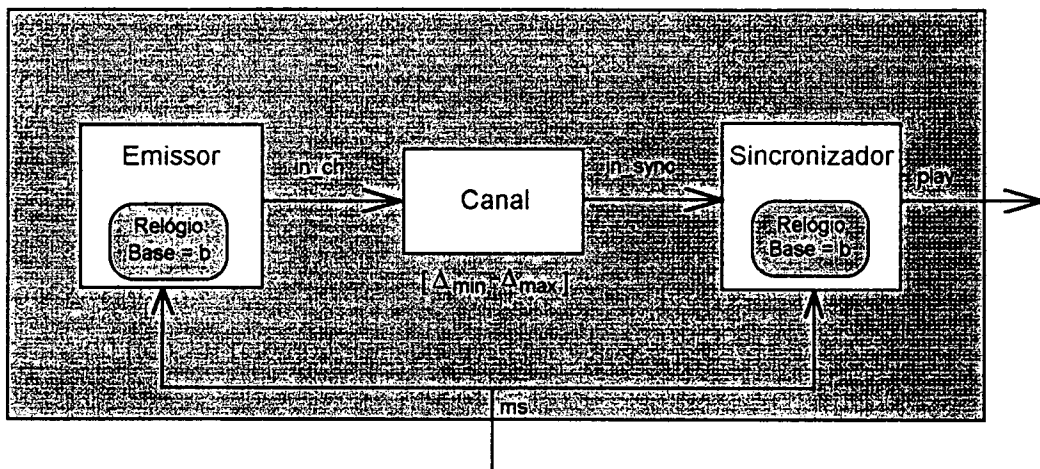


Figura 4.8 Estrutura modular Esterel do primeiro cenário de sincronização intra-fluxo.

A comunicação entre os módulos é a seguinte:

Para cada item de informação a transmitir, o emissor envia o sinal *in\_ch* ao canal, carregando a IU rotulada com o tempo corrente. O sinal *in\_sync* é emitido pelo canal ao sincronizador uma vez finalizada a transmissão. Oportunamente o sincronizador, no receptor, emite o sinal *play* carregando a IU a ser apresentada num dispositivo de apresentação externo ao sistema. O sistema recebe como entrada o sinal periódico *ms* (*tick* de milisegundos) o qual é utilizado pelos relógios locais de emissor e receptor para fornecer o tempo atual.

### Modelagem do Emissor

A especificação do emissor é simples. Neste caso o emissor modela o envio de um fluxo de período TETA. Por simplicidade é representado a rotulagem das IUs dentro do mesmo emissor (tarefa que em verdade deve ser executada pela entidade de sincronização emissora). Pressupõe-se que o tipo de dado *elt\_type* possui um campo adicional para armazenar o rótulo de tempo.

O módulo tem como entrada o sinal periódico *ms*, representando a passagem do tempo em milissegundos. O sinal *out\_src* é utilizado como sinal de saída e contém, como informação, as IUs.

Adicionalmente é utilizado o sinal interno *t*, que representa o tempo atual no emissor. Este sinal é produzido por um relógio local que corresponde a uma instância do módulo genérico Relógio. Este relógio local possui uma base de tempo, *SOURCE\_CLOCK\_SEED*, neste caso igual à base de tempo do relógio do receptor, *SINK\_CLOCK\_SEED*, cujo valor é *b* (veja Figura 4.8). Uma vez que ambos relógios tem a mesma base (*b*), e compartilham o sinal de entrada (*ms*), i. e. possuem a mesma frequência, o modelo é consistente com a hipótese de relógios comuns.

O corpo da especificação do emissor é composto pela seqüência de ações: obter, carimbar e enviar cada elemento, seguida de uma espera equivalente ao período do fluxo. Esta seqüência é repetida continuamente. A função *get\_elt()* permite obter as IUs, enquanto o procedimento *write\_stamp()* é utilizado para carimbar as IUs com o tempo atual *t*, no instante de seu envio.

A especificação do Emissor é a seguinte:

```

module Emissor:
  constant
    TETA,
    SOURCE_CLOCK_SEED: integer;
  type
    elt_type;
  input
    ms;
  output
    out_src(elt_type);
  function
    get_elt(): elt_type;
  procedure
    write_stamp (elt_type) (integer);

  signal t(integer) in
    var elt: elt_type in
      await ms;
      loop
        elt:= get_elt();
        call write_stamp (elt) (?t);
        emit out_src(elt);
        await TETA ms
      end
    end var
  ||
  copymodule Relógio [signal t/curr_time;
                    constant SOURCE_CLOCK_SEED/CLOCK_SEED]

end signal
end module

```

## Modelagem do Sincronizador

O comportamento do sincronizador neste caso é similar àquele representado no caso do canal de transmissão, no sentido de que a compensação do *jitter* de rede, é conseguida através da retenção das IUs, num *buffer* de política *FIFO* (*First-In-First-Out*), até um certo instante de liberação. O sincronizador deve também suportar a possibilidade de entradas e saídas simultâneas de IUs. Entretanto, no sincronizador, uma unidade de informação pode precisar ser entregue no mesmo instante de sua chegada quando experimenta um atraso atual igual ao valor do tempo de apresentação alvo, *TPT*. O cálculo do tempo de retenção das IUs, no sincronizador, é baseado nos rótulos de tempo que lhes acompanha. Desta forma, não é necessária uma fila auxiliar como aquela utilizada pelo canal de transmissão.

### A. Estrutura do Modelo do Sincronizador

A estrutura do modelo do sincronizador é mostrada na Figura 4.9.

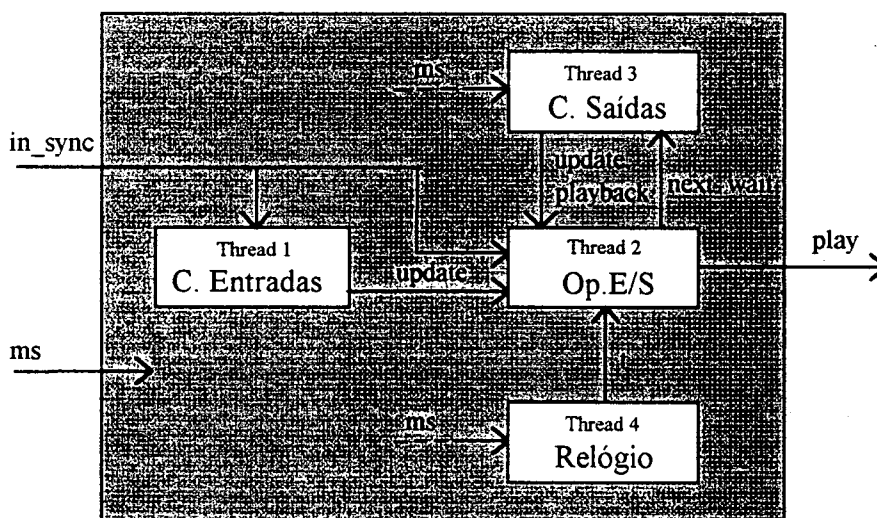


Figura 4.9 Estrutura do modelo do sincronizador do primeiro cenário de sincronização intra-fluxo.

Esta especificação é composta de quatro *threads* instanciados em paralelo para cuja comunicação são declarados os seguintes sinais internos:

- *update*, sinal que indica a atualização do *buffer*. Este sinal é emitido para cada saída ou entrada de IUs no *buffer* do sincronizador.
- *playback*, sinal que indica o instante da apresentação de uma IU e da sua conseqüente saída do *buffer*. Este sinal é emitido simultaneamente com o sinal *update*.

- *next\_wait*, sinal que quantifica o tempo de retenção do elemento na frente do *buffer*.

O comportamento dos *threads* é mostrado a seguir:

1) O primeiro *thread*, encarrega-se de coordenar a entrada de IUs no *buffer* emitindo o sinal *update* para cada chegada ao sincronizador, isto é, cada vez que o sinal externo de entrada *in\_sync* seja emitido.

2) O segundo *thread*, tem duas funções: *i)* fornecer diretamente a entrada e saída de IUs do *buffer* e *ii)* calcular seu tempo de retenção. Este *thread* é ativado cada vez que o sinal *update* é emitido.

- A entrada de uma IU no *buffer* é produzida na presença simultânea dos sinais *update* e *in\_sync*.
- A saída de uma IU do *buffer* ocorre na presença simultânea dos sinais *update* e *playback*. O sinal de saída *play* é emitido carregando como informação a IU em questão.
- O cálculo do tempo de retenção da IU é conseguido pela relação:

$$\text{tempo de retenção} = \text{rótulo de tempo} + TPT - \text{tempo atual}$$

Este cálculo é realizado cada vez que uma IU passa à frente do *buffer*, ou seja, cada vez que a variável *new\_wait* é verdadeira. O tempo de retenção calculado é então difundido pelo sinal interno *next\_wait*. Entretanto, para IUs chegando ao sincronizador no último instante, isto é, com tempo de retenção igual a zero, o sinal *playback* é emitido, permitindo sua atual apresentação nesse mesmo instante.

3) O terceiro *thread*, encarrega-se de coordenar a apresentação de IUs agindo como um temporizador. Assim, para cada emissão do sinal *next\_wait*, o *thread* aguarda um tempo equivalente ao valor deste sinal, quer dizer, ao tempo de retenção necessário para compensar o *jitter* de rede. Terminada esta espera o *thread* emite os sinais *update* e *playback* o quais produzem a apresentação (sinal *play*), e saída do *buffer*, da IU em questão.

4) O quarto *thread*, corresponde à instanciação de um relógio local, que fornece o valor do tempo atual utilizado para o cálculo dos tempos de retenção.

## B. Especificação Completa do Sincronizador

```

module Sincronizador:
constant
  TPT, SIZE_BUFF, SINK_CLOCK_SEED: integer,
  TRUE, FALSE: boolean;
type
  elt_type, queue_type;
input
  in_sync(elt_type), ms;
output
  play(elt_type);
function
  read_stamp(elt_type): integer,
  new_q(integer): queue_type,
  top_q(queue_type): elt_type,
  empty_q(queue_type): boolean;
procedure
  in_q (queue_type) (elt_type),
  out_q (queue_type) ();

signal update, playback, next_wait(integer), t(integer) in
  every in_sync do emit update end          % Thread coordenador de entradas
  ||
  var buff: queue_type, new_wait: boolean in % Thread de operações no buffer
    buff:= new_q(SIZE_BUFF+1);             % e do cálculo do tempo de retenção
  every update do
    new_wait:= FALSE;
    present in_sync then
      if (empty_q(buff)) then
        if (read_stamp(?in_sync) + TPT - ?t = 0) then emit playback
        else new_wait:= TRUE end
      end;
      call in_q (buff) (?in_sync)
    end;
    present playback then
      emit play(top_q(buff));
      call out_q (buff) ();
      if (not(empty_q(buff))) then new_wait:= TRUE end
    end;
    if (new_wait) then emit next_wait(read_stamp(top_q(buff)) + TPT - ?t) end
  end every
end var
  ||
  loop                                     % Thread coordenador de saídas
    await immediate next_wait;
    await (?next_wait) ms;
    emit update; emit playback
  end
  ||
  copymodule Relogio [signal t/curr_time; constant SINK_CLOCK_SEED/CLOCK_SEED]
end signal
end module

```

Na seqüência, de forma a completar a descrição do sincronizador, descrevem-se as declarações utilizadas.

### C. Declarações

As seguintes constantes são declaradas:

- *TPT*, tempo de apresentação alvo (*Target Presentation Time*).
- *SIZE\_BUFF*, comprimento do *buffer* do sincronizador.
- *SINK\_CLOCK\_SEED*, base de tempo do relógio do receptor.

Os tipos utilizados são:

- *elt\_type*, tipo de unidade de informação, ou unidade de dado, do fluxo a sincronizar.
- *queue\_type*, tipo de *buffer* que armazena dados tipo *elt\_type*.

Além das funções e procedimentos necessários para o manuseio da estrutura de dados fila, é declarada a função *read\_stamp()*, que permite consultar os rótulos de tempo de cada IU recebida.

### Modelagem do Sistema Global

Finalmente o módulo principal da especificação cria em paralelo instâncias dos módulos Emissor, Canal e Sincronizador. Para exemplificar este primeiro cenário, escolhem-se os dados seguintes: O emissor envia um fluxo de áudio com período igual a 30 ms. O canal garante atrasos mínimo e máximo de 10 e 15 ms respectivamente, ou seja introduz um *jitter* de 5 ms na transmissão de cada amostra de áudio. O Sincronizador é instanciado com um *TPT* igual ao máximo atraso do canal (15 ms), e com um tamanho de *buffer* capaz de conter no máximo 1 amostra de áudio. O mecanismo de sincronização compensa o *jitter* de rede e garante uma latência de apresentação igual a 15 ms.

```

module Fluxo:
type
  elt_type;
input
  ms;
output
  play(elt_type);
signal out_src(elt_type), in_sync(elt_type), t(integer) in
  copymodule Emissor [constant 30/TETA]
  ||

```



```

copymodule Canal [signal out_src/in_ch, in_sync/out_ch;
                  constant 10/D_MIN, 15/D_MAX]
||
copymodule Sincronizador [constant 15/TPT, 1/SIZE_BUFF]
end signal
end module

```

#### 4.2.4.2 O Modelo do Segundo Cenário

##### Estrutura da Especificação em Esterel

A estrutura modular da especificação em Esterel é mostrada na Figura 4.10. Neste modelo do sistema tem-se os seguintes módulos:

- Emissor, que gera um fluxo, rotulando cada unidade de informação, segundo o tempo fornecido por seu relógio local de base de tempo é  $b1$ , e cujo modelo já foi apresentado no primeiro cenário.
- Canal de transmissão, que introduz variações de atraso no intervalo  $[\Delta_{\min}, \Delta_{\max}]$ .
- Sincronizador, que reconstrói o fluxo através da compensação do *jitter* de rede, baseado na percepção de tempo de seu relógio local, cuja base de tempo é  $b2$ .

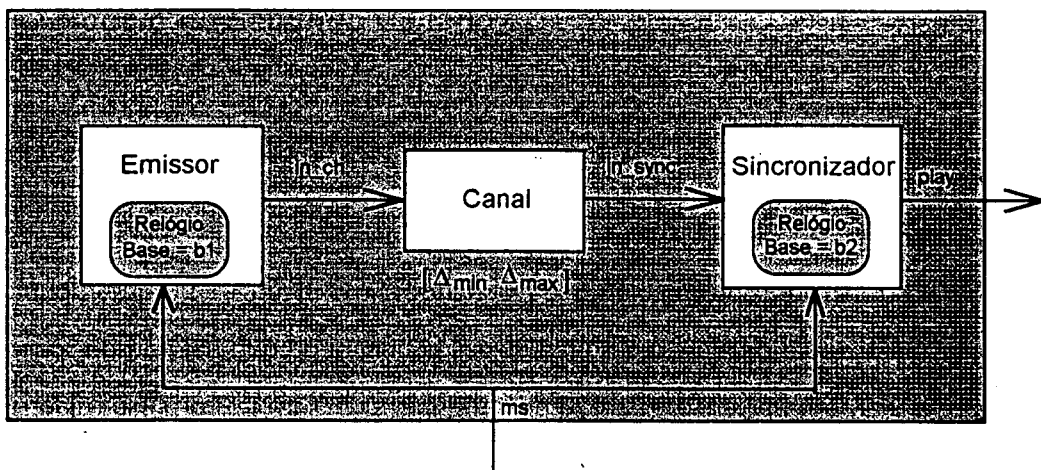


Figura 4.10 Estrutura modular Esterel do segundo cenário de sincronização intra-fluxo.

A comunicação entre os módulos é a seguinte:

Para cada item a transmitir, o emissor envia o sinal  $in\_ch$  ao canal, carregando a IU rotulada com o tempo corrente segundo seu relógio local. O sinal  $in\_sync$  é emitido pelo canal ao sincronizador uma vez finalizada a transmissão. Oportunamente, o sincronizador emite o sinal  $play$  carregando

como valor a IU a ser apresentada pela aplicação receptora. O sistema recebe como entrada o sinal periódico *ms* (*tick* de milissegundos) o qual é utilizado pelos relógios locais de emissor e receptor para fornecer o tempo atual.

### Modelagem do Sincronizador

A especificação deste sincronizador é similar àquela do sincronizador sob a hipótese de relógios comuns. A diferença se encontra basicamente no cálculo do tempo de retenção das IUs. Esta seção ressaltará este aspecto.

#### A. Estrutura do Modelo do Sincronizador

Esta especificação é composta de quatro *threads*, instanciados em paralelo, cuja comunicação é dada através dos sinais internos *update*, *playback*, *next\_wait* e *t*. Estes sinais, assim como a funcionalidade dos *threads*, já foram introduzidos na descrição do sincronizador do cenário anterior (veja na seção 4.2.4.1).

Entretanto, neste caso o comportamento do segundo *thread* difere basicamente no cálculo do tempo de retenção das IUs. Na presença simultânea dos sinais *update* e *in\_sync*, este *thread* realiza inicialmente o cálculo do tempo de apresentação da IU atual (*curr\_IUp*), segundo dois casos:

- No caso de tratar-se da primeira IU, o tempo de apresentação vem sendo dado por:

$$curr\_IUp = t + TPT - D\_MIN$$

- Para qualquer outra IU, o tempo de apresentação equivale a:

$$curr\_IUp = last\_IUp + curr\_IUs - last\_IUs$$

Onde *last\_IUp*, é o tempo de apresentação da IU imediatamente anterior à IU atual; *curr\_IUs*, é o rótulo de tempo da IU atual; e *last\_IUs*, corresponde ao rótulo de tempo da IU imediatamente anterior à IU atual.

Uma vez feito este cálculo, a IU é rotulada com seu tempo de apresentação. No caso de ser este tempo igual ao tempo atual, o sinal *playback* é emitido para a apresentação da IU nesse mesmo instante. Produz-se então a entrada da IU no *buffer* do sincronizador. Quando a IU passa à frente do *buffer*, é realizado o cálculo de seu tempo de retenção através da seguinte relação:

$$\text{tempo de retenção} = \text{tempo de apresentação} - \text{tempo atual}$$

Onde o tempo de apresentação da IU é obtido de seu rótulo de tempo, atualizado no instante da entrada da IU ao sincronizador.

