Universidade Federal de Pernambuco
Centro de Informática

Graduação em Engenharia da Computação

RINA - Uma Proposta para a Internet do Futuro

André Felipe Pereira de Melo

Trabalho de Graduação

Recife
3 de Outubro de 2013

Universidade Federal de Pernambuco
Centro de Informática

André Felipe Pereira de Melo

RINA - Uma Proposta para a Internet do Futuro

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Prof. José Augusto Suruagy Monteiro*

Recife
3 de Outubro de 2013

À minha amada mãe, Joselma.

Agradecimentos

A Deus, por todo cuidado e amor que tem por nós.

À minha mãe, pela total dedicação à educação dos seus filhos.

Ao meu pai, Antônio, pelo exemplo deixado.

Aos meus irmãos, Gabriella e Victor, pelos exemplos e apoio.

Ao meu cunhado Lyncoln, também por seu apoio e exemplo.

À minha namorada Thais, por estar sempre ao meu lado.

Aos meus tios Joadete e Adilson e às minhas primas Larissa e Tâmara, pelo apoio e companheirismo constantes em todas as fases da minha vida.

Aos meus amigos e colegas de curso por todo o conhecimento e aprendizagem em grupo.

Ao professor José Augusto Suruagy Monteiro, por toda a orientação durante a realização deste trabalho.

E a todos os que fazem parte da BankSystem Software Builder por toda compreensão e apoio.

Resumo

A Internet atual enfrenta problemas para suprir as demandas de serviços existentes como mobilidade, *streaming* de vídeo e *multihoming*. Para resolver esses problemas, pesquisadores estão se dedicando a elaborar novas arquiteturas de rede. Uma dessas arquiteturas, RINA, *Recursive InterNetwork Architecture*, propõe uma internet baseada em comunicação entre processos.

Este trabalho estudou essa nova proposta e utilizou um dos protótipos desenvolvidos para experimentar na prática os princípios do RINA.

A partir dessa experimentação, sentiu-se a necessidade de desenvolver uma ferramenta para auxílio na compreensão dos protocolos propostos.

Uma ferramenta foi desenvolvida e novas atividades foram sugeridas para auxiliar na compreensão desta proposta para a Internet do futuro.

Palavras-chave: RINA, arquitetura de redes, Internet do futuro, protótipo.

Abstract

The current Internet is facing problems to meet the demands of existing services such as mobility, video streaming and multihoming. Then, to solve these problems, researchers are devoting to develop new network architectures. One of these architectures, RINA, *Recursive InterNetwork Architecture*, proposes an internet based on inter-process communication.

This project has studied this new proposal and used one of the prototypes developed testing the principles of RINA in practice.

As a result, it was realised the need to develop a tool to aid in the understanding of the proposed protocols.

Therefore, a tool was developed and new activities have been suggested to support the understanding of this proposal for the future Internet.

Keywords: RINA, network architectures, future Internet, prototype.

Sumário

Lista de Figuras

CAPÍTULO 1

Introdução

"Imagine tentar utilizar um computador moderno somente com endereços físicos de memória, essa é a Internet hoje" [Day10a]. É com esta frase que Jonh Day sintetiza a realidade da Internet nos dias atuais. Apesar de parecer chocante à primeira vista, quando se observa com cuidado, vemos que trabalhamos hoje com uma arquitetura de redes desenvolvida nos anos 70 [Jai06] para atender requisitos especificados para suprir as demandas da época. Este modelo atual não tem respondido de maneira efetiva às novas exigências de serviços como segurança, redes sem fio e mobilidade [DMM08].

A primeira rede sem conexão, proposta pelo pesquisador francês Louis Pouzin em 1972, a CYCLADES, serviu como base para a arquitetura que temos hoje. Os desenvolvedores da ARPANET adotaram a ideia e montaram a arquitetura TCP/IP, e, a partir daí, se desenvolveu a Internet como a conhecemos [Nol11].

O modelo TCP/IP foi projetado como uma arquitetura em camadas, onde cada camada possui funções distintas e isoladas [MFCD09]. Porém, a falta de uma visão estruturada de como suprir a demanda atual de serviços tem levado à criação de soluções *ad-hoc* que provocam a violação da independência entre camadas, fazendo com que elementos como roteadores, por exemplo, inspecionem detalhadamente os pacotes em trânsito para efetuar operações que seriam da camada de aplicação ou transporte [DMM08].

A persistência em manter a estrutura atual da rede mundial tem trazido dificuldades para que serviços essenciais nos dias de hoje sejam supridos. Vamos expor a seguir alguns desses problemas.

1.1 Problemas na arquitetura atual

Nesta seção são apresentados os diversos problemas encontrados na arquitetura atual da Internet de acordo com [GL06].

1.1.1 IP

É comum encontrar definições do endereço IP como o identificador de um dispositivo em uma rede. Apesar de fazer sentido, esta afirmação não é absolutamente correta, uma vez que este endereço identifica a interface que conecta o dispositivo ao enlace físico da rede e não o dispositivo em si [KR09]. Quando um sistema muda o seu ponto de conexão, o seu endereço também muda, dificultando assim o alcance a sistemas móveis. Além disto, a natureza sem conexão do protocolo IP complica a garantia de QoS na comunicação entre sistemas.