## B. Especificação Completa do Sincronizador

```

module Sincronizador:
constant
  TPT, D_MIN, SIZE_BUFF, SINK_CLOCK_SEED: integer,
  TRUE, FALSE: boolean;
type
  elt_type, queue_type;
input
  in_sync(elt_type), ms;
output
  play(elt_type);
function
  read_stamp(elt_type): integer,
  new_q(integer): queue_type,
  top_q(queue_type): elt_type,
  empty_q(queue_type): boolean;
procedure
  write_stamp (elt_type) (integer),
  in_q (queue_type) (elt_type),
  out_q (queue_type) ();

signal update, playback, next_wait(integer), t(integer) in
  every in_sync do emit update end          % thread de coordenador de entradas
  ||
  var elt: elt_type, buff: queue_type,      % thread de operações no buffer
      new_wait, first:= TRUE: boolean,     % e cálculo do tempo de retenção
      curr_IUs, last_IUs, curr_IUp, last_IUp: integer in
      buff:= new_q(SIZE_BUFF+1);
  every update do
    new_wait:= FALSE;
    present in_sync then
      elt:= ?in_sync;
      curr_IUs:= read_stamp(elt);
      if (first) then
        curr_IUp:= ?t + (TPT - D_MIN);
        first:= FALSE
      else
        curr_IUp:= last_IUp + curr_IUs - last_IUs
      end;
      call write_stamp (elt) (curr_IUp);
      if (empty_q(buff)) then
        if (read_stamp(elt) = ?t) then emit playback
        ..else new_wait:= TRUE end
      end;
      call in_q (buff) (elt);
      last_IUp:= curr_IUp;
      last_IUs:= curr_IUs
    end;
  present playback then

```

```

        emit play(top_q(buff));
        call out_q (buff) ();
        if (not(empty_q(buff))) then new_wait:= TRUE end
    end;
    if (new_wait) then emit next_wait(read_stamp(top_q(buff)) - ?t) end
end every
end var
||
loop                                % thread coordenador de saídas
    await immediate next_wait;
    await (?next_wait) ms;
    emit update; emit playback
end
||
copymodule Relogio [signal t/curr_time; constant SINK_CLOCK_SEED/CLOCK_SEED]
end signal
end module

```

Neste modelo, além das declarações já introduzidas no cenário anterior é utilizada a constante  $D_{MIN}$ , a qual representa o atraso mínimo de transmissão do fluxo. Adicionalmente, o procedimento *write\_stamp()*, é utilizado para escrever os rótulos de tempo.

### Modelagem do Sistema Global

O módulo principal da especificação cria em paralelo instâncias dos módulos Emissor, Canal e Sincronizador. Para exemplificar este segundo cenário, escolhem-se os dados seguintes: O emissor envia um fluxo de vídeo com período igual a 40 ms. O canal garante atrasos mínimo e máximo de 10 e 15 ms respectivamente, ou seja introduz um *jitter* de 5 ms na transmissão de cada quadro de vídeo. O Sincronizador é instanciado com um *TPT* igual ao máximo atraso do canal (15 ms), e com um tamanho de *buffer* capaz de conter no máximo 1 quadro de vídeo. Finalmente, a base de tempo dos relógios de emissor e receptor diferem em 5 ms, isto é,  $SOURCE\_CLOCK\_SEED = 0$ , enquanto,  $SINK\_CLOCK\_SEED = 5$ . O mecanismo de sincronização compensa o *jitter* de rede e garante uma latência de apresentação compreendida no intervalo [15, 20] ms.

```

module Fluxo:
type
    elt_type;
input
    ms;
output
    play(elt_type);

signal out_src(elt_type), in_sync(elt_type) in
    copymodule Emissor [constant 40/TETA, 0/SOURCE_CLOCK_SEED]
    ||
    copymodule Canal [signal out_src/in_ch, in_sync/out_ch;

```

```

    constant 10/D_MIN, 15/D_MAX]
  ||
  copymodule Sincronizador [constant 15/TPT, 10/D_MIN, 1/SIZE_BUFF, 5/SINK_CLOCK_SEED]
end signal
end module

```

#### 4.2.4.3 O Modelo do Terceiro Cenário

##### Estrutura da Especificação em Esterel

A estrutura modular da especificação em Esterel é mostrada na Figura 4.11. Neste modelo do sistema tem-se os seguintes módulos:

- Emissor, que produz um fluxo de período nominal  $\theta_{nom}$ .
- Canal de transmissão, que introduz variações de atraso no intervalo  $[\Delta_{min}, \Delta_{max}]$ .
- Sincronizador, que compensa o *jitter* de rede e entrega o fluxo ao período de apresentação  $\theta_{nom}$ .

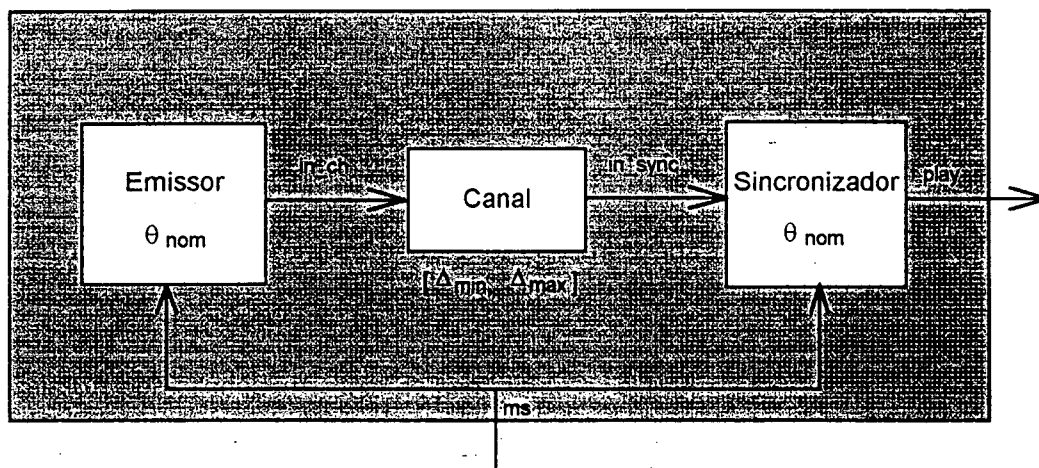


Figura 4.11 Estrutura modular Esterel do terceiro cenário de sincronização intra-fluxo.

##### Modelagem do Emissor

Este emissor limita-se ao envio de IUs segundo o período *TETA*. Neste caso não são necessários os rótulos de tempo. Observe-se que o fato de que emissor e sincronizador compartilhem o mesmo sinal *ms*, modela a hipótese de relógios a derivas comuns.

```

module Emissor:
constant
  TETA: integer;
type

```

```

elt_type;
input
ms;
output
out_src(elt_type);
function
get_elt(): elt_type;

await ms;
loop
emit out_src(get_elt());
await TETA ms
end
end module

```

## Modelagem do Sincronizador

### A. Estrutura do Modelo do Sincronizador

O modelo deste sincronizador é estruturalmente similar aos modelos dos sincronizadores dos cenários anteriores. Esta estrutura é mostrada na Figura 4.12.

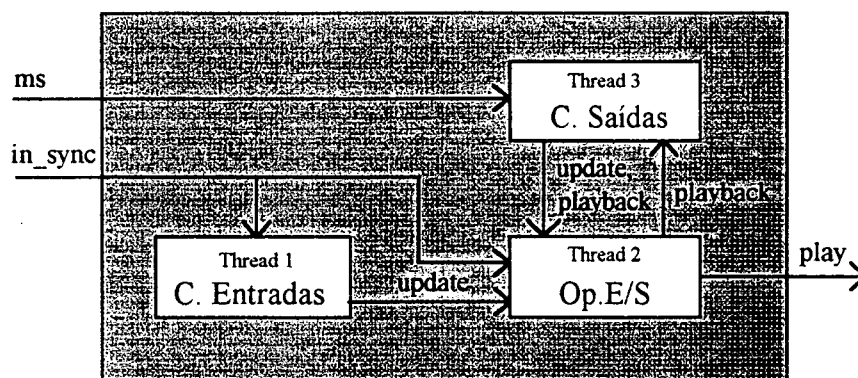


Figura 4.12 Estrutura do modelo do sincronizador do terceiro cenário de sincronização intra-fluxo.

O comportamento deste sincronizador é definido por três *threads* compostos em paralelo sendo que o primeiro *thread* já foi introduzido no primeiro cenário.

1. O segundo *thread* fornece a entrada e saída de IUs do *buffer*, e determina o início da apresentação:
  - A entrada de uma IU no *buffer* é produzida na presença simultânea dos sinais *update* e *in\_sync*. Para um número de entradas equivalente ao valor de  $PREFETCH + 1$ , o sinal *playback* é emitido, ocasionado o início da apresentação.

- A saída de uma IU do *buffer* ocorre na presença simultânea dos sinais *update* e *playback*. O sinal de saída *play* é emitido carregando como informação a IU em questão.
2. O terceiro *thread* age como temporizador conforme o período de apresentação *TETA*. Isto é, os sinais *update* e *playback* são emitidos a intervalos dados pelo valor de *TETA*. O sinal *playback* faz com que este *thread* seja iniciado.

## B. Especificação Completa do Sincronizador

```

module Sincronizador:
constant
  PREFETCH,
  TETA,
  SIZE_BUFF: integer,
  TRUE, FALSE: boolean;
type
  elt_type, queue_type;
input
  in_sync(elt_type), ms;
output
  play(elt_type);
function
  new_q(integer): queue_type,
  top_q(queue_type): elt_type,
  n_q(queue_type): integer;
procedure
  in_q (queue_type) (elt_type),
  out_q (queue_type) ();

signal update, playback in
  every in_sync do emit update end
  ||
  var buffer: queue_type, first:= TRUE: boolean in
    buffer:= new_q(SIZE_BUFF+1);
    every update do
      present in_sync then
        call in_q (buffer) (?in_sync);
        if (first and n_q(buffer) = PREFETCH + 1) then
          first:= FALSE; emit playback
        end
      end;
      present playback then
        emit play(top_q(buffer));
        call out_q (buffer) ()
      end
    end every
  end var
  ||
  [
  await playback;
  every TETA ms do
    emit update; emit playback
  end

```

```

]
end signal
end module

```

### Modelagem do Sistema Global

O módulo principal da especificação cria em paralelo instâncias dos módulos Emissor, Canal e Sincronizador. Neste caso, escolhe-se para exemplificar, um emissor que envia um fluxo de vídeo com período igual a 40 ms. O canal garante atrasos mínimo e máximo de 10 e 15 ms respectivamente (*jitter* de 5 ms). O Sincronizador é instanciado com um valor de *prefetch* igual a 1, o período de vídeo de 40 ms, e com um comprimento de *buffer* capaz de conter no máximo 2 quadros de vídeo.

```

module Fluxo:
type
  elt_type;
input
  ms;
output
  play(elt_type);
signal out_src(elt_type), in_sync(elt_type) in
  copymodule Emissor [constant 40/TETA]
  ||
  copymodule Canal [signal out_src/in_ch, in_sync/out_ch;
                    constant 10/D_MIN, 15/D_MAX]
  ||
  copymodule Sincronizador [constant 1/PREFETCH, 40/TETA, 2/SIZE_BUFF]
end signal
end module

```

#### 4.2.4.4 O Modelo do Quarto Cenário

##### Estrutura da Especificação em Esterel

A estrutura modular da especificação em Esterel é mostrada na Figura 4.13. Nesta especificação tem-se os seguintes módulos:

- Emissor, que produz um fluxo de período  $\theta_{in}$ .
- Canal de transmissão, que introduz variações de atraso no intervalo  $[\Delta_{min}, \Delta_{max}]$ .
- Sincronizador, que compensa o *jitter* de rede, e a diferença de períodos, entregando o fluxo ao período de apresentação  $\theta_{out}$ .



```

]
end signal
end module

```

### Modelagem do Sistema Global

O módulo principal da especificação cria em paralelo instâncias dos módulos Emissor, Canal e Sincronizador. Neste caso, escolhe-se para exemplificar, um emissor que envia um fluxo de vídeo com período igual a 40 ms. O canal garante atrasos mínimo e máximo de 10 e 15 ms respectivamente (*jitter* de 5 ms). O Sincronizador é instanciado com um valor de *prefetch* igual a 1, o período de vídeo de 40 ms, e com um comprimento de *buffer* capaz de conter no máximo 2 quadros de vídeo.

```

module Fluxo:
type
  elt_type;
input
  ms;
output
  play(elt_type);
signal out_src(elt_type), in_sync(elt_type) in
  copymodule Emissor [constant 40/TETA]
  ||
  copymodule Canal [signal out_src/in_ch, in_sync/out_ch;
                    constant 10/D_MIN, 15/D_MAX]
  ||
  copymodule Sincronizador [constant 1/PREFETCH, 40/TETA, 2/SIZE_BUFF]
end signal
end module

```

#### 4.2.4.4 O Modelo do Quarto Cenário

##### Estrutura da Especificação em Esterel

A estrutura modular da especificação em Esterel é mostrada na Figura 4.13. Nesta especificação tem-se os seguintes módulos:

- Emissor, que produz um fluxo de período  $\theta_{in}$ ,
- Canal de transmissão, que introduz variações de atraso no intervalo  $[\Delta_{min}, \Delta_{max}]$ ,
- Sincronizador, que compensa o *jitter* de rede, e a diferença de períodos, entregando o fluxo ao período de apresentação  $\theta_{out}$ .

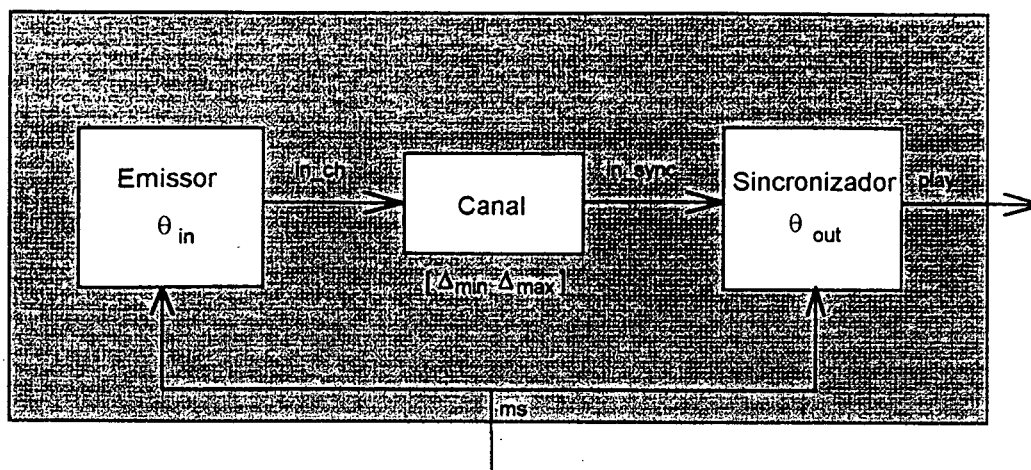


Figura 4.13 Estrutura modular Esterel do quarto cenário de sincronização intra-fluxo.

### Modelagem do Emissor

Trata-se do emissor periódico definido na solução anterior com período de produção igual a  $\theta_{in}$ . Note-se que a hipótese de relógios a derivas constantes é modelada com o emissor e o sincronizador compartilhando o sinal *ms* do relógio externo, porém com períodos de produção e apresentação diferentes.

### Modelagem do Sincronizador

#### A. Estrutura do Modelo do Sincronizador

O comportamento deste sincronizador representa uma extensão do sincronizador baseado na técnica do *prefetch*, apresentado na solução anterior. O modelo apresenta a mesma estrutura deste último, sendo definido por três *threads* compostos em paralelo, dos quais só o primeiro não difere. O comportamento dos outros *threads* é o seguinte:

1. O segundo *thread*, além de operar a entrada e saída de IUs do *buffer* e determinar o início da apresentação, fornece as operações *eliminar* ou *duplicar* segundo seja o caso:
  - A operação *eliminar* ocorre na presença dos sinais *update* e *delete*, e produz a saída da IU na frente do *buffer* do sincronizador. Em consequência, esta IU é omitida, sendo que a IU seguinte é apresentada no seguinte instante de apresentação.
  - A operação *duplicar* só ocorre na presença dos sinais *update* e *duplicate*, e produz uma apresentação adicional da última IU apresentada. O valor da última IU apresentada é

obtido do sinal *play*. A função *in\_f\_q()* fornece a entrada da IU diretamente na frente do *buffer*. O instante de apresentação da primeira IU corresponde-se com a primeira ativação da operação *duplicate*, no caso do período de emulação, *TETA\_EMU*, ser positivo.

2. O terceiro *thread*, por sua vez, é conformado por dois sub-*threads* que agem como temporizadores das apresentações e das operações *eliminar* ou *duplicar*, conforme os períodos *TETA\_OUT* e *TETA\_EMU* respectivamente. Isto é, os sinais *update* e *playback* são emitidos a intervalos dados pelo valor de *TETA\_OUT*. Similarmente, o sinal *update* é emitido simultaneamente com o sinal *duplicate*, no caso de ser *TETA\_EMU* > 0, ou simultaneamente com o sinal *delete* no caso contrário. O sinal *playback* faz com que este *thread* seja iniciado.