1.1.2 Modelo de Melhor Esforço

O fato de a Internet ter sido originalmente projetada para transmissão pura de dados, como transporte de arquivos e acesso remoto, fez surgir o Melhor Esforço como um princípio fundamental da arquitetura atual. Com o surgimento de demandas como VoIP, vídeo conferência e *streaming* em tempo real, a Internet tem o desafio de prover vários tipos de serviços de comunicação, especialmente serviços de áudio e vídeo, baseando-se em uma arquitetura originalmente projetada para suportar comunicações de dados simples.

1.1.3 Redundância nas camadas

A arquitetura que temos hoje foi projetada como uma arquitetura em camadas, onde cada uma possui funções específicas. Com a evolução da demanda de serviços, funções similares de correção de erros e controle de tráfego, por exemplo, precisaram ser implementadas em camadas diferentes, ocasionando uma quebra da divisão de camadas proposta originalmente. Isto mostra que a proposta inicial não é suficiente para suportar as necessidades atuais.

1.1.4 Segurança

Criptografia na camada de enlace de dados foi a única consideração sobre segurança feita na proposta da arquitetura original. Os projetistas originalmente não se preocuparam em desenvolver uma arquitetura completa realmente segura.

1.1.5 Complexidade

A arquitetura da Internet atual, juntamente com a implementação de seus hardwares e softwares, tem se tornado cada vez mais complexa. As adaptações feitas na arquitetura para a adequação ao ambiente de aplicações têm custado a adição de mais e mais complexidade à rede. Quanto mais a rede cresce, o aumento da complexidade teórica e prática indica os problemas que não foram pensados no projeto original, uma rede de uma escala bem menor.

Os problemas encontrados incentivaram a busca de novas propostas de arquiteturas em uma tentativa de suprir a demanda de serviços do cenário atual de aplicações de maneira direta e com menor complexidade. Vários estudos nesse sentido foram realizados e vamos apresentar a seguir algumas arquiteturas resultantes desse esforço.

1.2 Algumas alternativas propostas

Esta seção apresenta algumas alternativas de arquitetura propostas para construir uma rede mundial de computadores mais adequada às necessidades atuais.

1.2.1 RBA - *Role Based Network*

A arquitetura em camadas da Internet não tem suportado de maneira adequada a demanda atual de serviços, forçando a constante violação de camadas e a criação de várias subcamadas para

atender às novas necessidades sem alterar as interfaces já existentes [MFCD09].

Duas sugestões para a resolução do problema enfrentado pela Internet resultaram nos conceitos da arquitetura baseada em papéis: a sugestão de trocar o paradigma tradicional de divisão em camadas de protocolos por um modelo mais geral e a de prover um mecanismo de protocolo para adicionar metadados aos dados dos pacotes [BFH02].

Com o objetivo de reduzir a complexidade e aumentar a flexibilidade da arquitetura [BFH02] foi proposta a RBA, uma arquitetura que propõe uma rede sem camadas, formada por módulos dispostos de maneira não-hierárquica. Estes módulos são chamados de papéis e proveem uma interconexão mais rica do que a fornecida pela arquitetura atual [MFCD09].

A proposta de uma rede sem camadas torna difícil adquirir modularidade, o que é praticamente inerente à arquitetura em camadas. Para dar essa característica ao RBA, os cabeçalhos dos pacotes são tratados como um *heap*, onde cada cabeçalho tem tamanho variável e pode ser acessado em qualquer ordem [BFH02]. Esses cabeçalhos são chamados de cabeçalhos específicos de papel ou RSH, do inglês *Role-Specific Header*. Eles sinalizam a função do pacote de dados e servem para organizar seus metadados.

Este modelo é mais flexível que o modelo de camadas, fazendo com que as atuais violações de camada sejam substituídas por interação entre papéis.

1.2.2 Horizon

Horizon é uma proposta de arquitetura inteligente adaptada para redes virtuais onde um sistema automático de mecanismos inteligentes compreende o contexto e adapta o protocolo à situação [MFCD09]. A técnica de virtualização consiste em dividir um roteador em n fatias de roteadores virtuais isolados, criando n redes virtuais [dTeA09] onde cada rede é selecionada de acordo com o serviço a ser oferecido.

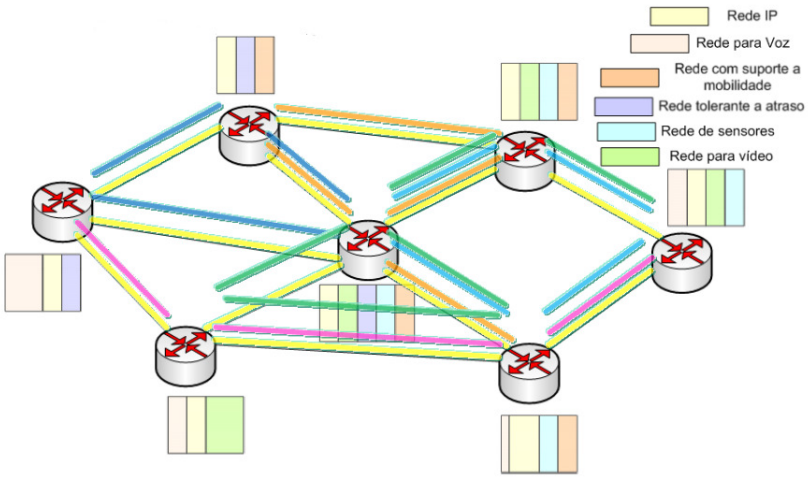

Figura 1.1 Exemplo de rede Horizon
Fonte: [MFCD09]

A Figura 1.1 ilustra um cenário de rede com a arquitetura Horizon onde roteadores são

virtualizados criando suas redes virtuais.