## B. Especificação Completa do Sincronizador

```

module Sincronizador:
constant
  PREFETCH,
  TETA_OUT,
  TETA_EMU,
  SIZE_BUFF: integer,
  TRUE, FALSE: boolean;
type
  elt_type, queue_type;
input
  in_sync(elt_type), ms;
output
  play(elt_type);
function
  new_q(integer): queue_type,
  top_q(queue_type): elt_type,
  n_q(queue_type): integer,
  absolute(integer): integer;
procedure
  in_q (queue_type) (elt_type),
  in_f_q (queue_type) (elt_type),
  out_q (queue_type) ();

signal update, playback, duplicate, delete in
every in_sync do emit update end
||
var buffer: queue_type, first:= TRUE: boolean in
  buffer:= new_q(SIZE_BUFF+1);
  every update do
    present in_sync then
      call in_q (buffer) (?in_sync);
      if (first and n_q(buffer) = PREFETCH + 1) then
        first:= FALSE; emit playback;
        if (TETA_EMU > 0) then emit duplicate end
      end
    end;
  present delete then

```

```

        call out_q (buffer) ()
    end;
    present playback then
        emit play(top_q(buffer));
        call out_q (buffer) ()
    end;
    present duplicate then
        call in_f_q (buffer) (?play)
    end;
end every
end var
||
[
await playback;
[
every TETA_OUT ms do
    emit update; emit playback
end
||
every absolute(TETA_EMU) ms do
    emit update;
    if (TETA_EMU > 0) then emit duplicate
    else emit delete
end
]
]
end signal
end module

```

### Modelagem do Sistema Global

O módulo principal é composto das instâncias em paralelo dos módulos Emissor, Canal e Sincronizador. Neste caso, escolhe-se um emissor que envia um fluxo de vídeo com período igual a 40 ms. O canal introduz um *jitter* de 5 ms na transmissão de cada quadro de vídeo (atrasos mínimo e máximo de 10 e 15 ms respectivamente). O Sincronizador é instanciado com um valor de *prefetch* igual a 1, um período de apresentação de 30 ms, um período de emulação 120 ms e com um comprimento de *buffer* capaz de conter no máximo 3 quadros de vídeo. Assim, uma vez iniciada a apresentação, o emulador compensará a diferença de períodos através de operações *duplicate* cada 120 ms.

```

module Fluxo:
type
    elt_type;
input
    ms;
output
    play(elt_type);
signal out_src(elt_type), in_sync(elt_type) in
    copymodule Emissor[constant 40/TETA_IN]
||

```

```

copymodule Canal [signal out_src/in_ch, in_sync/out_ch;
                  constant 10/D_MIN, 15/D_MAX]
||
copymodule Sincronizador [constant 1/PREFETCH,30/TETA_OUT, 20/TETA_EMU,3/SIZE_BUFF]
end signal
end module

```

#### 4.2.4.5 O Modelo do Quinto Cenário

##### Estrutura da Especificação em Esterel

A estrutura modular da especificação em Esterel é mostrada na Figura 4.14. Neste modelo do sistema tem-se os seguintes módulos:

- Servidor Multimídia, que garante a continuidade de apresentação escalonando o envio de IUs do fluxo periódico, baseado na percepção do tempo de seu relógio local, e nas mensagens de realimentação enviadas pelo midiafone.
- Canal de transmissão do fluxo, que introduz variações de atraso fim-a-fim no intervalo  $[\Delta^m_{\min}, \Delta^m_{\max}]$ .
- Canal de transmissão das mensagens de realimentação, que introduz variações de atraso fim-a-fim no intervalo  $[\Delta^f_{\min}, \Delta^f_{\max}]$ .
- Midiafone, com variação do período de apresentação  $[\theta_{\min}, \theta_{\max}]$ .

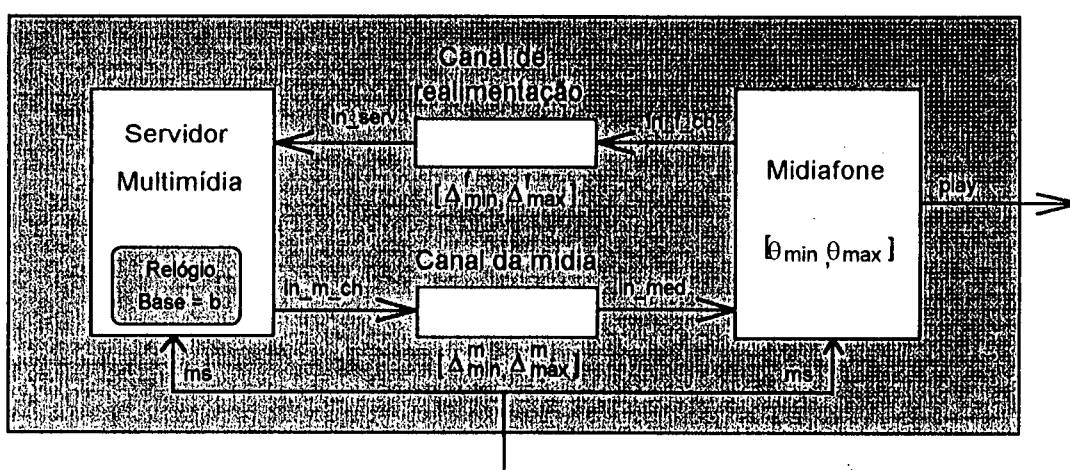


Figura 4.14 Estrutura modular Esterel do quinto cenário de sincronização intra-fluxo.

## Modelagem do Servidor

### A. Estrutura do Modelo do Servidor

A estrutura do modelo do servidor multimídia é mostrada na Figura 4.15. A interface deste módulo é dada pelos sinais:

- *in\_serv*, sinal de entrada das mensagens de realimentação.
- *in\_m\_ch*, sinal de saída das IUs.
- *ms*, sinal de entrada de *tick* cada milissegundo.

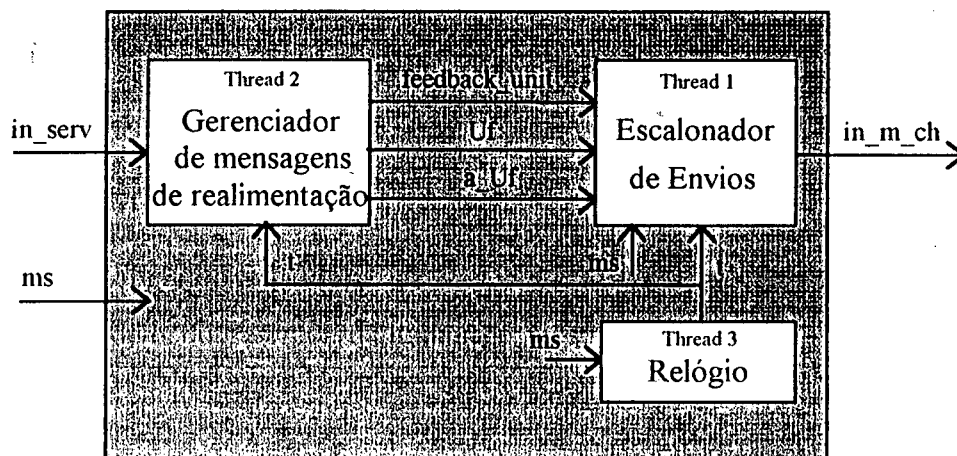


Figura 4.15 Estrutura do modelo do servidor multimídia do quinto cenário de sincronização intra-fluxo.

Para a comunicação entre *threads* são utilizados os seguintes sinais internos:

- *Uf*, que carrega o número de seqüência da última mensagem de realimentação recebida pelo servidor.
- *a\_Uf*, que carrega o tempo de chegada da última mensagem de realimentação ao servidor.
- *feedback\_unit*, que é utilizado como *flag* indicador da chegada da primeira mensagem de realimentação.
- *t*, que é o tempo atual do relógio do servidor.

O comportamento de cada thread é o seguinte:

1) O primeiro *thread* encarrega-se basicamente de obter, marcar e enviar as IUs, após um certo tempo de espera. Inicialmente, este *thread* obtém uma IU através da função *get\_elt()* e verifica se deve ser marcada, através da função *tobemarked()*. Se for o caso, a IU é marcada utilizando a função *write\_mark()*. A IU é então enviada. O tempo de espera da seguinte IU (valor do sinal *teta*) é estimado segundo o caso de ter ou não recebido a primeira mensagem de realimentação, isto é, segundo o valor do sinal *feedback\_unit*. O *thread* aguarda este tempo de espera para então repetir sucessivamente as ações obter, marcar, enviar e aguardar.

- A função *next\_time()*, utilizada antes da recepção da primeira mensagem de realimentação, retorna o valor de tempo equivalente ao último instante no qual a IU de número de seqüência *n\_seq* deve ser transmitida conforme a:

$$next\_time = (n\_seq - 1) * TETA\_MIN - (D\_MAX - D\_MIN)$$

Visando evitar fome no receptor, esta função assume o pior caso onde os atrasos atuais de transmissão de IUs são máximos, e o período de apresentação no midiafone é mínimo. Entretanto, valores atuais de atrasos e períodos diferentes aos assumidos, conduzem a uma iminente saturação do *buffer* do midiafone. Por exemplo, valores de atrasos mínimos e período de apresentação máximo, provocam a mais rápida acumulação. Assim, esta função é só utilizada até a chegada da primeira mensagem de realimentação.

- As funções *earliest\_next\_time()*, *latest\_next\_time()* e *scheduler()* são utilizadas após a recepção da primeira mensagem de realimentação.

A função *earliest\_next\_time()*, retorna o valor de tempo *earliest*, equivalente ao instante mais próximo no qual a IU de número de seqüência *n\_seq* deve ser transmitida conforme a:

$$a\_Uf - D\_MIN\_F + (n\_seq - SIZE\_BUFF - Uf) * TETA\_MAX - D\_MIN\_M$$

A função *latest\_next\_time()*, retorna o valor de tempo *latest*, equivalente ao último instante no qual a IU de número de seqüência *n\_seq* deve ser transmitida conforme a:

$$a\_Uf - D\_MAX\_F + (n\_seq - Uf) * TETA\_MIN - D\_MAX\_M$$

Finalmente, a função *scheduler()*, retorna um valor de tempo de envio dentro do intervalo [*earliest*, *latest*], segundo a política de escalonamento definida. Por exemplo, com envios

num tempo igual a *earliest*, persegue-se manter a ocupação do *buffer* perto de seu comprimento. Pelo contrario, com envios num tempo igual a *latest*, procura-se manter a ocupação do *buffer* ao mínimo.

- O tempo de espera da IU é determinado subtraindo o tempo atual, do valor de tempo de envio calculado. Este valor é difundido pelo sinal *teta* para a espera do tempo correspondente.

- 2) O segundo *thread* aguarda inicialmente pela chegada da primeira mensagem de realimentação. Após esta chegada, atualiza seu *flag* indicador, ou sinal *feedback\_unit*. No mesmo instante os sinais *a\_Uf* e *Uf* são emitidos contendo o tempo de chegada e o número de seqüência da mensagem. Posteriores esperas por mensagens de realimentação são seguidas da emissão dos sinais *a\_Uf* e *Uf* com seu respectivos tempos de chegada e números de seqüência.
- 3) O terceiro *thread* corresponde à criação de uma instância do relógio local.

## B. Especificação Completa do Servidor Multimídia

```

module Servidor:
constant
  D_MIN_M, D_MAX_M,
  D_MIN_F, D_MAX_F,
  TETA_MIN, TETA_MAX,
  N_FEEDBACK, SIZE_BUFF: integer,
  TRUE, FALSE: boolean;
type
  elt_type;
input
  out_f_ch(integer), ms;
output
  in_m_ch(elt_type);
function
  get_elt(): elt_type,
  n_elt(elt_type): integer,
  tobemarked(elt_type, integer): boolean,
  next_time(integer, integer, integer, integer): integer,
  earliest_next_time(integer, integer, integer, integer, integer, integer, integer): integer,
  latest_next_time(integer, integer, integer, integer, integer, integer): integer,
  scheduler(integer, integer): integer;
procedure
  write_mark (elt_type) ();

signal
  Uf(integer), a_Uf(integer), tetat(integer), feedback_unit(boolean), t(integer) in
emit feedback_unit(FALSE);
await ms;
[
var elt: elt_type, earliest, latest: integer in

```



```

loop
  elt:= get_elt();
  if (tobemarked(elt,N_FEEDBACK)) then
    call write_mark (elt) ()
  end;
  emit in_m_ch (elt);
  if (not(?feedback_unit)) then
    emit teta(next_time(n_elt(elt)+1,TETA_MIN,D_MAX_M,D_MIN_M) -?t)
  else
    earliest:=earliest_next_time(?a_Uf,D_MIN_F,n_elt(elt)+1,SIZE_BUFF,?Uf,TETA_MAX,D_MIN_M)-?t;
    latest:= latest_next_time(?a_Uf,D_MAX_F,n_elt(elt)+1,?Uf,TETA_MIN,D_MAX_M) - ?t;
    emit teta(scheduler(earliest, latest));
  end;
  await ?teta ms
end loop
end var
||
await out_f_ch;
emit feedback_unit(TRUE);
loop
  emit a_Uf(?t);
  emit Uf(?out_f_ch);
  await out_f_ch
end
}
||
copymodule Relogio [signal t/curr_time]
end signal
end module

```

### C. Declarações

Os seguintes parâmetros são declarados na forma de constantes:

- $D\_MIN\_M$  e  $D\_MAX\_M$ , atrasos mínimo e máximo do canal de transmissão do fluxo.
- $D\_MIN\_F$  e  $D\_MAX\_F$ , atrasos mínimo e máximo do canal de transmissão de mensagens de realimentação.
- $TETA\_MIN$  e  $TETA\_MAX$ , valores mínimo e máximo de variação do período de apresentação.
- $N\_FEEDBACK$ , período de rotulagem de marcas nas IUs.
- $SIZE\_BUFF$ , comprimento do *buffer* do midiafone.

As seguintes funções e procedimentos são ainda declarados como suporte para a especificação:

- $get\_elt()$ , permite obter as IUs com seus respectivos números de seqüência.

- *n\_elt()*, consulta o número de seqüência da IU.
- *write\_mark()*, procedimento que marca uma IU, para seu posterior envio como mensagem de realimentação.
- *tobemarked()*, conforme o valor de *N\_FEEDBACK*, esta função determina se a IU a enviar dever ser marcada.
- *next\_time()*, estima o tempo mais distante de envio da seguinte IU antes da chegada da primeira mensagem de realimentação.
- *earliest\_next\_time()*, estima o tempo mais próximo de envio da seguinte unidade de informação baseando-se na última mensagem de realimentação recebida. Esta estimativa visa evitar a saturação do *buffer* receptor.
- *latest\_next\_time()*, estima o tempo mais distante de envio da seguinte unidade de informação baseando-se na última mensagem de realimentação recebida. Esta estimativa visa evitar a fome do *buffer* receptor.
- *scheduler()*, esta função abstrai uma política de envio baseada na estimativa do intervalo válido de envio fornecido pelas duas funções anteriores.

## Modelagem do Midiafone

### A. Estrutura do Modelo do Midiafone

A estrutura do modelo do midiafone é mostrada na Figura 4.16. A interface deste modelo é dada pelos sinais: *out\_m\_ch*, sinal de entrada que indica a chegada das IUs a apresentar; *in\_f\_ch*, sinal de saída, para o envio de mensagens de realimentação ao servidor; *play*, sinal de saída, para a apresentação de IUs; e *ms*, sinal de entrada de *tick* cada milissegundo.

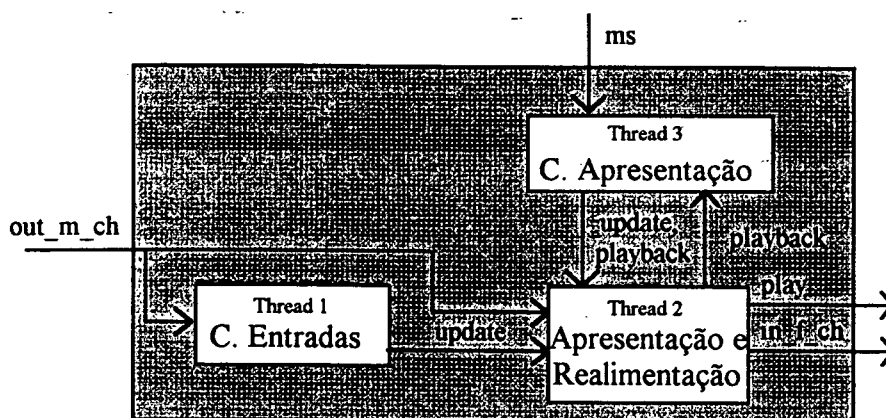


Figura 4.16 Estrutura do modelo do midiafone do quinto cenário de sincronização intra-fluxo.

Esta especificação é composta de três *threads* instanciados em paralelo para cuja comunicação são utilizados os seguintes sinais internos:

- *update*, sinal que indica a atualização do *buffer*. Este sinal é emitido para cada saída o entrada de IUs no *buffer* do midiafone.
- *playback*, sinal de indicação do instante de sinalização da apresentação de uma IU (*play*) é de sua conseqüente saída do *buffer*. Este sinal é emitido simultaneamente com o sinal *update*.

O comportamento de cada *thread* é o seguinte:

- 1) O primeiro *thread*, encarrega-se de coordenar a entrada de IUs no *buffer* emitindo o sinal *update* para cada chegada ao midiafone, isto é, cada vez que o sinal externo de entrada *out\_m\_ch* seja emitido.
- 2) O segundo *thread*, tem duas funções: fornecer diretamente a entrada e saída de IUs do *buffer* e enviar mensagens de realimentação ao servidor. Este *thread* é ativado cada vez que o sinal *update* é emitido.
  - A entrada de uma IU no *buffer* é produzida na presença simultânea dos sinais *update* e *out\_m\_ch*. Para um número de entradas equivalente ao valor do *PREFETCH*, o sinal *playback* é emitido, provocando o início da apresentação.
  - A saída de uma IU do *buffer* ocorre na presença simultânea dos sinais *update* e *playback*. O sinal de saída *play* é emitido carregando como informação a IU em questão.

Adicionalmente, no instante da saída de cada IU é verificado se está marcada. Em caso afirmativo, o sinal *in\_f\_ch* é emitido, como mensagem de realimentação, carregando o número de seqüência da IU apresentada.

- 3) O terceiro *thread*, encarrega-se de coordenar a apresentação de IUs introduzindo certa variação não determinística. Uma vez iniciada a apresentação, isto é, uma vez recebido o sinal *playback*, este *thread* inicia um ciclo de esperas seguidas da emissão dos sinais *update* e *playback*. O tempo de espera é fornecido pela função *teta\_midiafone()*, que introduz a variação aleatória baseada no valor dos parâmetros *TETA\_MIN* e *TETA\_MAX*.

### B. Especificação Completa do Midiafone

```

module Midiafone:
constant
  TETA_MIN, TETA_MAX,
  PREFETCH, SIZE_BUFF: integer,
  TRUE, FALSE: boolean;
type
  elt_type, queue_type;
input
  out_m_ch(elt_type), ms;
output
  play(integer), in_f_ch(integer);
function
  n_elt(elt_type): integer,
  info_elt(elt_type): integer,
  marked_elt(elt_type): boolean,
  new_q(integer): queue_type,
  top_q(queue_type): elt_type,
  n_q(queue_type): integer,
  teta_midiafone(integer, integer): integer;
procedure
  in_q (queue_type) (elt_type),
  out_q (queue_type) ();

signal update, playback in
  every out_m_ch do emit update end
  ||
  var buffer: queue_type, first:= TRUE: boolean in
    buffer:= new_q(SIZE_BUFF+1);
    every update do
      present out_m_ch then
        call in_q (buffer) (?out_m_ch);
        if (first and n_q(buffer) = PREFETCH) then
          emit playback; first:= FALSE
        end
      end;
    present playback then
      emit play(info_elt(top_q(buffer)))
      if (marked_elt(top_q(buffer))) then

```

```

        emit in_f_ch(n_elt(top_q(buffer)))
    end;
    call out_q (buffer) ();
end;
end every
end var
||
[
await playback;
loop
    await teta_midiafone(TETA_MIN, TETA_MIN) ms;
    emit update; emit playback
end
]
end signal
end module

```

### C. Declarações

Os seguintes parâmetros são declarados na forma de constantes: *TETA\_MIN* e *TETA\_MAX*, valores mínimo e máximo do período de apresentação; *PREFETCH*, número de IUs a receber antes do começo da apresentação; e *SIZE\_BUFF*, comprimento do *buffer* do midiafone.

Além das funções que operam sobre o tipo *queue\_type*, são declaradas as seguintes funções:

- *n\_elt()*, consulta o número de seqüência da IU.
- *info\_elt()*, obtém apenas a IU em si, isto é, sem seu número de seqüência.
- *marked\_elt()*, retorna verdadeiro no caso da IU estar marcada.
- *teta\_midiafone()*, função que introduz a variação do período de apresentação do midiafone, entre os valores *TETA\_MIN* e *TETA\_MAX*.

### Modelagem do Sistema Global

O módulo principal é composto de instâncias em paralelo dos módulos Servidor, Canal e Midiafone. Neste caso, escolhe-se um exemplo de aplicação onde as duas instâncias do módulo Canal, representam o canal de transmissão do fluxo, com atrasos no intervalo [20,30] ms, e o canal de transmissão das mensagens de realimentação, com garantia de transmissão no intervalo [6,15] ms. O midiafone apresenta um fluxo de vídeo com período nominal igual a 40 ms e variação de mais ou menos 1 ms. O *buffer* do midiafone tem um comprimento de 5 unidades, e deve respeitar o *prefetch* de 1 unidade de informação do fluxo. Conforme os parâmetros

anteriores, estimou-se a taxa mínima de mensagens de realimentação necessárias para manter a continuidade,  $N\_FEEDBACK$ . Neste caso este valor é 87. O Servidor é instanciado com esta taxa de realimentação e com os parâmetros anteriores do sistema.

```

module Feedback:
constant
  D_MIN_M, D_MAX_M,      % [20,30] ms
  D_MIN_F, D_MAX_F,      % [6,15] ms
  TETA_MIN, TETA_MAX,    % [39,41] ms
  PREFETCH,              % 1 unidade
  N_FEEDBACK,            % 87 unidades
  SIZE_BUFF: integer;    % 5 unidades
type
  elt_type;
input
  ms;
output
  play(integer);
signal
  in_m_ch(elt_type), in_med(elt_type), in_f_ch(integer), in_serv(integer) in

  copymodule Servidor [signal in_serv/out_f_ch]
  ||
  copymodule Canal [signal in_m_ch/in_ch, in_med/out_ch;
                    constant D_M_MIN/D_MIN, D_M_MAX/D_MAX]
  ||
  copymodule Canal [signal in_f_ch/in_ch, in_serv/out_ch;
                    constant D_F_MIN/D_MIN, D_F_MAX/D_MAX]
  ||
  copymodule Midiafone [signal in_med/out_m_ch]
end signal
end module

```

### 4.3 Modelagem de Técnicas de Sincronização Inter-Fluxo

Nesta seção é apresentado um problema clássico de sincronização inter-fluxo: a sincronização de lábios. O modelo da solução apresentada constitui uma extensão do modelo utilizado no primeiro cenário de sincronização intra-fluxo.

#### 4.3.1 O Cenário de Sincronização Inter-Fluxo

##### *Especificação e Requisitos do Sistema*

Requer-se a apresentação sincronizada de fluxos de áudio e vídeo (sincronização de lábios) gerados desde dois emissores remotos. Os fluxos deverão ser apresentadas em dispositivos de apresentação num mesmo nó receptor. Os canais de transmissão são totalmente confiáveis, com

atrasos de transmissão limitados. Deve-se respeitar certa latência máxima de apresentação. Trata-se de um sistema de relógios comuns. A Figura 4.17 mostra a estrutura deste sistema.

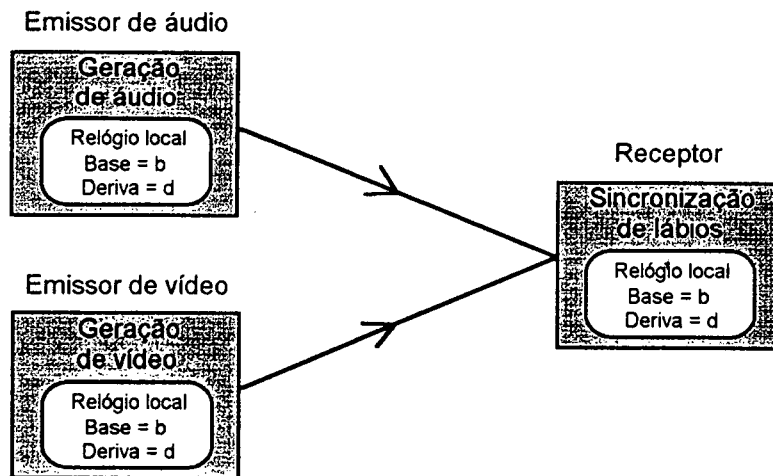


Figura 4.17 Sincronização de lábios num sistema de relógios comuns.

### **Solução do Problema**

A estratégia de sincronização utilizada nesta solução é uma extensão do esquema de sincronização intra-fluxo baseado em *time-stamping*, para sistemas de relógios comuns. Sob esta hipótese, é possível definir um tempo de apresentação comum para os fluxos, assim como uma única referência de início de geração destes. Desta forma, consegue-se estender o mecanismo para a sincronização inter-fluxo. Esta idéia é a base dos mecanismos propostos em [Escobar91] e [Dairaine94] já introduzidos na seção 2.5.3.1.

### **Atendimento dos Requisitos:**

- Neste esquema os fluxos a sincronizar não precisam ser periódicos.
- O esquema garante a não ocorrência de *jitter* de comunicação, e a entrega sincronizada dos fluxos envolvidos.
- A latência de apresentação é fixada no valor do máximo atraso fim-a-fim dos fluxos envolvidos.

### 4.3.2 Modelagem do Cenário de Sincronização Inter-Fluxo

#### *Estrutura da Especificação em Esterel*

A estrutura modular da especificação em Esterel é mostrada na Figura 4.18. Neste modelo do sistema tem-se os seguintes módulos:

- Emissor de áudio, que gera o fluxo de áudio, rotulando cada amostra.
- Emissor de vídeo, que gera o fluxo de vídeo, rotulando cada quadro. Os emissores começam a geração de fluxos simultaneamente.
- Canal de áudio, que introduz variações de atraso no intervalo  $[\Delta^S_{\min}, \Delta^S_{\max}]$ .
- Canal de vídeo, que introduz variações de atraso no intervalo  $[\Delta^V_{\min}, \Delta^V_{\max}]$ .
- Sincronizador de áudio, que compensa o *jitter* do canal de áudio, segundo um tempo de apresentação alvo, *TPT*, comum a ambos sincronizadores.
- Sincronizador de vídeo, que compensa o *jitter* do canal de vídeo.
- Relógio global, que fornece uma visão de tempo comum a todos os elementos do sistema, segundo a hipótese de relógios comuns.

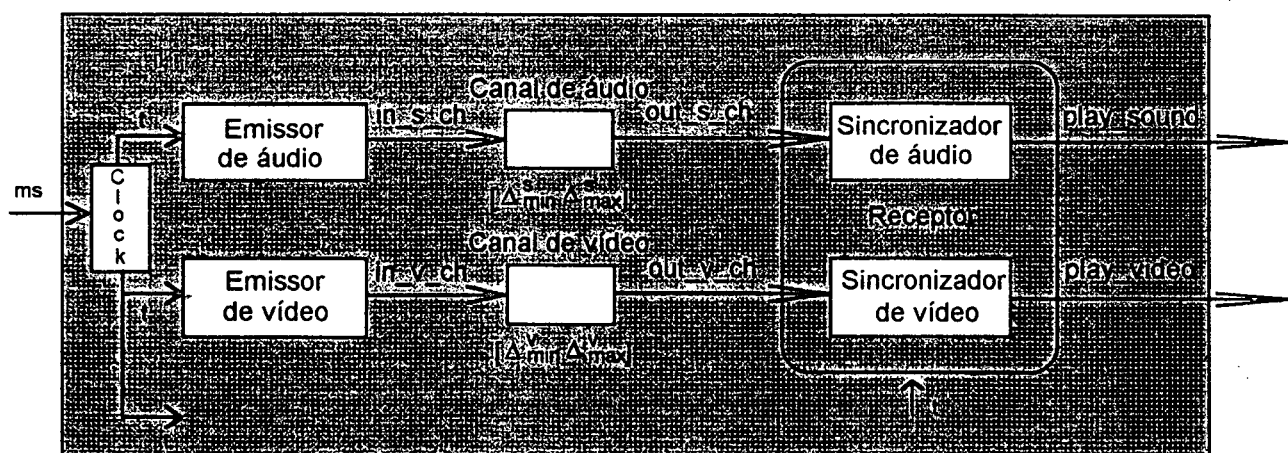


Figura 4.18 Estrutura modular Esterel do cenário de sincronização inter-fluxo.





```

copymodule Sincronizador [signal out_v_ch/in_sync, play_video/play;
                        constant 2/SIZE_BUFF, 30/TPT]
||
copymodule Relogio
end signal
end module

```

## 4.4 Modelagem de Técnicas de Sincronização Multidestino

Nesta seção é apresentado um cenário de sincronização multidestino sob a hipótese de relógios não sincronizados. O modelo da solução apresentada constitui uma extensão do quinto cenário de sincronização intra-fluxo.

### 4.4.1 O Cenário de Sincronização Multidestino

#### Especificação e Requisitos do Sistema

Requer-se a sincronização dos fluxos: áudio e vídeo, gerados desde um único nó emissor, até dois dispositivos de apresentação remotos localizados possivelmente em diferentes receptores. O período de apresentação dos dispositivos varia dentro de limites conhecidos. Os canais de transmissão são considerados como totalmente confiáveis, e garantem limites de atrasos fim-a-fim. Não é desejada qualquer descontinuidade no fluxo de áudio. O fluxo de vídeo pode sofrer eventuais descontinuidades para manter a sincronização respeito ao fluxo de áudio entretanto, só é tolerado um certo limite de assincronia entre os fluxos. Os relógios do sistema não estão sincronizados. A estrutura deste sistema é apresentada na Figura 4.19.

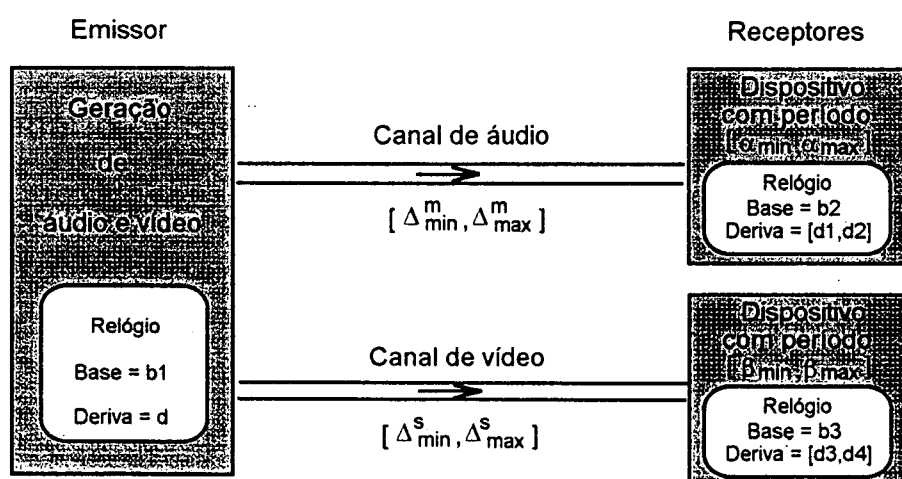


Figura 4.19 Sincronização de lábios num sistema de relógios não sincronizados.

## 4.4.2 Modelagem do Cenário de Sincronização Multidestino

### *Estrutura da Especificação em Esterel*

A estrutura modular da especificação em Esterel é mostrada na Figura 4.20. Neste modelo do sistema tem-se os seguintes módulos:

- Servidor Multimídia, que garante a continuidade e a sincronização dos fluxos áudio e vídeo, escalonando os envios das IUs de cada fluxo e oportunamente duplicando ou pulando IUs do fluxo *slave* (fluxo de vídeo). O servidor utiliza as mensagens de realimentação enviadas pelos midiafones.
- Canal de transmissão de áudio (fluxo *master*), que introduz variações de atraso fim-a-fim no intervalo  $[\Delta^m_{\min}, \Delta^m_{\max}]$ .
- Canal de transmissão das mensagens de realimentação do midiafone de áudio, que introduz variações de atraso fim-a-fim no intervalo  $[\Delta^f_{\min}, \Delta^f_{\max}]$ .
- Canal de transmissão do vídeo (fluxo *slave*), que introduz variações de atraso fim-a-fim no intervalo  $[\Delta^s_{\min}, \Delta^s_{\max}]$ .
- Canal de transmissão das mensagens de realimentação do midiafone de vídeo, que introduz variações de atraso fim-a-fim no intervalo  $[\Delta^f_{\min}, \Delta^f_{\max}]$ .
- Midiafone de áudio, com variação do período de apresentação  $[\alpha_{\min}, \alpha_{\max}]$ .
- Midiafone de vídeo, com variação do período de apresentação  $[\beta_{\min}, \beta_{\max}]$ .

### **Solução do Problema**

O mecanismo chamado *Multiple Feedback Techniques* apresentado em [Ramanathan93] (introduzido na seção 2.6.2) garante a continuidade e sincronização de fluxos periódicos em dispositivos de apresentação remotos aqui chamados de midiafones. Esta técnica é empregada por um Servidor Multimídia e corresponde a uma extensão do mecanismo de sincronização intra-fluxo apresentado na seção 2.4.1.2.

A continuidade de cada fluxo é garantida como já foi descrito. Para o fornecimento da sincronização inter-fluxo é definido um fluxo *master*, geralmente aquele com requisitos de sincronização intra-fluxo mais estritos. Adicionalmente as IUs possuem rótulos de tempo relativos (*RTS - Relative Time Stamps*) que representam seu tempo de apresentação em relação ao início da apresentação do fluxo. Estes rótulos de tempo são fornecidos pelo servidor no momento de armazenamento dos fluxos, e facilitam a ressincronização na apresentação. As mensagens de realimentação permitem estimar a dessincronização entre os fluxos. A ressincronização é então conseguida pelas operações eliminar e duplicar aplicadas oportunamente pelo servidor nas IUs do fluxo *slave*.

#### **Atendimento dos Requisitos:**

- Os fluxos a sincronizar devem ser periódicos.
- Garante-se a continuidade de ambos fluxos porém, o fluxo *slave*, está sujeito a operações *delete/duplicate* para compensar possíveis perdas de sincronização inter-fluxo.
- A máxima assincronia inter-fluxo depende da exatidão de sua estimação, a qual por sua vez, depende do período de mensagens de realimentação. O mecanismo permite calcular o período de mensagens de realimentação necessário para respeitar o valor desejado de máxima assincronia.
- A latência de apresentação varia conforme o atraso atual da unidade de informação associada ao valor do *prefetch*.

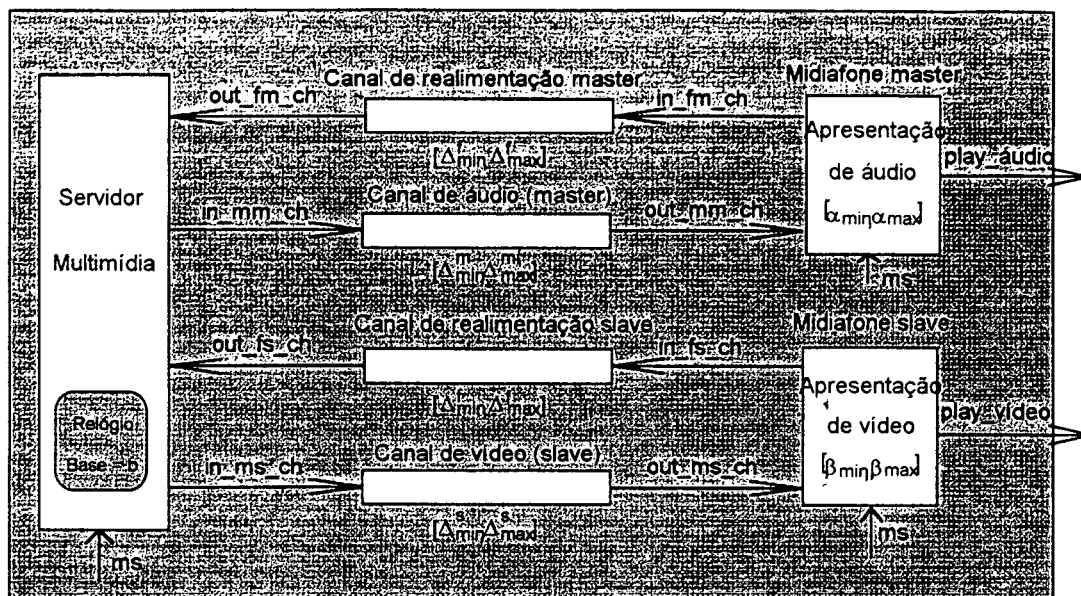


Figura 4.20 Estrutura modular Esterel do cenário de sincronização multidestino.

## Modelagem do Servidor

### A. Estrutura do Modelo do Servidor

A Figura 4.21 mostra a estrutura do modelo do servidor multimídia. A interface do módulo é dada pelos sinais:

- $out\_fm\_ch$  e  $out\_fs\_ch$ , sinais de entrada das mensagens de realimentação de *master* e *slave*.
- $in\_mm\_ch$  e  $in\_ms\_ch$ , sinais de saída das IUs dos fluxos *master* e *slave*.
- $ms$ , sinal de entrada de *tick* cada milisegundo.