O conjunto de mecanismos inteligentes responsável pela escolha da rede faz parte do plano de pilotagem. O plano de pilotagem é responsável pela escolha dos algoritmos locais em função dos parâmetros de qualidade de serviço, segurança, energia e mobilidade, pelo gerenciamento de recursos, adaptação a situações imprevistas e recuperação de falhas e ataques. Um processo de aprendizagem existe neste modelo para tornar o sistema capaz de pilotar a rede em tempo real [MFCD09].

Cada rede virtual pode ser instanciada e removida dependendo de suas necessidades. O plano de pilotagem se responsabiliza por distribuir os recursos determinando qual rede virtual será utilizada pelo usuário. Isto fornece as ferramentas necessárias para que os prestadores de serviços executem serviços fim-a-fim com diferentes parâmetros de qualidade e serviço [MFCD09].

1.2.3 KP - *Knowledge Plane*

A Internet transporta dados sem saber exatamente o significado de seu conteúdo e o seu propósito. Quando algum erro ocorre, os sistemas nas extremidades percebem, porém o núcleo da rede não pode informar que algo está errado pois não tem ideia do comportamento esperado [CPRW03].

Para solucionar este problema, esta proposta sugere a construção de uma rede que tenha uma visão global do seu propósito, sendo capaz de se auto-organizar, tanto a partir do recebimento de um conjunto de instruções quanto à medida que os requisitos mudam, e descobrir automaticamente a ocorrência de erros e consertá-los ou explicar ao usuário se não for capaz [CPRW03].

Este objetivo é atingido através da criação de um sistema dentro da rede, chamado de Plano de Conhecimento, que constrói e mantém modelos do que a rede precisa fazer com o objetivo de prover serviços e instruções a outros elementos na rede [CPRW03]. Esta estrutura deve ser inteligente e distribuída, buscando, agregando e gerenciando informações sobre o comportamento e operação da rede [MFCD09].

O KP também visa simplificar o gerenciamento do roteamento da rede que, na arquitetura atual, é feito de forma manual com base em políticas, o que torna a rede susceptível a falhas, resistente a mudanças e de alta complexidade [MFCD09].

O sistema distribuído proposto tem suas partes sendo executadas em *hosts* e servidores na rede, coletando observações, restrições e afirmações e, a partir da aplicação de regras a esses dados, gerando observações e respostas [CPRW03].

O maior objetivo é construir uma nova geração de rede que possa guiar o seu próprio desenvolvimento e configuração, diagnosticar os seus problemas e tomar decisões para resolvê-los.

1.2.4 RINA - *Recursive InterNetwork Architecture*

RINA é uma proposta de arquitetura que tem como base o conceito de comunicação entre processos. Robert Metcalfe em 1972 generalizou o modelo de comunicação entre processos e definiu redes como um provedor dos meios pelos quais processos em computadores diferentes se comunicam. A partir deste conceito, John Day definiu redes como comunicação entre pro-

cessos e somente comunicação entre processos [DMM08] e desenvolveu o RINA baseando-se nessa premissa.

A principal característica desta alternativa é a recursividade. Diferentemente da arquitetura atual que divide a estrutura da rede em várias camadas sobrepostas possuindo diferentes funções, esta proposta defende o uso de uma única camada que provê comunicação entre processos, ou IPC, do inglês *Inter-Process Communication*, de maneira distribuída e se repete provendo as mesmas funções e mecanismos em escopos diferentes [IRA13a].

Essa estrutura que se repete, chamada de DIF, do inglês *Distributed IPC Facility*, é escalável, ou seja, podem existir tantas DIF's quantas forem necessárias à rede [DMM08]. Essa característica evita alguns dos problemas atuais, como o crescimento das tabelas de roteamento, por exemplo, e fornece suporte a *multihoming* e mobilidade com pouco ou nenhum custo [DMM08].

Uma maior segurança também se configura como uma característica importante desta alternativa. Nela, o escopo dos nomes e das permissões de acesso de cada aplicação se restringe apenas à DIF à qual a aplicação está registrada. Dessa maneira, somente o nome da aplicação de destino de uma comunicação é conhecido e não seu endereço. Por este escopo restrito e também por não haver portas bem conhecidas, uma segurança maior é obtida no processo de comunicação em rede [DMM08].

Outra característica é a existência de apenas um protocolo de comunicação, o CDAP, do inglês, *Common Distributed Application Protocol*, ao invés de vários como existem atualmente: HTTP, FTP, SSH, entre outros. Isto fornece às aplicações uma maior simplicidade e homogeneidade na troca de informações.

Um ponto relevante é que RINA não propõe uma transição brusca da arquitetura atual para a nova [Day11]. O que é proposto é uma adoção gradativa fazendo uso da vantagem que a DIF proporciona de poder operar acima, abaixo ou ao lado da Internet [Day11].

1.2.5 Alternativa escolhida para estudo

A primeira alternativa apresentada, a RBA, propõe a substituição do modelo atual em camadas por um modelo em papéis. Apesar de ser interessante por simplificar a estrutura da rede, este modelo não possui a modularidade como uma característica inerente, precisando criar cabeçalhos especiais para obter uma estrutura modular. Além disso, para esse modelo ser adotado toda a arquitetura atual deve ser substituída, o que é impraticável em termos estruturais e econômicos.

A segunda alternativa, a Horizon, permite o aproveitamento da estrutura e até a convivência com a Internet atual, porém o sistema automático de compreensão de contexto e adaptação de protocolo de acordo com a situação não é implementado de maneira simples e direta, sendo necessária muita experimentação para obter um bom desempenho no cenário real.

Da mesma maneira, a terceira alternativa, a KP, não propõe uma abordagem direta e simples para resolver os problemas atuais. Nela, um sistema distribuído inteligente que torne a rede capaz de se auto-organizar precisa ser implementado e nenhuma garantia de diminuição na complexidade da rede é fornecida.

Dentre as alternativas apresentadas, a última se destaca por sua simplicidade e caráter inovador. Ao contrário das demais, RINA não propõe a utilização de um novo modelo de separação

de papéis na rede ou de sistemas inteligentes, ela propõe um retorno aos princípios básicos para a comunicação em redes e resume a arquitetura de uma internet como uma única camada que se repete fornecendo serviços de comunicação entre processos.

Por essa abordagem direta e promissora, essa última alternativa foi escolhida para o estudo apresentado neste trabalho.

1.3 Objetivos do Trabalho

Nesta seção serão apresentados os objetivos deste projeto.

1.3.1 Objetivo Geral

Este trabalho tem como objetivo geral estudar uma nova proposta de arquitetura para a Internet, a RINA, experimentá-la na prática e contribuir para a sua compreensão.

1.3.2 Objetivos Específicos

Através da instalação e execução do protótipo JAVA da arquitetura RINA, desenvolvido pela organização i2CAT, este trabalho tem como objetivo observar o comportamento e funcionamento, na prática, das principais características da arquitetura.

É também um objetivo deste trabalho a adaptação de um analisador de protocolos para entender o protocolo de aplicação CDAP de modo a solidificar o conhecimento do processo de estabelecimento de conexão e troca de mensagens da arquitetura.

1.4 Roteiro

No Capítulo 2, a arquitetura RINA será apresentada em mais detalhes, sendo exposto o raciocínio, passo a passo, que gerou a proposta e as funcionalidades dos seus componentes.

No Capítulo 3, o RINA na Prática, será apresentado o protótipo JAVA, desenvolvido pela organização i2CAT, e outras prototipagens efetuadas com o objetivo de amadurecer os conceitos da arquitetura e auxiliar na sua aprendizagem.

Logo em seguida, no Capítulo 4, será exposta a adaptação feita em um analisador de protocolos com o objetivo de auxiliar a análise das mensagens trocadas pelo protótipo JAVA.

Finalmente, no último capítulo, serão apresentadas conclusões e trabalhos futuros serão sugeridos.

CAPÍTULO 2

RINA

Repensar os princípios básicos da comunicação em redes ao invés de adaptar a estrutura atual da Internet para suprir as novas demandas de serviços foi a principal motivação que levou à proposta RINA.

Neste capítulo, vamos apresentar a evolução do raciocínio que deu origem à arquitetura e seus principais módulos.

2.1 Retorno aos Fundamentos

Diante do cenário atual de problemas enfrentados na arquitetura da Internet que utilizamos apresentados sucintamente no capítulo anterior, o pesquisador John Day propôs uma reflexão sobre os princípios básicos de redes visando obter uma maior compreensão sobre as dificuldades presentes.

A partir do reconhecimento desses fundamentos, surgiu a proposta de uma nova arquitetura. Uma arquitetura que propõe estabelecer uma estrutura de comunicação em rede mais adequada à demanda de serviços requeridos pela rede mundial de computadores.

O raciocínio seguido por John Day, apresentado em seu livro [Day08] e sintetizado em sua apresentação [Day10a], tem início na análise da comunicação entre dois processos dentro de um mesmo sistema, conforme ilustrado na Figura 2.1. A partir desta análise, pode-se observar o que é requerido para que esta comunicação seja efetuada e, a partir deste ponto, começar a evoluir o cenário tendo em mente o objetivo de atingir o cenário de comunicação de uma rede de redes, uma internet.

Na Figura 2.1 temos dois processos, A e B, dentro de um mesmo sistema, trocando dados entre si. Para que a comunicação seja iniciada, o processo A envia para o módulo do sistema operacional que provê IPC, *IPC Facility*, uma requisição de alocação de um canal de comunicação com o processo B com algumas características definidas. Ao receber esta requisição, o sistema operacional sabe a sua origem e, utilizando suas regras de busca, encontra o processo de destino. Tendo encontrado, checa se A tem acesso a B e, caso tenha, estabelece o fluxo.

Deve-se notar que neste cenário, por ambas as aplicações estarem no mesmo sistema, temos algumas facilidades. O sistema operacional sabe que se B não for encontrado não há necessidade de procurá-lo em outro lugar e, então, ele pode negar a requisição. Além disto, os nomes das aplicações são únicos e o canal de comunicação é confiável, logo não é preciso que haja um sistema complexo de localização de processos por nome, nem uma garantia de entrega de dados.