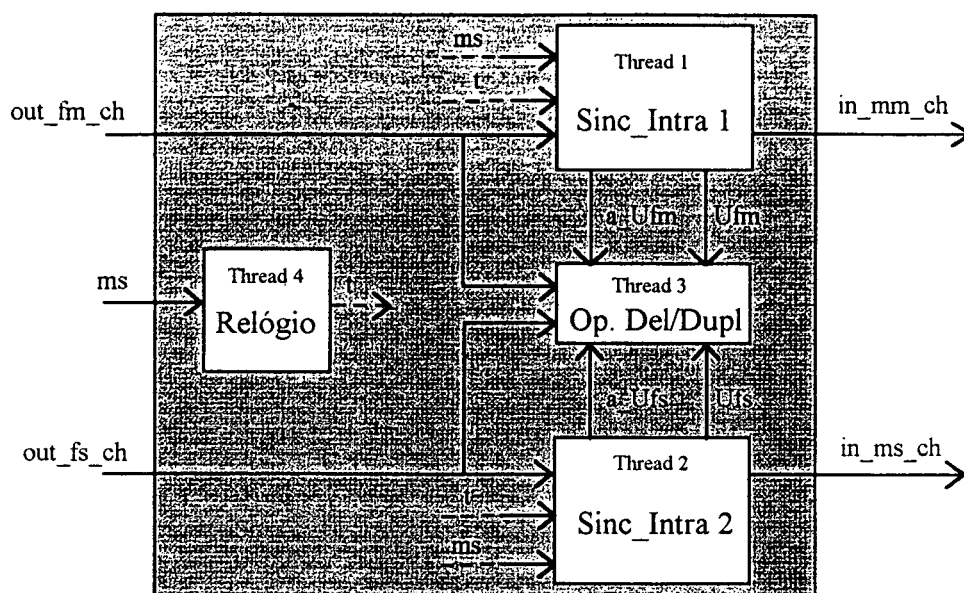


Figura 4.21 Estrutura do servidor multimídia do cenário de sincronização multideestino.

Esta especificação é composta de quatro *threads* instanciados em paralelo para cuja comunicação são declarados os seguintes sinais internos:

- *Ufm* e *Ufs*, carregam o número de seqüência das últimas mensagens de realimentação recebidas desde os midiafones *master* e *slave*, respectivamente.
- *a\_Ufm* e *a\_Ufs*, carregam o tempo de chegada das últimas mensagens de realimentação recebidas desde os midiafones *master* e *slave*, respectivamente.
- *t*, tempo atual no relógio do servidor.

O comportamento de cada *thread* é o seguinte:

1. O primeiro e segundo *thread* correspondem a duas instâncias do módulo *Sinc\_intra*, utilizado para fornecer a sincronização intra-fluxo de áudio e vídeo por separado, encapsulando o mecanismo de sincronização intra-fluxo apresentado na seção 4.2.4.4. Este módulo não será apresentado aqui, mas pode ser encontrado no Apêndice B.
2. O terceiro *thread* aguarda inicialmente pela chegada das primeiras mensagens de realimentação de áudio e vídeo. Uma vez recebidas, estima-se a dessincronização dos fluxos *master* e *slave*, através da função *calc\_shift()*. Os valores dos sinais *Ufm*, *Ufs*, *a\_Ufm* e *a\_Ufs*, providos pelas instâncias de módulo *Sinc\_intra*, são utilizados como

parâmetros neste cálculo. Seguidamente, pulam-se ou duplicam-se IUs do fluxo *slave*, no caso de ser necessário, conforme o valor de dessincronização obtido. De aqui em diante, esta estimação, e as subseqüentes operações, são realizadas para cada mensagem de realimentação recebida do midiafone *slave*.

3. O quarto *thread* corresponde à criação de uma instância do relógio local.

## B. Especificação Completa do Servidor Multimídia

```

module Servidor:
constant
  MASTER, SLAVE,
  D_MIN_MASTER, D_MAX_MASTER,
  D_MIN_SLAVE, D_MAX_SLAVE,
  TETA_MIN_MASTER, TETA_MAX_MASTER,
  TETA_MIN_SLAVE, TETA_MAX_SLAVE,
  SIZE_BUFF_MASTER, SIZE_BUFFER_SLAVE,
  N_FEEDBACK_MASTER, N_FEEDBACK_SLAVE: integer;
type
  elt_type;
input
  out_fm_ch(integer), out_fs_ch(integer), ms;
output
  in_mm_ch(elt_type), in_ms_ch(elt_type);
function
  calc_shift(integer, integer, integer, integer): integer;
procedure
  delete () (integer),
  duplicate () (integer);
signal
  Ufm(integer), Ufs(integer), a_Ufm(integer), a_Ufs(integer), t(integer) in

copymodule Sinc_intra [signal
  out_fm_ch/out_f_ch, in_mm_ch/in_m_ch, Ufm/Uf, a_Ufm/a_Uf;
  constant
  MASTER/MIDIA, D_MIN_MASTER/D_MIN, D_MAX_MASTER/D_MAX,
  SIZE_BUFF_MASTER/SIZE_BUFF, TETA_MIN_MASTER/TETA_MIN,
  TETA_MAX_MASTER/TETA_MAX, N_FEEDBACK_MASTER/N_FEEDBACK]
||
copymodule Sinc_intra [signal
  out_fs_ch/out_f_ch, in_ms_ch/in_m_ch, Ufs/Uf, a_Ufs/a_Uf;
  constant
  SLAVE/MIDIA, D_MIN_SLAVE/D_MIN, D_MAX_SLAVE/D_MAX,
  SIZE_BUFF_SLAVE/SIZE_BUFF, TETA_MIN_SLAVE/TETA_MIN,
  TETA_MAX_SLAVE/TETA_MAX, N_FEEDBACK_SLAVE/N_FEEDBACK]
||
var shift: integer in
  [await out_fm_ch || await out_fs_ch];
loop
  shift:= calc_shift(?a_Ufm, ?Ufm, ?a_Ufs, ?Ufs);
  if (shift > 0) then call delete () (shift) end;
  if (shift < 0) then call duplicate ()(-shift) end;
  await out_fs_ch

```

```

    end loop
  end var
  ||
  copymodule Relogio [signal t/curr_time]
end signal
end module

```

### C. Declarações

Os seguintes parâmetros são declarados na forma de constantes:

- *D\_MIN\_MASTER* e *D\_MAX\_MASTER*, atrasos mínimo e máximo do canal de transmissão de áudio (fluxo *master*).
- *D\_MIN\_M* e *D\_MAX\_M*, atrasos mínimo e máximo do canal de transmissão de vídeo (fluxo *slave*).
- *D\_MIN\_F* e *D\_MAX\_F*, atrasos mínimo e máximo dos canais de transmissão de mensagens de realimentação.
- *TETA\_MIN\_MASTER* e *TETA\_MAX\_MASTER*, valores mínimo e máximo de variação do período de apresentação de áudio.
- *TETA\_MIN\_SLAVE* e *TETA\_MAX\_SLAVE*, valores mínimo e máximo de variação do período de apresentação de vídeo.
- *N\_FEEDBACK\_MASTER* e *N\_FEEDBACK\_SLAVE*, períodos de rotulagem de marcas nas IUs de áudio e vídeo, respectivamente.
- *SIZE\_BUFF\_MASTER* e *SIZE\_BUFF\_SLAVE*, comprimentos dos *buffer* dos midiafones de áudio e vídeo, respectivamente.
- *MASTER* e *SLAVE*, identificam os fluxos de áudio e vídeo.

Os tipos de unidades de informação dos fluxos a sincronizar são declarados como *elt\_type*. Este tipo de dado assume a existência de um campo adicional para transportar o número de seqüência da IU.

As seguintes funções e procedimentos são ainda declarados como suporte para a especificação:



- *calc\_shift()*, esta função retorna um número inteiro equivalente ao número de unidades de dessincronização entre os fluxos *master* e *slave*. Um valor positivo (negativo) indica que o fluxo *slave* está atrasado (adiantado) respeito ao fluxo *master*. Esta função recebe como argumentos os números de seqüência e os tempos de chegadas das últimas mensagens de realimentação recebidas desde os midiafones *master* e *slave*.
- *delete()*, este procedimento faz com que, na seqüência de envios, sejam omitidas as próximas  $N$  IUs do fluxo *slave*, visando compensar o atraso do fluxo *slave* respeito ao fluxo *master*. O numero  $N$ , de IUs a omitir, é passado como parâmetro deste procedimento.
- *duplicate()*, este procedimento faz com que a próxima unidade de informação seja objeto dos seguintes  $N$  envios, visando compensar o adiantamento deste fluxo respeito ao fluxo *master*. O numero  $N$ , de envios a repetir é passado como parâmetro deste procedimento.

### **Modelagem do Sistema Global**

O módulo principal é composto das seguintes instâncias em paralelo (ver Figura 4.20):

- Uma instância do módulo Servidor Multimídia.
- Quatro instâncias do módulo Canal de transmissão, representando os canais de áudio e vídeo, e seus canais de realimentação.
- Duas instâncias do módulo Midiafone, representando os midiafones de áudio e vídeo.

Os canais de realimentação apresentam atrasos no intervalo [6,15] ms, o canal de áudio no intervalo [15,20] ms, e o canal de vídeo no intervalo [20,30] ms. O midiafone de áudio tem um período de apresentação que varia no intervalo [29,31] ms. O período de apresentação no midiafone de vídeo varia no intervalo [39,41] ms. Cada midiafone tem um *buffer* de 5 unidades de comprimento, e deve respeitar o *prefetch* de 1 unidade de informação. Conforme estes valores, estimaram-se as taxas mínimas de mensagens de realimentação necessárias para manter a continuidade em ambos fluxos: `N_FEEDBACK_MASTER` de período 65 e `N_FEEDBACK_SLAVE` de período 87. O Servidor é instanciado com todos estes parâmetros do sistema.

```

module feedback:
constant
  D_MIN_F, D_MAX_F,           % [6,15] ms
  D_MIN_MASTER, D_MAX_MASTER, % [15,20] ms
  D_MIN_SLAVE, D_MAX_SLAVE,   % [20,30] ms
  PREFETCH_MASTER, PREFETCH_SLAVE, % 1, 1
  TETA_MIN_MASTER, TETA_MAX_MASTER, % [29,31] ms
  TETA_MIN_SLAVE, TETA_MAX_SLAVE, % [39,41] ms
  SIZE_BUFF_MASTER, SIZE_BUFF_SLAVE, % 5, 5 unidades
  N_FEEDBACK_MASTER, N_FEEDBACK_SLAVE % 65, 87 unidades
: integer;
type
  elt_type;
input
  ms;
output
  play_master(integer), play_slave(integer);

signal
  in_m_ch(elt_type), out_m_ch(elt_type), in_fm_ch(integer), out_fm_ch(integer),
  in_s_ch(elt_type), out_s_ch(elt_type), in_fs_ch(integer), out_fs_ch(integer) in

  copymodule Servidor
  ||
  copymodule Canal [signal in_m_ch/in_ch, out_m_ch/out_ch;
                    constant D_MIN_MASTER/D_MIN, D_MAX_MASTER/D_MAX]
  ||
  copymodule Canal [signal in_s_ch/in_ch, out_s_ch/out_ch;
                    constant D_MIN_SLAVE/D_MIN, D_MAX_SLAVE/D_MAX]
  ||
  copymodule Canal [signal in_fm_ch/in_ch, out_fm_ch/out_ch;
                    constant D_MIN_F/D_MIN, D_MAX_F/D_MAX]
  ||
  copymodule Canal [signal in_fs_ch/in_ch, out_fs_ch/out_ch;
                    constant D_MIN_F/D_MIN, D_MAX_F/D_MAX]
  ||
  copymodule Midiafone [signal out_m_ch/midiafone_in,
                        play_master/play,
                        in_fm_ch/feedback_out;
                        constant PREFETCH_MASTER/PREFETCH,
                        TETA_MIN_MASTER/TETA_MIN,
                        TETA_MAX_MASTER/TETA_MAX,
                        SIZE_BUFF_MASTER/SIZE_BUFF]
  ||
  copymodule Midiafone [signal out_s_ch/midiafone_in,
                        play_slave/play,
                        in_fs_ch/feedback_out;
                        constant PREFETCH_SLAVE/PREFETCH,
                        TETA_MIN_SLAVE/TETA_MIN,
                        TETA_MAX_SLAVE/TETA_MAX,
                        SIZE_BUFF_SLAVE/SIZE_BUFF]

end signal
end module

```

## 4.5 Conclusão

Neste capítulo foi mostrada a utilização da linguagem Esterel na especificação de problemas de sincronização em sistemas multimídia distribuídos. Em particular foram especificadas várias técnicas de abordagem temporal nos domínios de sincronização intra-fluxo, inter-fluxo e multidestino. Foram expostos e discutidos aspectos específicos de cada técnica.

A estratégia de especificação baseou-se, em princípio, na identificação de módulos e ou estruturas genéricas reutilizáveis nos diversos esquemas. Em particular, a estrutura do módulo *Canal de Comunicação*, serviu de esqueleto para a construção de vários dos módulos sincronizadores os quais se ajustaram a uma mesma estrutura funcional. A especificação de cada módulo exigiu uma completa identificação de seus elementos. Assim foi necessário a definição de sinais de entrada e saída, tipos de dados, funções e procedimentos para a geração de um código executável que representa a especificação.

A hipótese síncrona na linguagem Esterel proporcionou uma abstração simples e eficiente para a especificação dos aspectos de sincronização multimídia. Assim, os fluxos de dados foram modelados como sinais e eficientemente manuseados pelas construções temporais oferecidas pela linguagem. O tempo foi tratado como um sinal de entrada ao sistema, e manuseado com as mesmas construções temporais dos outros sinais conseguindo-se assim, um tratamento homogêneo dos aspectos temporais da especificação.

Uma vez realizada a modelagem, no capítulo seguinte será apresentada a validação de cada um dos cenários especificados, através de ferramentas que permitem a compilação e simulação de especificações na linguagem Esterel.

# Capítulo 5

## *Validação e Avaliação das Especificações*

### **5.1 Introdução**

Neste capítulo é apresentado o processo de validação das especificações dos problemas de sincronização multimídia introduzidos no capítulo anterior com o objetivo de mostrar as potencialidades do modelo Esterel e das ferramentas associadas. É descrita a validação através da simulação de cada um dos cenários, apresentando critérios para a escolha de casos de teste, o atendimento da qualidade de serviço esperada e a identificação de condições de erro. Vários exemplos ilustram as simulações realizadas. No final do capítulo são apresentadas algumas conclusões.

### **5.2 Simulação dos Cenários de Sincronização Intra-Fluxo**

#### **5.2.1 Aspectos Gerais da Simulação**

O processo de simulação em geral visou conferir o atendimento da qualidade de serviço esperada para cada cenário especificado. Nos três primeiros cenários, caracterizados pela sincronização de fluxos estocásticos em sistemas de relógios comuns (Cenário 1), fluxos estocásticos em sistemas de relógios a derivas comuns (Cenário 2) e fluxos periódicos em sistemas de relógios a derivas comuns (Cenário 3), foram considerados os seguintes itens:

- Preservação da assinatura temporal do fluxo.
- Atendimento dos limites de latência de apresentação.
- Condições de utilização do *buffer* do sincronizador.

A simulação do quarto cenário, dedicado à sincronização de fluxos periódicos em sistemas de relógios a derivas constantes, visou verificar, além dos últimos dois itens anteriores, a correta alteração da assinatura temporal.

No cenário de sincronização de fluxos periódicos em sistemas de relógios não sincronizados (Cenário 5) o interesse principal da simulação centrou-se na constatação da preservação da continuidade de apresentação, i.e. a não ocorrência de saturação ou fome no *buffer* do receptor.

Para a identificação de eventuais condições de erro foi utilizado um módulo observador dos sinais de entrada e saída, dos módulos: Emissor e Sincronizador de cada sistema. O Observador é instanciado em paralelo com os módulos: Emissor, Canal e Sincronizador. Para o cinco cenários são utilizados três módulos observadores que utilizam princípios de validação similares: o cálculo da latência de apresentação e realizado subtraindo o tempo de apresentação da primeira IU, do seu correspondente tempo de emissão; o estado do *buffer* é determinado pela diferença do número atual de entradas e saídas de IUs no sincronizador; a preservação ou correta alteração da assinatura temporal é verificada pela comparação entre as relações temporais das IUs geradas pelo emissor e aquelas apresentadas. No caso destas condições não corresponderem às esperadas, um sinal de erro é emitido. A especificação destes módulos pode ser vista no Apêndice C.

Adicionalmente foi avaliado o desempenho dos módulos sincronizadores utilizando o código da especificação gerado pelo compilador Esterel, acrescentado com uma interface de execução geradora de estatísticas.

## 5.2.2 Simulação do Primeiro Cenário

O mecanismo de sincronização baseado em *time-stamping* [Santoso93], especificado na seção 4.2.4.1, foi objeto desta simulação.

Nesta proposta os valores de latência ( $L$ ) e requisito de armazenamento do *buffer* receptor ( $B$ ) vem sendo dados por:

- $L = \Delta_{\max}$
- $B = \lceil (\Delta_{\max} - \Delta_{\min}) / s_{\min} \rceil$

Onde  $\Delta_{\min}$  e  $\Delta_{\max}$  representam os valores de atrasos mínimo e máximo do fluxo envolvido, e  $s_{\min}$  denota o mínimo tempo entre a emissão de uma IU e a IU subsequente. A notação  $\lceil x \rceil$  indica o valor inteiro superior ou igual a  $x$ .

As condições limites identificadas neste sistema foram relativas à utilização do *buffer* receptor. Especificamente, para valores máximos de atrasos de comunicação é conseguida a mínima utilização do *buffer*. Valores inferiores de atrasos fazem com que esta utilização aumente, sendo que, no caso da emissão de um fluxo de período  $\theta$  ( $s_{\min} = \theta$ ), atinge-se o valor máximo quando todos os atrasos atuais são mínimos e a proporção  $(\Delta_{\max} - \Delta_{\min})/\theta$  constitui um número inteiro. Isto é devido ao fato de que valores inteiros  $n = (\Delta_{\max} - \Delta_{\min})/\theta$ , produzem só entradas e saídas simultâneas de IUs no *buffer*, depois deste ter atingido uma carga máxima numa fase inicial.

A seguinte simulação ilustra alguns dos resultados mais importantes.

### Simulação

Utilizaram-se os valores seguintes: período do fluxo  $\theta = 5$  ms, atrasos de rede no intervalo [10, 15] ms, capacidade de armazenamento  $B = 1$ , e valor de tempo de apresentação alvo  $TPT = 15$ . Todos os atrasos atuais das IUs foram forçados a ser mínimos (10 ms), visando obter uma máxima utilização do *buffer*, ou originar uma situação de saturação no caso de algum erro de especificação. A Tabela mostra os resultados desta simulação.