Evoluindo o cenário para a comunicação entre dois processos em sistemas diferentes, con-

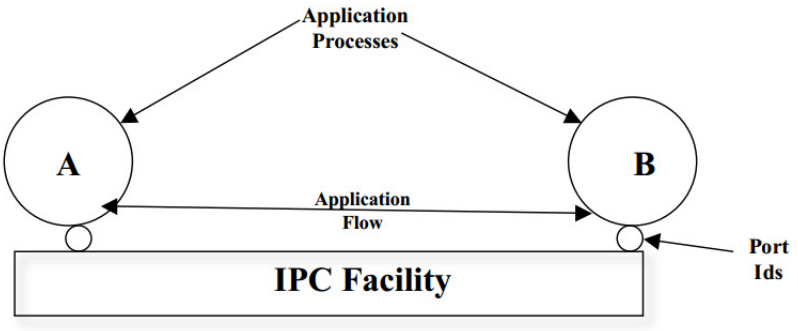

Figura 2.1 Duas aplicações se comunicando no mesmo sistema
Fonte: [Day10a]

forme ilustrado na Figura 2.2, veremos que será preciso adicionar algumas funcionalidades para que a comunicação seja estabelecida.

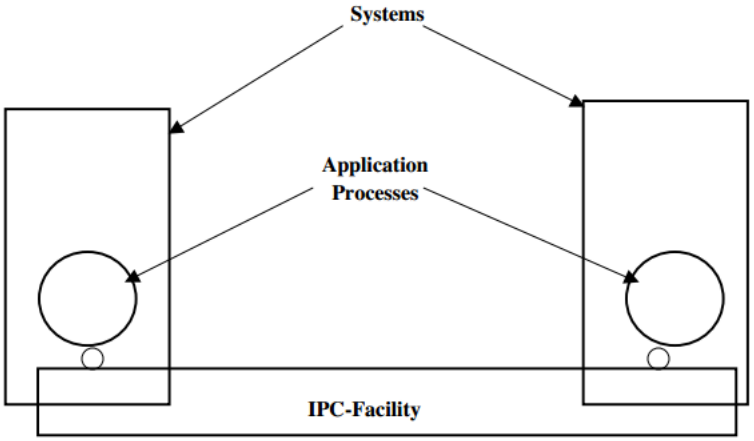

Figura 2.2 Duas aplicações se comunicando em dois sistemas diferentes
Fonte: [Day10a]

O primeiro ponto a se observar nesse caso é que o IPC não é mais um serviço disponibilizado por um único sistema operacional. Temos agora uma aplicação distribuída que fornece os mecanismos de IPC entre os dois sistemas, uma DIF *(Distributed IPC Facility)*.

A primeira necessidade que surge para o estabelecimento da comunicação é a existência de um gerenciamento para encontrar a aplicação de destino. Como não se tem mais um único sistema, devemos ter um meio para definir se o destino está local ou remoto e suas regras de controle de acesso devem estar visíveis a ambos os sistemas.

Como temos uma comunicação entre sistemas diferentes, erros de transmissão são mais prováveis de ocorrer e é necessário que haja uma maneira de controlar tais erros. Sendo assim, uma segunda necessidade é constatada: a presença de algum tipo de protocolo para controlar a transferência de dados.

Devido às necessidades citadas acima, criou-se um protocolo para controlar a busca e o acesso entre os sistemas, o IAP, do inglês *IPC Access Protocol*, e outro protocolo para o controle da transmissão de dados, o EFCP, do inglês *Error and Flow Control Protocol*.

O próximo passo agora é analisar o cenário onde múltiplas aplicações se comunicam simultaneamente entre dois sistemas, conforme ilustrado na Figura 2.3.

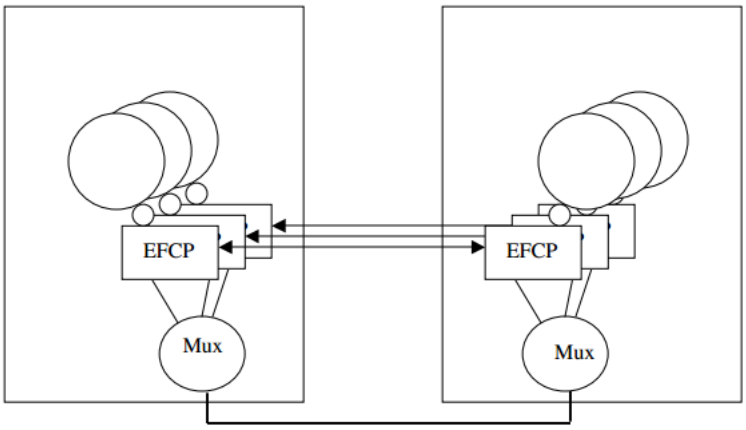

Figura 2.3 Várias aplicações se comunicando entre dois sistemas diferentes
Fonte: [Day10a]

Neste ponto, vemos que o EFCP agora precisará distinguir um fluxo de outro para ter a possibilidade de identificar quais mensagens serão enviadas por quais fluxos. Além disto, precisaremos de um gerenciamento do único canal de comunicação existente entre os sistemas para que ele seja utilizado simultaneamente por várias aplicações. Deveremos também permitir que várias instâncias de uma mesma aplicação existam e se comuniquem. Tudo isto nos leva à adição de um módulo de multiplexação para gerenciar o acesso à única mídia física e a um incremento nas funcionalidades do EFCP para dar suporte às novas demandas de serviço.

Continuando com a evolução do nosso cenário, vamos analisar o caso de múltiplas aplicações se comunicando entre múltiplos sistemas, conforme ilustrado na Figura 2.4.

Uma primeira solução que surge é replicar a estrutura ilustrada na Figura 2.3 em cada um dos canais de comunicação que ligam os sistemas. Desta maneira, as aplicações em um sistema têm a possibilidade de se comunicar com outras aplicações em sistemas diferentes. Porém, para que isto ocorra neste cenário, as aplicações precisam definir qual dos canais disponíveis irão utilizar para atingir o destino.

Para que a responsabilidade de escolha do link físico não fique sobre as aplicações, é necessária a adição de um módulo que seja responsável por esta tomada de decisão. O nosso cenário agora fica conforme ilustrado na Figura 2.5.

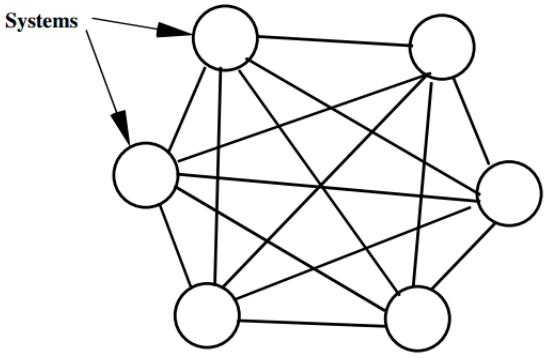

Figura 2.4 Várias aplicações se comunicando entre vários sistemas
Fonte: [Day10a]

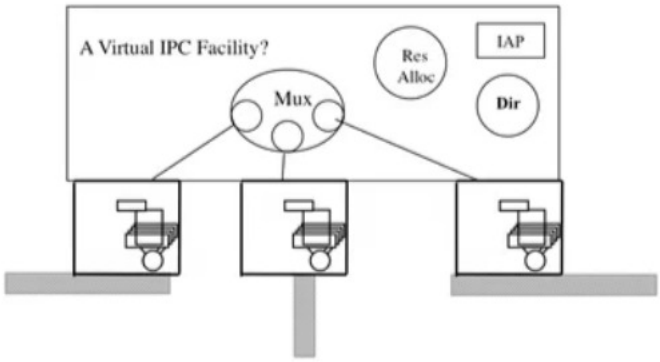

Figura 2.5 Reorganizando o cenário
Fonte: [Day11]

Temos então uma DIF para cada link e uma aplicação acima para gerenciar o seu uso e fornecer aos usuários uma interface comum de comunicação. Podemos perceber aqui que a estrutura da aplicação colocada para gerenciar o uso das DIF's, é bastante semelhante à estrutura das DIF's gerenciadas por ela. Esta semelhança vai além da semelhança estrutural, pois a funcionalidade deste gerenciador nada mais é do que estabelecer uma comunicação entre processos multiplexando os fluxos, a diferença reside apenas na camada inferior. Deparamo-nos, então, com uma DIF acima das outras anteriormente estabelecidas.

À medida que o número de sistemas aumenta dentro desta arquitetura, a quantidade de links necessários se torna impraticável, pois cada sistema precisa de uma conexão direta com todos os demais. Por conta disto, esta abordagem não se mostra escalável e tampouco financeiramente viável.

Para solucionar esse problema, partindo do princípio que todas as aplicações não se comunicam com todas as outras ao mesmo tempo, vamos modificar a estrutura para que a existência de uma mídia física conectando diretamente todos os sistemas presentes na rede não seja mais

necessária.

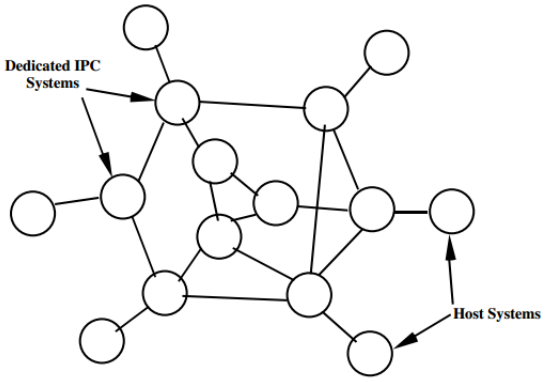

Figura 2.6 Tornando a arquitetura viável
Fonte: [Day11]

Conforme ilustrado na Figura 2.6, dedicando sistemas à realização de IPC, realizando repasse e multiplexação, reduz-se a quantidade de links necessários. Porém, tais sistemas dedicados podem ocasionar congestionamentos momentâneos e erros de bits podem ocorrer em suas memórias, o que leva à necessidade de termos o EFCP operando acima desses sistemas para garantir os parâmetros de qualidade de serviço requeridos pelo usuário.

Com isto, chegamos à estrutura ilustrada na Figura 2.7.