Tempo relativo ao começo da emissão de IUs (ms).	Eventos de Entrada e/ou Saída de IUs no sincronizador.	Estado do buffer em número de IUs.	Número de seqüência da IU apresentada
10	E	1	
15	E/S	1	1
20	E/S	1	2
25	E/S	1	3
30	E/S	1	4
35	E/S	1	5

**Tabela 5.1 Resultados da simulação do esquema de sincronização intra-fluxo baseado em time-stamping, num sistema de relógios comuns.**

Nesta tabela é possível identificar que o valor de latência, ou tempo de apresentação da primeira unidade de informação, é 15 ms. O período de apresentação corresponde-se com o período de emissão (5 ms), respeitando-se assim a assinatura temporal do fluxo. O *buffer* do sincronizador foi utilizado ao máximo como esperado, devido a que as IUs foram forçadas a sofrer atrasos mínimos, e o valor da proporção  $(\Delta_{\max} - \Delta_{\min})/\theta$  constitui um número inteiro (1). É importante mencionar que após a chegada da primeira unidade de informação todas as entradas e saídas do sincronizador ocorreram no mesmo instante deixando inalterado o estado do *buffer* na sua carga

visando provocar uma situação de fome no caso de um erro de especificação, ou bem como esperado, a mínima utilização do *buffer* receptor. A latência esperada é a mínima, e corresponde ao valor de  $\Delta_{\max}$  (25 ms). A Tabela mostra os resultados obtidos.

Tempo relativo ao começo da emissão de IUs (ms).	Eventos de Entrada e/ou Saída de IUs no sincronizador.	Estado do buffer em número de IUs.	Número de seqüência da IU apresentada.
10	E	1	
25	S	0	1
55	E/S	0	2
85	E/S	0	3
115	E/S	0	4
145	E/S	0	5

**Tabela 5.2 Resultados da primeira simulação do esquema de sincronização intra-fluxo baseado em time-stamping, num sistema de relógios a derivas comuns.**

Nesta tabela tem-se um breve regime transitório até 55 ms, antes da entrada do sistema ao regime permanente onde só ocorrem entradas e saídas simultâneas. A latência de apresentação é de 25 ms como esperada. O período de apresentação corresponde-se com o período de emissão, respeitando-se assim a assinatura temporal do fluxo. O *buffer* do sincronizador foi utilizado ao mínimo como esperado, sendo que as IUs que entram no sincronizador são apresentadas nesse mesmo instante, deixando o *buffer* vazio.

### Segunda Simulação

Numa outra simulação variou-se apenas os atrasos de rede entre [10, 15] ms. O atraso atual da primeira IU foi forçado ao máximo (15 ms). As IUs seguintes foram forçadas a sofrer atrasos mínimos. A latência esperada é a máxima, e corresponde ao valor de  $2 \cdot \Delta_{\max} - \Delta_{\min}$  (20 ms). A Tabela mostra os resultados desta simulação.

Tempo relativo ao começo da emissão de IUs (ms).	Eventos de Entrada e/ou Saída de IUs no sincronizador.	Estado do buffer em número de IUs.	Número de seqüência da IU apresentada.
15	E	1	
20	S	0	1
40	E	1	
50	S	0	2
70	E	1	
80	S	0	3
100	E	1	
110	S	0	4

**Tabela 5.3 Resultados da segunda simulação do esquema de sincronização intra-fluxo baseado em time-stamping, num sistema de relógios a derivas comuns.**

Identifica-se nesta tabela uma latência de apresentação de 20 ms como esperada. O período de apresentação (30 ms) corresponde-se com o período de emissão, respeitando-se assim a assinatura temporal do fluxo. A utilização do *buffer* ficou em torno do 33%, isto é, o 33% do tempo o *buffer* armazenou uma unidade de informação, enquanto o tempo restante ficou vazio. Note-se que apesar do tipo de atrasos induzidos, a máxima utilização do *buffer* não foi atingida, devido à defasagem entre os instantes de entrada e saída de IUs no *buffer*, pois a proporção  $2 * (\Delta_{\max} - \Delta_{\min}) / \theta$  não constitui um número inteiro.

### 5.2.4 Simulação do Terceiro Cenário

A técnica do *prefetch* [Dairaine94] especificada na seção 4.2.4.3, foi objeto desta simulação.

Para esta técnica os valores *prefetch* (P), de latência (L) e requisito de armazenamento do *buffer* receptor (B) vem sendo dados por:

- $P = \lceil (\Delta_{\max} - \Delta_{\min}) / \theta_{\text{nom}} \rceil$
- $L \in [\theta_{\text{nom}} * P + \Delta_{\min}, \theta_{\text{nom}} * P + \Delta_{\max}]$
- $B = 2 * P$

Onde  $\theta_{\text{nom}}$  denota o período nominal do fluxo.

Nesta proposta a latência de apresentação depende do atraso atual da IU associada ao valor do *prefetch*. Assim p. ex., um valor de  $P = 1$  faz com que o tempo de chegada da segunda IU ao sincronizador determine o início da apresentação. A capacidade de armazenamento é também



fixada conforme ao valor do *prefetch*. Observe-se que estes valores de latência e requisitos de armazenamento são superiores àqueles dos esquemas anteriores.

As condições limites identificadas neste sistema foram relativas à utilização do *buffer* receptor e à latência de apresentação. Em particular, o atraso mínimo da IU associada a P, seguido de atrasos máximos de posteriores IUs, produz a menor latência de apresentação e requer a mínima utilização do *buffer* receptor. Pelo contrario, o atraso máximo da IU associada a P, seguido de atrasos mínimos das seguintes IUs, origina a maior latência de apresentação e aumenta a utilização do *buffer*, sendo que a máxima utilização é só atingida quando a proporção  $(\Delta_{\max} - \Delta_{\min})/\theta_{\text{nom}}$  resulta num número inteiro. Nestas circunstâncias produzem-se só entradas e saídas simultâneas de IUs no *buffer*, depois deste ter atingido sua carga máxima numa fase inicial.

### Simulação

Nesta simulação, fixou-se o valor do período do fluxo  $\theta_{\text{nom}} = 30$  ms, os atrasos de rede no intervalo [10, 40] ms, a capacidade de armazenamento  $B = 2$ , e o valor de *prefetch*  $P = 1$ . O atraso atual da IU associada a P, (a segunda IU) foi forçado a sofrer um atraso de 40 ms (máximo). As IUs seguintes foram forçadas a sofrer atrasos mínimos. Sob estas condições, e devido que  $(\Delta_{\max} - \Delta_{\min})/\theta_{\text{nom}} = 1$ , espera-se provocar uma situação de máxima utilização do *buffer* receptor em regime permanente. Neste caso a latência esperada corresponde ao valor de  $\theta_{\text{nom}}*P+\Delta_{\max}$ , i.e. 70 ms. A Tabela mostra os resultados desta simulação.

Tempo relativo ao começo da emissão de IUs (ms).	Eventos de Entrada e/ou Saída de IUs no sincronizador.	Estado do buffer em número de IUs.	Número de seqüência da IU apresentada.
40	E	1	
70	E/S	1	1
71	E	2	
100	E/S	2	2
130	E/S	2	3
160	E/S	2	4

**Tabela 5.4** Resultados da simulação do esquema de sincronização intra-fluxo baseado na técnica do *prefetch*, num sistema de relógios a derivas comuns.

Nesta tabela tem-se um breve regime transitório até 100 ms antes da entrada do sistema ao regime permanente onde só ocorrem saídas e entradas simultâneas. O início da apresentação corresponde

ao mesmo instante da chegada da segunda IU ao sincronizador (70 ms), assim conseguindo-se um *prefetch* de uma unidade (estado do *buffer* = 1). O período de apresentação corresponde-se com o período de emissão, respeitando-se assim a assinatura temporal do fluxo. Em regime permanente, o *buffer* do sincronizador foi utilizado a sua máxima capacidade, sendo que as IUs que entram no sincronizador são apresentadas nesse mesmo instante, deixando o *buffer* cheio.

### 5.2.5 Simulação do Quarto Cenário

A técnica do emulador de período [Dairaine94] especificada na seção 4.2.4.4, foi objeto desta simulação.

Para este mecanismo, os valores *prefetch* (P), de latência (L) e requisito de armazenamento do *buffer* receptor (B) vem sendo dados por:

- $P = \lceil (\Delta_{\max} - \Delta_{\min}) / \theta_{in} \rceil$
- $L \in [\theta_{in} * P + \Delta_{\min}, \theta_{in} * P + \Delta_{\max}]$
- $B = 2 * P + 1$

Onde  $\theta_{in}$  denota o período de emissão do fluxo.

Nesta proposta a latência de apresentação depende do atraso atual da IU associada ao valor do *prefetch*, tal qual a proposta anterior. A capacidade de armazenamento é também fixada conforme ao valor do *prefetch*. Observe-se que o valor de requisito de armazenamento é superior numa unidade a seu correspondente valor na proposta anterior.

As condições limites identificadas neste sistema foram relativas à utilização do *buffer* receptor e à latência de apresentação. Uma vez que este esquema utiliza a técnica do *prefetch*, para compensar o *jitter* de comunicação, foram consideradas condições limites semelhantes àquelas do esquema anterior. Entretanto, neste caso a maior utilização do *buffer*, em regime permanente, não coincide com a utilização de sua máxima capacidade devido a que a virtual diferença, entre os períodos de emissão e apresentação, produz uma defasagem entre as atuais entradas e saídas de IUs no *buffer receptor* que determina uma oscilação contínua da sua carga. Esta situação é ilustrada numa das simulações do esquema.

As seguintes simulações ilustram alguns dos resultados mais importantes.

### Primeira Simulação

Esta simulação corresponde ao caso do emulador de período com operações *delete*. Neste caso, fixou-se o valor do período de emissão do fluxo  $\theta_{in} = 30$  ms, o período de apresentação  $\theta_{out} = 40$  ms, os atrasos de rede no intervalo  $[10, 40]$  ms, a capacidade de armazenamento  $B = 3$ , e o valor de *prefetch*  $P = 1$ . Para estes períodos de emissão e apresentação corresponde um período de emulação  $\theta_{emu} = 120$  ms. O atraso atual da IU associada a  $P$ , (a segunda IU) foi forçado a sofrer um atraso máximo. As IUs seguintes foram forçadas a sofrer atrasos mínimos. Sob estas condições, e devido ao fato que a relação  $(\Delta_{max} - \Delta_{min})/\theta_{in} = 1$ , espera-se provocar a maior utilização do *buffer* receptor. A latência esperada corresponde ao valor de  $\theta_{in} * P + \Delta_{max}$ , i.e. 70 ms. Na Tabela são apresentados os resultados desta simulação, sendo que as operações *delete* são ressaltadas com asteriscos.

Tempo relativo ao começo da emissão de IUs (ms).	Entrada e/ou Saída de IUs no sincronizador.	Estado do buffer em número de IUs.	IU apresentada.
40	E	1	
70	E/S	1	1
71	E	2	
100	E	3	
110	S	2	2
130	E	3	
150	S	2	3
160	E	3	
190	E/S *	2	5
220	E	3	
230	S	2	6
250	E	3	
270	S	2	7
280	E	3	
310	E/S *	2	9

Tabela 5.5 Resultados da primeira simulação da técnica do emulador de período, num sistema de relógios a derivas constantes.

Nesta tabela observa-se um regime transitório até 190 ms, instante no qual ocorre a primeira, de uma seqüência de operações *delete*, com período  $\theta_{emu}$  (120 ms). O início da apresentação corresponde ao mesmo instante da chegada da segunda IU ao sincronizador (70 ms), assim conseguindo-se um *prefetch* de uma unidade. Em regime permanente, a maior utilização do *buffer* correspondeu ao armazenamento de 2 IUs durante o 50% do tempo, e de 3 IUs o restante 50%.

As operações periódicas *delete*, fizeram com que o fluxo apresentado corresponder a uma dilatação do fluxo gerado, ao mesmo tempo impedindo a saturação do *buffer* receptor. A seqüência de IUs apresentadas foi: 1, 2, 3, 5, 6, 7 e 9. A Figura 5.1 ilustra o efeito de dilatação.

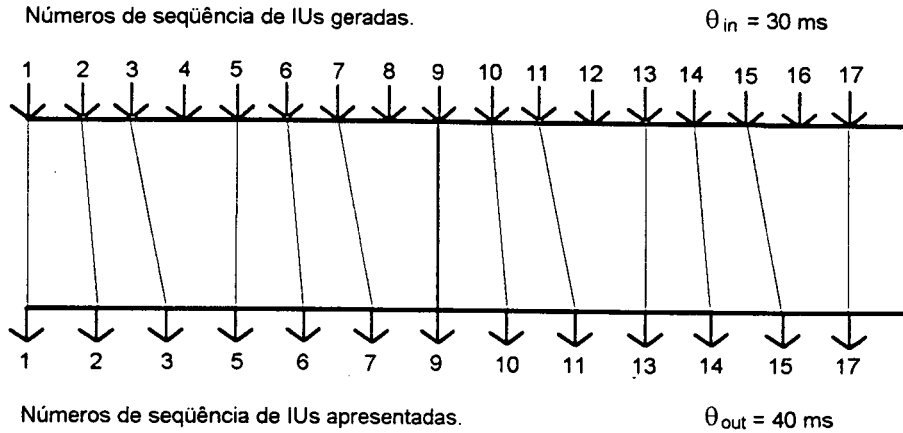


Figura 5.1 Dilatação de um fluxo sob o esquema do emulador de período.

Nesta figura pode-se visualizar que as operações *delete* compensam periodicamente o atraso cumulativo da assinatura temporal do fluxo apresentado. Esta propriedade é fundamental para a aplicação desta técnica no caso da sincronização com outros fluxos.

### Segunda Simulação

Esta simulação corresponde a uma sessão de simulação do emulador de período com operações *duplicate*. Neste caso, fixou-se o valor do período de emissão do fluxo  $\theta_{in} = 40$  ms, o período de apresentação  $\theta_{out} = 30$  ms, os atrasos de rede no intervalo [10, 50] ms, a capacidade de armazenamento  $B = 3$ , e o valor de prefetch  $P = 1$ . Para estes períodos de emissão e apresentação corresponde um período de emulação  $\theta_{emu} = 120$  ms. Os atrasos atuais das IUs são aleatórios e em consequência, a latência esperada está contida no intervalo [50, 90] ms. A mostra os resultados obtidos nesta simulação, sendo que as operações *duplicate* são ressaltadas por asteriscos.

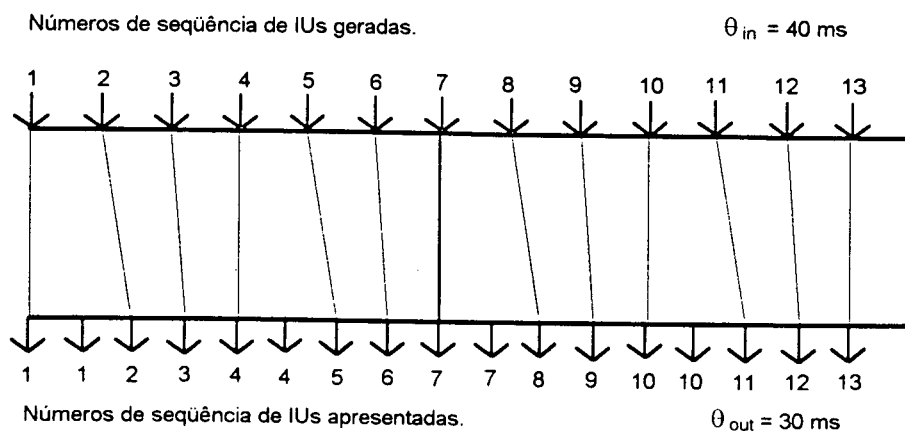
Tempo relativo ao começo da emissão de IUs (ms).	Entrada e/ou Saída de IUs no sincronizador.	Estado do buffer em número de IUs.	IU apresentada.
28	E	1	
57	E/S *	2	1
87	S	1	1
117	S	0	2
125	E	1	

147	S	0	3
154	E	1	
177	S *	1	4
182	E	2	
207	S	1	4
234	E	2	
237	S	1	5
256	E	2	
267	S	1	6
297	S *	1	7

**5.6 Resultados da segunda simulação da técnica do emulador de período, num sistema de relógios a derivas constantes.**

Nesta tabela pode-se observar que o início da apresentação corresponde ao mesmo instante da chegada da segunda IU ao sincronizador (57 ms), sendo que este valor está contido no intervalo [50, 90] ms como esperado. Adicionalmente, devido ao fato dos atrasos serem aleatórios, a utilização do *buffer* varia conforme os atrasos atuais das IUs. Neste caso, a maior carga do *buffer* correspondeu ao armazenamento de 2 IUs, não atingindo a capacidade máxima do *buffer* (3 unidades).

As operações periódicas *duplicate*, fizeram com que o fluxo apresentado corresponder a uma compressão do fluxo gerado, ao mesmo tempo impedindo a fome do *buffer* receptor. A seqüência de IUs apresentadas foi: 1, 1, 2, 3, 4, 4, 5, 6, 7 e 7. O efeito de compressão ilustra-se na Figura 5.2.



**Figura 5.2 Exemplo de compressão de um fluxo sob o esquema do emulador de período.**

Nesta figura pode-se visualizar que as operações *duplicate*, aplicadas com período  $\theta_{emu}$ , impedem o adiantamento contínuo da assinatura temporal do fluxo apresentado. Similarmente ao

caso da emulação de período com operações *delete*, esta propriedade é fundamental para a aplicação desta técnica no caso da sincronização com outros fluxos.

### 5.2.6 Simulação do Quinto Cenário

O mecanismo de sincronização intra-fluxo baseado em mensagens de realimentação [Ramanathan93] especificado na seção 4.2.4.5, foi objeto desta simulação.

Neste esquema foram exploradas, em particular, as condições que provocam a maior e menor carga do *buffer*. Assim por exemplo, uma carga máxima do *buffer* é provocada com a combinação dos seguintes fatores:

- O atraso máximo da primeira IU, seguido de atrasos mínimos das IUs posteriores.
- Um período máximo de apresentação de IUs no midiafone.
- Uma política de envios de IUs no tempo mais próximo possível.

Os seguintes parâmetros serviram de base para um conjunto de simulações relativas à sincronização de um fluxo de áudio:

- Atrasos de transporte fim-a-fim das IUs entre [40,60] ms.
- Atrasos de transporte fim-a-fim das mensagens de realimentação entre [1,15] ms.
- Período de apresentação de IUs no midiafone,  $\theta = 30$  ms, sendo que é possível uma variação de 0,1 ms (*drift* do relógio de  $\pm 0,33\%$ ).

Para estes parâmetros, o número máximo de IUs entre envios de mensagens de realimentação ou período de realimentação ( $N$ ), é função da capacidade de armazenamento disponível no midiafone. A Tabela mostra os valores do período de realimentação, para diferentes valores de capacidade de armazenamento.