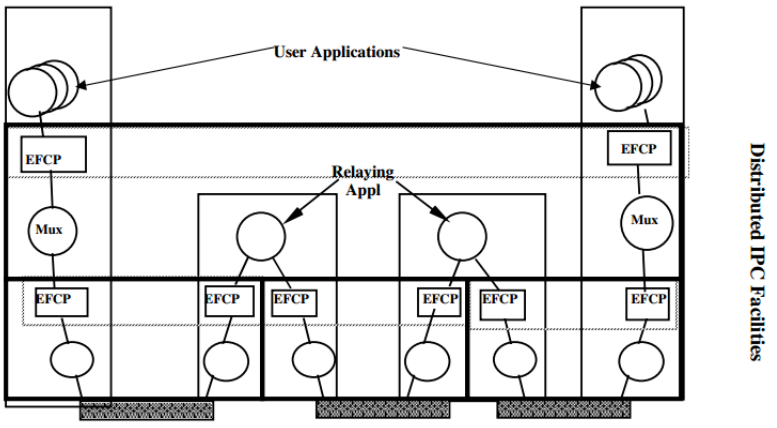

Figura 2.7 O modelo IPC
Fonte: [Day10a]

Observe que os retângulos destacados na figura acima realçam uma estrutura similar. Esta estrutura nada mais é do que uma aplicação IPC distribuída, uma DIF, que se repete. Com isso, John Day conclui que não são necessárias várias camadas diferentes como as que temos

na Internet atual. Nesta nova proposta, há apenas uma camada que se repete, consistindo de uma aplicação distribuída que gerencia funções IPC, mostrando que redes é um problema de computação distribuída e não de telecomunicações. Isto forma uma base para uma teoria geral de redes [Day10a].

2.2 Os Elementos de um Processo IPC

Os elementos de um processo IPC são divididos em três grupos. São eles: o protocolo de aplicação CDAP, os módulos IPC e o gerenciamento de IPC. Vamos, nessa seção, explanar o funcionamento de cada um deles.

2.2.1 O protocolo de aplicação CDAP

O CDAP é uma adaptação do protocolo CMIP, *Common Management Information Protocol*, que era usado para transferir informações entre o agente e o gerente em sistemas de gerenciamento de redes. No RINA, possui o papel de permitir a troca de dados estruturados entre dois processos de aplicação [IRA13a].

Ele define um conjunto completo de operações e mensagens que são utilizadas para o estabelecimento de conexão, como conectar, desconectar e autenticar, e operações em objetos, como criar, deletar, ler, escrever, iniciar e parar [rin13].

Também é utilizado em funções de gerenciamento de camada dentro da DIF, efetuando operações como registro na DIF, gerenciamento de segurança, alocação de portas, controle de acesso, monitoramento de QoS, gerenciamento de fluxo e roteamento [GTP+11]. Nele, as mensagens enviadas podem ser codificadas de qualquer maneira, desde que haja um acordo prévio entre as partes da comunicação [rin13].

Ao contrário da arquitetura atual, que possui vários protocolos de aplicação, o RINA propõe apenas o uso do CDAP. Nesta proposta, o que muda de aplicação para aplicação são os objetos manipulados e não o protocolo [Nol11].

2.2.2 Módulos IPC

O processo IPC possui os seguintes módulos: Delimitação de SDU's, do inglês *Source Data Unit*, Proteção de SDU's, EFCP, do inglês *Error and Flow Control Protocol*, Repasse e Multiplexação. Vamos a seguir explanar cada um deles.

2.2.2.1 Delimitação de SDU's

Consiste de mecanismos que são utilizados para indicar o começo e o fim de uma SDU [TGD+11]. Esta delimitação pode ser feita a partir de um padrão especial de bits utilizado para denotar os limites ou de um campo no cabeçalho indicando o tamanho, o que pode ser utilizado para calcular o final da SDU [TGD+11].

O IPC pode fragmentar ou combinar SDU's de acordo com a necessidade da comunicação e este módulo é utilizado para garantir que o serviço mantenha a identidade na entrega dos

pacotes, ou seja, para garantir que as mesmas SDU's enviadas sejam entregues ao destino [Day10b].

2.2.2.2 Proteção de SDU's

São mecanismos utilizados para evitar erros de *bytes* e prover confidencialidade e integridade à SDU [TGD+11].

Alguns mecanismos incluídos são: proteção de dados corrompidos, *checksums*, *hop count*, *time to live* e criptografia.

2.2.2.3 EFCP

O EFCP é o protocolo de transferência de dados do RINA que foi desenvolvido tendo como base o delta-t [Wat81]. Este protocolo tem os seus mecanismos separados das políticas e, por consequência, é dividido em dois módulos, o módulo de transferência de dados e o módulo de controle.

O módulo de transferência de dados, ou DTP, do inglês *Data Transfer Protocol*, implementa os mecanismos que estão fortemente ligados à SDU transportada, como fragmentação, sequenciamento, concatenação e separação. Já o módulo de controle de transferência de dados, ou DTCP, do inglês *Data Transfer Control Protocol*, implementa os mecanismos que estão fracamente acoplados à SDU, como controle de transmissão, controle de retransmissão e controle de fluxo [TGD+11].

2.2.2.4 Repasse

O módulo de repasse é responsável por encaminhar os pacotes recebidos pelo processo IPC, através da checagem do endereço presente no cabeçalho, para o processo de destino [TGD+11].

2.2.2.5 Multiplexação

Este módulo é responsável por mapear os fluxos vindos de uma camada em um determinado nível acima da camada à qual se encontra o processo IPC para fluxos de camadas em níveis inferiores [TGD+11].