Capacidade do Buffer (unid. de áudio)	Período de realimentação, N.
2	100
3	249
4	399
5	546
6	698
7	847
8	997

**Tabela 5.7** Períodos de realimentação correspondentes a distintos valores de capacidade de armazenamento.

Nesta tabela, observam-se incrementos do período de realimentação, em torno de 150 IUs, por incrementos de uma unidade na capacidade de armazenamento, ou seja, uma relação linear entre estes valores na faixa considerada. Assim optou-se, sem nenhuma consideração especial, pelo valor de capacidade de 5 unidades com período de realimentação igual a 546 IUs.

A Tabela resume os resultados do conjunto de simulações para um tempo de simulação equivalente a 100.000 ms.

Atrasos	Envios	$\theta$	B = 0	B = 1	B = 2	B = 3	B = 4	B = 5
FERL	Latest	min	75,87	24,13	0,00	0,00	0,00	0,00
FERL	Latest	max	8,72	26,76	27,20	26,81	10,52	0,00
FERL	Earliest	min	7,10	22,99	26,69	26,99	15,93	0,31
FLRE	Latest	max	0,07	14,39	27,09	27,20	25,14	6,11
FERL	Earliest	max	0,45	0,90	0,90	3,15	83,16	11,44
FLRE	Earliest	max	0,00	0,20	0,85	0,90	21,33	76,72

**Tabela 5.8** Respostas do simulador do esquema de sincronização intra-fluxo baseado em mensagens de realimentação.

Nesta tabela a coluna *Atrasos*, representa os atrasos atuais das IUs. Tem-se aqui dois tipos de atrasos: FLRE (*First-Late-Rest-Early*), caso onde só o atraso da primeira IU é máximo, e os restantes são mínimos; e FERL (*First-Early-Rest-Late*) onde só o atraso da primeira IU é mínimo, e os restantes são máximos. A coluna *Envios*, representa a política de envios do Servidor Multimídia. Esta política pode ser para os tempos de envio mais próximos possíveis (Earliest) ou bem para o último instante de envio possível (Latest). A coluna de cabeçalho  $\theta$ , representa o período atual de apresentação do midiafone. Os valores máximo e mínimo foram considerados aqui. Finalmente, as colunas com cabeçalhos  $B = n$  (com  $n = 0, 1, 2, 3, 4$  e  $5$ ), representam os percentuais de utilização do *buffer* para cada um dos seus estados de carga.

Estes resultados mostram como aumenta a utilização do *buffer* segundo estes três fatores considerados. Num extremo a tupla <FERL, Latest, min> concentra a carga do *buffer* nos estados: vazio (75,87%), e de uma unidade (24,13%); em outro extremo a tupla <FLRE, Earliest, max> produz a concentração da carga nos estados: cheio (76,72%), e de quatro unidades (21,33%).

Cabe ressaltar que o valor do período de realimentação utilizado ( $N = 546$ ), foi conferido experimentalmente. As simulações mostraram que valores superiores a 546 unidades produzem situações de descontinuidade. Em particular atrasos tipo FLRE, combinados com um período de apresentação máximo no midiafone, produzem fome antes da chegada da primeira mensagem de realimentação ao Servidor.

### 5.3 Simulação do Cenário de Sincronização Inter-Fluxo

A extensão do esquema de sincronização intra-fluxo baseado em *time-stamping* [Santoso93] para fluxos periódicos em sistemas de relógios comuns, especificado na seção 4.3.2, foi também simulada.

Os critérios de simulação deste esquema de sincronização coincidiram em geral com os utilizados na simulação do esquema de sincronização intra-fluxo sob a mesma hipótese de relógios comuns do sistema. A especificação consistiu da instanciação em paralelo de módulos sincronizadores intra-fluxo, independentes entre si, com seus respectivos emissores e canais de comunicação. Os diferentes parâmetros dos fluxos envolvidos deveram ser considerados globalmente.

Em particular, para o tempo de apresentação alvo,  $TPT$ , fixado ao máximo atraso de comunicação dos fluxos envolvidos, o valor de latência comum ( $L$ ) e o requisito de armazenamento de cada *buffer* receptor ( $B^i$ ) vem sendo dado por:

- $L = TPT$
- $B^i = \lceil (TPT - \Delta_{\min}^i) / s_{\min}^i \rceil$

Onde  $\Delta_{\min}^i$  corresponde ao atraso mínimo de comunicação do fluxo  $i$ , e  $s_{\min}^i$  denota o mínimo tempo entre a emissão de uma IU e a IU subsequente, no fluxo  $i$ .



Nas diferentes simulações, tal qual era esperado, os fluxos envolvidos não apresentaram assincronia. Isto deveu-se principalmente ao fato de que o esquema de sincronização intra-fluxo, aplicado a cada fluxo, garante uma latência fixa, conseguindo assim uma sincronização inicial dos fluxos que é mantida posteriormente pela preservação individual de suas assinaturas temporais.

Em relação à utilização dos *buffers* o aspecto mais importante foi que o requisito de armazenamento do fluxo mais rápido, viu-se incrementado pelo valor de máximo atraso do fluxo mais lento (valor do *TPT*).

#### **5.4 Simulação do Cenário de Sincronização Multidestino**

O mecanismo de sincronização inter-fluxo baseado em mensagens de realimentação [Ramanathan93] especificado na seção 4.4.2, foi objeto desta simulação.

Esta proposta de sincronização inter-fluxo garante a continuidade dos fluxos periódicos em sistemas de relógios não sincronizados com a técnica aplicada no seu semelhante intra-fluxo. O interesse da simulação centrou-se especificamente na assincronia (*skew*) entre fluxos.

Em particular, a assincronia depende do período de realimentação dos fluxos envolvidos, sendo que para menores períodos de realimentação obtém-se uma menor assincronia entre fluxos. Esta relação entre assincronia e período de realimentação foi verificada a partir da simulação.

Os seguintes parâmetros serviram de base para o conjunto de simulações relativas à sincronização de fluxos de áudio (fluxo *master*) e vídeo (fluxo *slave*):

- Atrasos de transporte fim-a-fim das IUs entre [40,60] ms.
- Atrasos de transporte fim-a-fim das mensagens de realimentação incluídos entre [1,15] ms.
- Período de apresentação de áudio,  $\theta_a = 30$  ms, sendo que é possível uma variação de 0,1 ms (*drift* do relógio de  $\pm 0,33\%$ ).
- Período de apresentação de vídeo,  $\theta_v = 40$  ms, sendo que é possível uma variação de 0,1 ms (*drift* do relógio de  $\pm 0,25\%$ ).
- Capacidade de armazenamento dos dois *buffers* igual a 5.

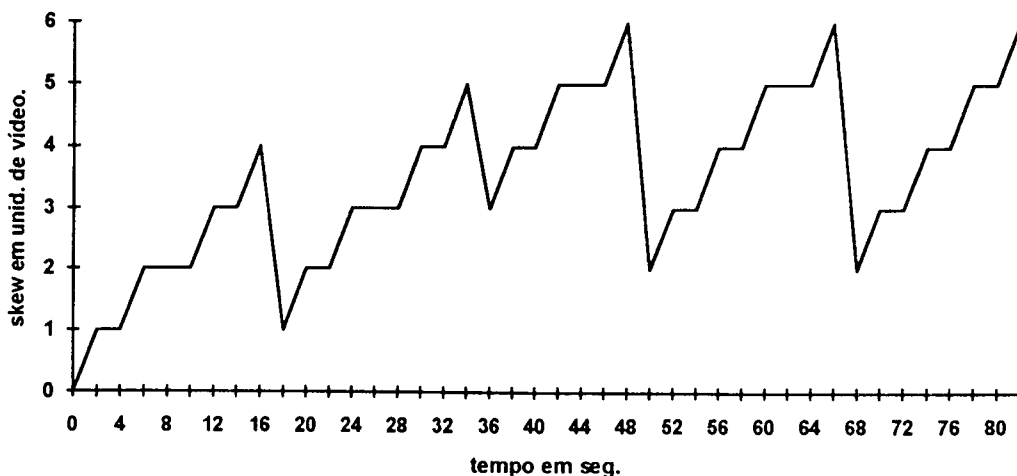
- Períodos máximos de realimentação dos fluxos:  $N_{\text{áudio}} = 546$  e  $N_{\text{vídeo}} = 798$ .

As simulações foram realizadas com o código da especificação gerada pelo compilador, ligada com uma interface de execução geradora de estatísticas da assincronia entre fluxos. Para efeito de estudo, o período de apresentação do fluxo de áudio foi forçado a ser mínimo, e o período de vídeo foi forçado a ser máximo, visando avaliar o mecanismo sob estas condições extremas de apresentação. A Tabela resume os resultados para um tempo de simulação equivalente a 1.000 seg.

$N_{\text{áudio}}$	$N_{\text{vídeo}}$	Skew médio (Unid. de vídeo)	Skew médio (ms)	Skew máximo (Unid. de vídeo)	Skew máximo (ms)
546	798	3,06	123	6,21	248
273	399	1,55	62	3,42	137
200	300	1,15	46	2,76	110
137	200	0,75	30	2,08	83

**Tabela 5.9** Valores de assincronia para diferentes períodos de realimentação, no esquema de sincronização multidestino baseado em mensagens de realimentação.

Considerando um valor máximo de assincronia aceitável de 150 ms [Stefani92], e os parâmetros escolhidos, o mecanismo só consegue atender este requisito para valores de  $N_{\text{áudio}}$  e  $N_{\text{vídeo}}$  próximos à metade dos máximos períodos de realimentação. Isto é, valores de  $N_{\text{áudio}}$  e  $N_{\text{vídeo}}$  próximos a 273 e 399 respectivamente. Por outro lado, independentemente dos valores de parâmetros escolhidos, o esquema apresenta uma oscilação da assincronia durante todo o tempo de apresentação do fluxo. A Figura 5.3 mostra esta variação para os valores máximos de períodos de realimentação, durante 80 segundos de apresentação.



**Figura 5.3** Variação da assincronia de apresentação entre fluxos áudio e vídeo, em unidades de vídeo.

Cabe ressaltar que a exatidão da estimativa de assincronia de fluxos realizada pelo servidor depende igualmente do período de realimentação. Assim, neste gráfico, pode-se observar que a assincronia não é compensada totalmente após as ações periódicas de resincronização (operações *delete*) realizadas pelo servidor, pouco antes de atingir-se os picos de assincronia.

## **5.5 Conclusão**

Neste capítulo foi apresentada a validação por simulação de problemas de sincronização em sistemas multimídia distribuídos especificados na linguagem Esterel. Em particular foram abordadas várias técnicas da abordagem temporal nos domínios de sincronização intra-fluxo, inter-fluxo e multidestino. Aspectos específicos de cada técnica foram discutidos.

Pode-se afirmar que a geração do código executável dos módulos sincronizadores constitui em si, um objetivo fundamental na utilização de Esterel como linguagem de especificação no desenvolvimento de sistemas multimídia.

## Capítulo 6

### *Considerações Finais*

O trabalho apresentado neste documento mostrou como pode ser utilizada a técnica de descrição formal Esterel nas fases de especificação e validação, no desenvolvimento formal de Sistemas Multimídia Distribuídos. A problemática da sincronização temporal foi abordada particularmente em cada um de seus aspectos: intra-fluxo, inter-fluxo e multidestino.

Alguns dos principais mecanismos de sincronização propostos na literatura foram utilizados na resolução de problemas de sincronização expostos na forma de cenários que representaram diferentes exigências e restrições de sincronização.

Uma metodologia de desenvolvimento baseada em Esterel foi proposta e aplicada com sucesso, abrangendo as fases de especificação, validação e avaliação de desempenho dos sistemas.

A hipótese síncrona na linguagem Esterel proporcionou uma abstração simples e eficiente para a especificação dos aspectos de sincronização multimídia. Assim, os fluxos de dados foram modelados como sinais e eficientemente manuseados pelas construções temporais oferecidas pela linguagem. O tempo foi tratado como um sinal de entrada ao sistema, e manuseado com as mesmas construções temporais dos outros sinais conseguindo-se assim, um tratamento homogêneo dos aspectos temporais da especificação.

A validação dos mecanismos de sincronização especificados foi realizada através da simulação sobre o código executável obtido diretamente da compilação das especificações. Este fato representou uma grande vantagem, permitindo visualizar rapidamente o comportamento das especificações mesmas e facilitando a detecção de erros num estágio inicial de especificação. Adicionalmente, a simulação permitiu a análise de desempenho de certas características dos mecanismos de sincronização especificados.

Entretanto, a principal limitação da simulação é que ela não constitui uma prova formal da correção de uma especificação, pois não é possível garantir que abrange o espaço total de estados do sistema. Alternativamente, o processo de **verificação** utiliza métodos matemáticos rigorosos

para provar que uma especificação satisfaz certas propriedades desejáveis. Ferramentas tais como AUTO [Vergamini92] e TempEst [Jagadeesan95] podem ser utilizadas para a verificação de especificações Esterel, mas encontram dificuldades no caso de especificações que utilizem variáveis ou sinais que carreguem informações, e cujo controle de fluxo dependa de valores gerados em tempo de execução [Jagadeesan95]. Sendo este o caso dos modelos apresentados neste trabalho, a simulação constituiu uma opção razoável.

Uma das características mais importantes das especificações realizadas é o encapsulamento do comportamento tempo-real. Pode-se afirmar que a geração do código executável dos módulos sincronizadores representou uma das principais motivações para a escolha de Esterel, entre outras linguagens (LOTOS, Estelle, etc.), para o desenvolvimento de sistemas multimídia. Estes códigos podem ser utilizados num contexto computacional distribuído como sugerido em [Stefani92].

Sem dúvida, a consideração da implementação final destes mecanismos, pode ser a fase seguinte do trabalho. Assim por exemplo, o sistema operacional a utilizar deve oferecer as primitivas de sincronização eficientes que o *hardware* e *software* de aplicações multimídia precisam para interações multimídia complexas, permitindo o suporte integral de atividades fim-a-fim. Entretanto, os sistemas operacionais mais populares tais como UNIX, mostram-se inadequados para o suporte da maioria de aplicações multimídia [Bulterman91].

Finalmente, este trabalho trouxe um conhecimento mais específico sobre as questões de sincronização multimídia, visando principalmente, adquirir uma experiência orientada à concepção formal de sistemas multimídia.

## Referências Bibliográficas

- [Allen83] J.F. Allen, *Maintaining Knowledge About Temporal Intervals*. Communications ACM, Vol 26, Nov. 1983.
- [Anderson91] D.P. Anderson, G. Homsy, *A Continuous Media I/O Server and Its Synchronization Mechanism*. IEEE Computer, October 1991.
- [Berry88] G. Berry, G. Gonthier, *The ESTEREL Synchronous Programming Language: Design, Semantics and Implementation*. INRIA Report 842, Institute National de Recherche en Informatique et Automatique (INRIA), 1988.
- [Besse93] L. Besse, L. Dairaine, E. Horlait, *Synchronisations Multimedia Distribuées: Problématique, Concepts et Modèle*. Colloque Francophone sur L'Ingenierie des Protocoles. Montreal, Canada, 7-9 Septembre 1993.
- [Blair93] G. Blair, L. Blair, H. Bowman, A. Chetwynd, *Formal Support for the Specification and Construction of Distributed Multimedia Systems (The Tempo Project)*. Final Project Deliverable, December 1993.
- [Blakowski92] G. Blakowski, J. Hubel, U. Langrehr, M.Muhlhauser, *Tool Support for the Synchronization and Presentation of Distributed Multimedia*. Computer Communications, Vol. 15 No. 10. December 1992.
- [Bolognesi87] T. Bolognesi, E. Brinksma, *Introduction to the ISO Specification Language LOTOS*. Computer Network and ISDN Systems, No 23, 1991.
- [Bucci95] G. Bucci, M. Campanai, P. Nesi, *Tools for Specifying Real-Time Systems*. Real Time Systems, Vol. 8, No 2/3 March/May 1995.
- [Bulterman91] D. Bulterman, *Multimedia Synchronization and Unix, or If Multimedia Support is the Problem, is Unix the Solution ?* EurOpen Autumn 1991 Conference.

- [Camargo95] M.S. de Camargo, *Tornando a Linguagem LOTOS Apta para Especificar Sistemas Dependentes do Tempo*. Tese de Doutorado, Universidade Federal de Santa Catarina. Florianópolis, Brasil, Janeiro 1995.
- [Carmo92] L.F.R.C. Carmo; P. de Saqui-Sannes, J.P. Courtiat, *Basic Synchronisation Concepts in Multimedia Systems*. Proceedings of the 3rd Workshop on Network and Operating System Support for Digital Audio and Video, San Diego. Nov. 12-13, 1992.
- [Carmo93] L.F.R.C. Carmo, J.P. Courtiat, *Towards Multimedia Communication Services*. Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems. Lisbonne, 22-24 September 1993.
- [Carmo94a] L.F.R.C. Carmo, J.P. Courtiat, *Implementing Intra-stream Synchronization by means of Conditional Dependency Expressions*. Proceedings of the 5th IFIP Conference on High Performance Networking, France, 1994.
- [Carmo94b] L.F.R.C. Carmo, *Methodes de Synchronisation dans les Systemes Repartis Multimedia: Une approche Integrant Relations de Causalité e Contraintes Temporelles*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, Novembre 1994.
- [Coulson93a] G. Coulson, and G. Blair, *Micro-Kernel Support for Continuous Media in Distributed Systems*. Internal Report No. MPG-93-04. Department of Computing, Lancaster University, Lancaster LA1 4YR, UK., 1993.
- [Coulson93b] G. Coulson, G. Blair, B. Robin, and D. Shepherd, *Extending the Chorus Micro-Kernel to Support Continuous Media Applications*. 4th international Workshop on Network and Operating System Support for Digital Audio and Video. Lancaster UK, November 1993.
- [Coulson93c] G. Coulson, *Multimedia Support in Open Distributed Systems*. PhD Thesis. Lancaster University, April 1993.
- [Courtiat94] J.P. Courtiat, L.F.R.C. Carmo, R.C. de Oliveira, *A New Mechanism for Achieving Inter-stream synchronization in Multimedia Communication*



*Systems*. IEEE International Conference on Multimedia Computing and Systems, Boston. May 1994.

- [Courtia95] J.P. Courtiat, R.C. de Oliveira, L. Andriantsiferana, *Specification and Validation of Multimedia Protocols using RT-LOTOS*.
- [Dairaine92] L. Dairaine, E. Horlait, *Synchronisations dans les Systemes Multimedia Distribués*. MASI 92.91. Institut Blaise Pascal. Paris, France. Novembre 1992.
- [Dairaine93] L. Dairaine, H. Santoso, and E. Horlait, *Deterministic and Statistical Intra-flow Synchronization Services*. MASI 93.49 Institut Blaise Pascal. Paris, France. September 1993
- [Dairaine94] L. Dairaine, *Techniques de Synchronisation dans les Systèmes de Communication Haut-Débit Multimedia*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, France. Septembre, 1994.
- [Diaz94] M. Diaz, P. Senac, *Time Stream Petri Nets: A Model for Multimedia Streams Synchronization*. Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragosa, Spain. June 1994.
- [Dupuy92] S. Dupuy, W. Tawbi, E. Horlait, *Protocols for High-Speed Multimedia Communications Networks*. Computer Communications, Vol. 15 No. 6. July/August 1992.
- [Escobar91] J. Escobar, D. Deutsch, C. Partridge, *A Multi-Service Flow Synchronization Protocol*. Research Report BBN System and Technologie Division, 10 Moulton St. Cambridge, MA 02138 March 14th, 1991.
- [Escobar94] J. Escobar, C. Partridge, D. Deutsch, *Flow Synchronization Protocol*. IEEE/ACM Transactions on Networking, Vol 2, No 2, April 1994.
- [Ehley94] L. Ehley, B. Furth, M. Ilyas, *Evaluation of Multimedia Synchronization Techniques*. Proceedings of the IEEE International Conference on Multimedia Computing and Systems ICMCS'94, USA, 1994.

- [Farines89] J. M. Farines, E. Cantú, H. E. Garnousset, C.A. Maziero, *ARP: Uma Ferramenta para o Desenvolvimento de Software em Aplicações Distribuídas*. Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos. Florianópolis, Brasil. Setembro 1989.
- [Ferrari90] D. Ferrari, *Client Requirements for Real-Time Communication Services*. IEEE Communications Magazine, Vol. 28, No 11, November 1990.
- [Ferrari92] D. Ferrari, *Distributed Delay Jitter Control In Packet-Switching Internetworks*. Computer Communications, Vol 15 No 6, July/August 1992.
- [Furth94] B. Furth, *Multimedia Systems: An Overview*. IEEE Multimedia, Vol. 1, No 1, Spring 1994.
- [Govindan92] R. Govindan, *Operating System Mechanisms for Continuous Media*. Department of Electrical Engineering and Computer Sciences. University of California, Berkeley. Berkeley, CA 94709.
- [Heijenck94] G. Heijenck, X. Hou, I.G. Niemegeers, *Communication Systems Supporting Multimedia Multiuser Applications*. IEEE Network Magazine, January/February 1994.
- [Jagadeesan95] L.J. Jagadeesan, C. Puchol, J.E. von Olnhausen, *Safety Property Verification of Esterel Programs and Applications to Telecommunications Software*. Proceedings of the 7th Conference on Computer-Aided Verification, July 1995.
- [Jeffay92] K. Jeffay, D.L. Stone, F.D. Smith, *Kernel Support for Live Digital Audio and Video*. Computer Communications, Vol. 15 No. 6, July/August 1992.
- [Karmouch93] A. Karmouch, *Multimedia Communication Systems*. Simposio Brasileiro de Redes de Computadores, SBRC, May 1993.
- [Koegel94] J. Koegel, *Multimedia Systems*. ACM Press, 1994.
- [Leung90] W. Leung, et al., *A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching Networks*. IEEE Journal on Selected Areas in Communications. Vol. 8 No. 3, April 1990.

- [Little90a], T.D.C. Little, A. Ghafoor, *Synchronization and Storage Models for Multimedia Objects*. IEEE Journal on Selected Areas in Communications. Vol. 8 No. 3, April 1990.
- [Little90b] T.D.C. Little, A. Ghafoor, *Network Considerations for Distributed Multimedia Object Composition and Communication*. IEEE Network Magazine, November 1990.
- [Little91] T.D.C. Little, A. Ghafoor, *Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks*. IEEE Computer, October 1991.
- [Little92] T.D.C. Little, A. Ghafoor, *Scheduling of Bandwidth-Constrained Multimedia Traffic*. Computer Communications, Vol. 15 No. 6, July/August 1992.
- [Meyer93] T. Meyer, W. Effelsberg, R. Steinmetz, *A Taxonomy on Multimedia Synchronization*. Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems. Lisboa, 22-24 September 1993.
- [Nicolaou90] C. Nicolaou, *An Architecture for Real-Time Multimedia Communication Systems*. IEEE Journal on Selected Areas in Communications. Vol. 8 No. 3, April 1990.
- [Partridge91] C. Partridge, *Isochronous Applications Do Not Require Jitter-Controlled Networks*. Research Report No 1257. Swedish Institute of Computer Science, September 1991.
- [Ramanathan93] S. Ramanathan, P.V. Rangan, *Feedback Techniques for Intra-media Continuity and Inter-media Synchronization in Distributed Multimedia Systems*. Computer Journal, Vol. 36, No. 1, 1993.
- [Santoso93] H. Santoso, L. Dairaine, S. Fdida, E. Horlait, *Preserving Temporal Signature: a Way to Convey Time Constrained Flow*. MASI 93.48 Institut Blaise Pascal. Paris, France. September 1993.

- [Saparamadu93] V. Saparamadu, A. Seneviratne, M. Fry, *A Review of Inter Media Synchronization Schemes*. Proceedings of the First International Conference on Multi-Media Modeling Vol.1. Singapore 9-12, November 1993.
- [Shepherd90] D. Shepherd, M. Salmony, *Extending OSI to Support Synchronization Required by Multimedia Applications*. Computer Communications, Vol. 13 No 7, September 1990.
- [Shepherd92] D. Shepherd, D. Hutchinson, F. Garcia, G. Coulson, *Protocol Support for Distributed Multimedia Applications*. Computer Communications, Vol. 15 No. 6, July/August 1992.
- [Schuermann90] G. Schuermann, U. Holzmann-Kaiser, *Distributed Multimedia Information Handling and Processing*. IEEE Network Magazine, November 1990.
- [Stefani92] J. Stefani, L. Hazard, F. Horn, *Computacional Model for Distributed Multimedia Applications Based On a Synchronous Programming Language*. Computer Communications, Vol. 15, No 2, March 1992.
- [Steinmetz90] R. Steinmetz, *Synchronizaton Properties in Multimedia Systems*. IEEE Journal on Selected Areas in Communications, Vol. 8, No 3, April 1990.
- [Steinmetz92] R. Steinmetz, J.C. Fritzsche, *Abstractions for Continuous-Media Programming*. Computer Communications, Vol. 15 No. 6. July/August 1992.
- [Tawbi92] W. Tawbi, L. Fedoui, E. Horlait, *Management of Multimedia Applications QoS on ATM Networks*. MASI 92.92 Institut Blaise Pascal. Paris, France. Novembre 1992.
- [Walter83] B. Walter, *Timed Petri Nets for Modelling and Analysing Protocols with Time*. Proceedings of the IFIP Conference On Protocols Specification, Testing and Verification, III, 1983, North Holland, H. Rudin & C. Est eds.
- [Williams94] N. Williams, G. Blair, *Distributed Multimedia Applications: A Review*. Computer Communications, Vol. 17 No. 2. February 1994.
- [Yavatkar92] R. Yavatkar, *MCP: A Protocol for Coordination and Temporal Synchronization in Multimedia Collaborative Applications*. Proceedings of

the 12th International Conference on Distributed Computing Systems.  
Yokohana, Japan. June 9-12, 1992.

# Apêndice A

## *Elementos de Base da Linguagem Esterel*

Este apêndice apresenta os elementos de bases da linguagem Esterel necessários para entender o conjunto de especificações realizadas neste trabalho. Uma descrição completa da linguagem pode ser encontrada em [Berry88].

Um programa Esterel pode ser visto como uma conjunto de processos paralelos que se comunicam instantaneamente entre si, e com o ambiente, através da difusão de sinais. Este é o único mecanismo de comunicação suportado por Esterel: os sinais locais correspondem à comunicação interna entre processos, enquanto os sinais de entrada e saída fornecem a comunicação com o ambiente. Em Esterel, o tempo aparece como um sinal de entrada gerado por um relógio externo.

Um sinal é difundido através da instrução **emit** (p. ex. **emit SIN (exp)** define a difusão do sinal SIN com o valor exp). Os sinais carregam dois tipo de informação:

1. Uma informação de controle que define se um sinal está presente ou não durante um instante de execução.
2. Um valor que é persistente e pode ser consultado como qualquer outra variável ou constante.

O valor corrente de um sinal pode ser consultado através do operador ?

As reações as sinais baseiam-se numa única primitiva básica:

**do stat watching exp** SIN

Onde *stat* corresponde a um comando, *exp* é uma expressão inteira opcional, e SIG é um sinal. O comando *stat* começa imediatamente; se *stat* termina antes da seguinte exp-ésima ocorrência do sinal SIN, o comando **do watching** também termina. Em outro caso, o comando **do watching** é

terminado no instante da  $exp$ -ésima ocorrência do sinal SIN, sendo que o comando *stat* é imediatamente abortado (*killed*).

Alguns dos comandos imperativos da linguagem são os seguintes:

- Atribuição: *X:= expressão*.
- Instrução condicional: **if *exp* then *stat1* else *stat2* end.**
- Ciclo infinito: **loop *stat* end.**
- A instrução: **halt.** A única instrução que consome tempo.
- Operador de seqüência: *stat1; stat2*. Definindo o começo do comando *stat2* imediatamente após o término do comando *stat1*.
- Operador paralelo: *stat1 || stat2*. Definindo o começo instantâneo de ambos comandos sendo que a instrução só termina quando ambos comandos terminarem.

A unidade de programação em Esterel é o módulo. Esterel possui o comando **copymodule**, que permite a expansão de um módulo e oferece facilidades de substituição de sinais e constantes. A sintaxe é a seguinte:

**copymodule *ident\_mod* [*lista de substituição*]**

Onde *ident\_mod* é o identificador do módulo, e *lista de substituição* corresponde a uma lista opcional de substituição de sinais e/ou constantes. O módulo é copiado textualmente com as substituições indicadas.

# Apêndice B

## Módulo de Sincronização : Sinc\_Intra

Este apêndice inclui o módulo Sinc\_Intra, que forma parte da especificação do cenário de sincronização multideestino da seção 4.4.2.

```
module Sinc_Intra :
constant
  D_MIN_M,D_MAX_M,
  D_MIN_F, D_MAX_F,
  TETA_MIN, TETA_MAX,
  MIDIA, SIZE_BUFF, N_FEEDBACK : integer,
  TRUE, FALSE : boolean;
type
  elt_type;
input
  out_f_ch(integer), ms, t(integer);
output
  in_m_ch(elt_type),
  Uf(integer), % Unidade de dado recebido em feedback
  a_Uf(integer); % Tempo de chegada de Uf ao server
function
  get_media_elt(integer) : elt_type,
  n_elt(elt_type) : integer,
  tobemarked(elt_type,integer) : boolean,
  next_time(integer,integer,integer,integer) : integer,
  earliest_next_time(integer,integer,integer,integer,integer,integer,integer) : integer,
  latest_next_time(integer,integer,integer,integer,integer,integer) : integer,
  scheduler(integer,integer) : integer;
procedure
  write_mark (elt_type) ();
signal
  teta(integer), feedback_unit(boolean) in % Periodo de envio e sinal de ajuste do periodo

  emit feedback_unit(FALSE);
  await ms;
[
  var elt : elt_type, earliest, latest : integer in
  loop
    elt := get_media_elt(MIDIA);
    if (tobemarked(elt.N_FEEDBACK)) then
      call write_mark (elt) ()
    end;
    emit in_m_ch (elt);
    if (not(?feedback_unit)) then
      emit teta(next_time(n_elt(elt)+1, TETA_MIN, D_MAX_M, D_MIN_M) - ?t)
    else
      earliest :=
      earliest_next_time(?a_Uf.D_MIN_F.n_elt(elt)+1, SIZE_BUFF, ?Uf, TETA_MAX, D_MIN_M)-?t;
      latest :=
```



```
        latest_next_time(?a_Uf, D_MAX_F, n_elt(elt)+1, ?Uf, TETA_MIN, D_MAX_M) - ?t;
        emit teta(scheduler(earliest,latest));
    end;
    await ?teta ms
end loop
end var
||
await out_f_ch;
emit feedback_unit(TRUE);
loop
    emit a_Uf(?t);
    emit Uf(?out_f_ch);
    await out_f_ch
end
]
end signal
end module
```

# Apêndice C

## Módulos Observadores

Este apêndice inclui os módulos observadores que foram utilizados para a simulação e verificação de especificações da seção 5.3.

### C.1 Módulo Observador dos Cenários Intra-Fluxo 1, 2 e 3.

```
module Observador_1 :
constant
  TETA,
  SIZE_BUFF,
  MIN_LATEN,
  MAX_LATEN : integer;
type
  elt_type;
input
  out_src(elt_type), in_sync(elt_type), play(elt_type), ms;
output
  error(integer);
function
  info_elt(elt_type) : integer;
                                     % confiere a latência
signal t(integer) in
  var t_first, latencia, n : integer in
    await immediate out_src;
    t_first := ?t;
    await play;
    n := info_elt(?play);
    latencia := ?t - t_first;
    if (latencia < MIN_LATEN or latencia > MAX_LATEN) then
      emit error(0)
    end
  loop
                                     % confiere a preservação da assinatura temporal
  do
    await play;
    n := n+1;
    if (info_elt(?play) <> n) then emit error(1) end
    watching (TETA + 1) ms
    timeout emit error(2) end
  end loop
end var
||
var cur := 0 : integer in
  every ms do cur := cur + 1; emit t(cur) end
end
end signal
||
                                     % confiere os limites do buffer
signal in(integer), out(integer), update in
```

```

emit out(0);
[
var n_in := 0 : integer in
  every in_sync do n_in := n_in+1; emit update; emit in(n_in) end
end
||
var n_out := 0 : integer in
  every play do n_out := n_out+1; emit update; emit out(n_out) end
end
||
var b : integer in
  every update do
    b := ?in-?out;
    if (b < 0 or b > SIZE_BUFF) then emit error(3) end
  end
end
]
end signal
end module

```

## C.2 Módulo Observador do Quarto Cenário Intra-Fluxo

```

module Observador_2 :
constant
  TETA,          % teta_out
  SIZE_BUFF,
  MIN_LATEN,
  MAX_LATEN : integer;
type
  elt_type;
input
  out_src(elt_type), in_sync(elt_type), play(elt_type), ms;
output
  error(integer);
signal t(integer) in
  var t_first, latencia : integer in  %confiere a latência
    await immediate out_src;
    t_first := ?t;
    await play;
    latencia := ?t - t_first;
    if (latencia < MIN_LATEN or latencia > MAX_LATEN) then
      emit error(0)
    end
    loop                                % confiere a correta alteração da assinatura temporal
      do
        await play;
        watching (TETA + 1) ms
        timeout emit error(2) end
      end loop
    end var
  ||
  var cur := 0 : integer in
    every ms do cur := cur + 1; emit t(cur) end
  end

```

```

end signal
||                                     % confiere os limites do buffer
signal in(integer), out(integer), update in
  emit out(0);
  [
  var n_in := 0 : integer in
    every in_sync do n_in := n_in+1; emit update; emit in(n_in) end
  end
  ||
  var n_out := 0 : integer in
    every play do n_out := n_out+1; emit update; emit out(n_out) end
  end
  ||
  var b : integer in
    every update do
      b := ?in-?out;
      if (b < 0 or b > SIZE_BUFF) then emit error(3) end
    end
  end
  ]
end signal
end module

```

### C.3 Módulo Observador do Quinto Cenário Intra-Fluxo

```

module Observador_3 :
constant
  SIZE_BUFF,
  MIN_LATEN,
  MAX_LATEN : integer;
type
  elt_type;
input
  out_src(elt_type), in_sync(elt_type), play(elt_type), ms;
output
  error(integer);

signal t(integer) in
  var t_first, latencia : integer in      % confiere a latência
    await immediate out_src;
    t_first := ?t;
    await play;
    latencia := ?t - t_first;
    if (latencia < MIN_LATEN or latencia > MAX_LATEN) then
      emit error(0)
    end
  end var
  ||
  var cur := 0 : integer in
    every ms do cur := cur + 1; emit t(cur) end
  end
end signal
||                                     % confiere os limites do buffer
signal in(integer), out(integer), update in
  emit out(0);

```

```

[
var n_in := 0 : integer in
  every in_sync do n_in := n_in+1; emit update; emit in(n_in) end
end
||
var n_out := 0 : integer in
  every play do n_out := n_out+1; emit update; emit out(n_out) end
end
||
var b : integer in
  every update do
    b := ?in-?out;
    if (b < 0 or b > SIZE_BUFF) then emit error(3) end
  end
end
]
end signal
end module

```

#### C.4 Módulo Observador do Cenário Multidestino

% Chamada: copymodule Observa\_Skew [constant X/MIN\_SKEW, Y/MAX\_SKEW]

```

module Observa_Skew :
constant
  TETA_NOMINAL_MASTER,
  TETA_NOMINAL_SLAVE,
  MIN_SKEW, MAX_SKEW % Skew em relação ao master. P.ex. skew > 0 => master adiantado
  : integer; % Lipsync [Stefani92] (audio master) MIN_SKEW = -15 ms, MAX_SKEW = 150 ms
input
  play_master(integer), play_slave(integer), ms;
output
  error(integer);
function
  calc_skew(integer,integer,integer,integer,integer,integer) : integer;

signal t_slave(integer), t(integer) in
  loop
    await play_slave;
    emit t_slave(?t)
  end loop
  ||
  var skew : integer in
    [await play_master || await play_slave];
    present play_master then
      skew := calc_skew(?play_master. ?t. TETA_NOMINAL_MASTER, ?play_slave. ?t_slave.
TETA_NOMINAL_SLAVE);
      if (skew < MIN_SKEW or skew > MAX_SKEW) then emit error(skew) end
    end;
    loop
      await play_master;
      skew := calc_skew(?play_master. ?t. TETA_NOMINAL_MASTER, ?play_slave. ?t_slave.
TETA_NOMINAL_SLAVE);
      if (skew < MIN_SKEW or skew > MAX_SKEW) then emit error(skew) end
    end loop
  end loop
end module

```

```
end var
||
var cur := 0 : integer in
  every ms do cur := cur + 1; emit t(cur) end
end
end signal
end module
```