2.2.3 Gerenciamento de IPC

Este elemento engloba o componente de alocação de fluxo, os mecanismos de autenticação, a alocação de recursos, o RIB, do inglês *Resource Information Base* e o RIB Daemon [TGD+11]. A seguir, vamos descrever cada um desses componentes.

- Alocação de Fluxo: Componente responsável por alocar e desalocar pedidos de conexão e por gerenciar cada novo fluxo que passa através do processo IPC.
- Mecanismos de Autenticação: São funções relacionadas à segurança, como autenticação e controle de acesso.

- Alocação de Recursos: Componente que gerencia a alocação de recursos e monitora os recursos de uma DIF através do compartilhamento de informações com outros processos IPC.
- RIB: Base de dados que armazena todas as informações disponíveis ao processo IPC representadas como objetos.
- RIB Daemon: Responsável por responder a requisições de informações partindo de outros membros da DIF, notificando-os periodicamente e/ou na ocorrência de certos eventos, ou de membros externos. O RIB Daemon também garante que as outras tarefas de gerenciamento possuem as informações que estão sendo requeridas.

2.3 Adoção

Uma característica importante do RINA é a sua proposta de implantação. Ao invés de propor uma completa substituição da arquitetura da Internet atual para que seus princípios sejam implantados, o que seria inviável na prática, RINA propõe um processo de adoção gradativa [Day].

A estrutura recursiva da arquitetura permite a implementação de DIF's em uma camada acima do IP, abaixo ou até mesmo lado a lado [Day]. Logo, esta arquitetura pode conviver harmoniosamente com a Internet atual e ser desenvolvida de maneira gradativa. Aplicações que já existem na Internet atual poderiam continuar utilizando-a da mesma maneira como a utilizam hoje e novas aplicações adotariam RINA à medida que seus benefícios sejam úteis a elas [Day11].

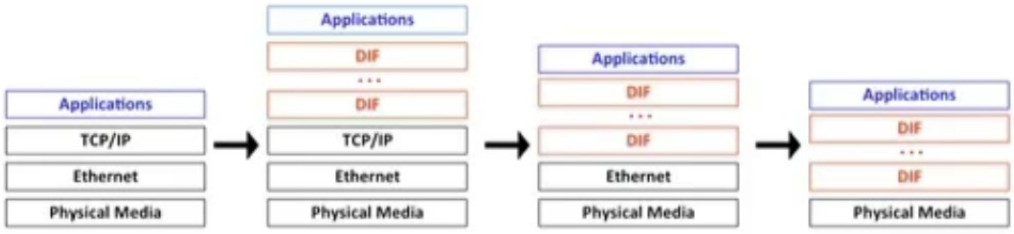

Figura 2.8 Modelo de adoção
Fonte: [Day11]

A Figura 2.8 ilustra uma estratégia de adoção que se baseia nesta flexibilidade das DIF's, ou seja, na característica de poder implementá-las na camada que se deseja. Um primeiro passo dessa estratégia é implantar as DIF's acima da camada TCP/IP e, a partir disto, mover a estrutura para camadas mais abaixo [Day11] até atingir o cenário desejado, que é a completa adoção.

Baseando-se nesta estratégia, alguns esforços têm sido efetuados para observar, na prática, o comportamento da rede em cada um desses cenários. Além disto, estes esforços visam apri-

morar e solidificar os conceitos e princípios do RINA a partir desta observação. Um exemplo de esforço neste sentido é a implementação de um protótipo da rede em JAVA para que o comportamento do RINA em uma camada acima do TCP/IP seja analisado. Este protótipo será melhor exposto no próximo capítulo. Outro exemplo é o projeto IRATI, que visa implementar um protótipo do RINA sobre a Ethernet para experimentar e avaliar o seu desempenho em comparação ao TCP/IP [IRA13b].

CAPÍTULO 3

RINA na Prática

Toda nova proposta de arquitetura precisa ter seus conceitos analisados em um cenário real de comunicação em rede para que seu comportamento, suas vantagens e desvantagens sejam observadas na prática e seus conceitos e princípios amadurecidos.

No processo de aprofundamento da pesquisa sobre RINA, a necessidade de prototipagem surgiu de imediato, não só como uma maneira de avaliar experimentalmente os benefícios teóricos introduzidos pela arquitetura, mas também como um veículo para forçar os pesquisadores a pensar, de uma maneira mais profunda, nos detalhes de tópicos de pesquisa ainda em aberto [GTB+12].

Para este fim, foi desenvolvido pela organização i2CAT [i2C13] um protótipo em JAVA que implementa os principais aspectos da arquitetura sobre a camada TCP/IP.

Este capítulo expõe, em sua primeira seção, as características, limitações e o uso deste protótipo em um cenário simples de comunicação em rede e, na seção seguinte, apresenta, de maneira sucinta, outras prototipagens desenvolvidas.

3.1 Protótipo JAVA

Este protótipo foi desenvolvido com o propósito de experimentar os conceitos e auxiliar na aprendizagem dos princípios fundamentais da arquitetura.

Para sua implementação, foi escolhida a linguagem JAVA por sua característica multiplataforma, dando a possibilidade de instalação e uso para usuários de diferentes sistemas.

O *software* disponibilizado é um *software* livre, o que permite ao usuário ter acesso ao código fonte e manipulá-lo de acordo com sua curiosidade.

O protótipo é de fácil instalação e possui um guia prático que pode ser encontrado na *wiki* do projeto no repositório *Github*[1].

De imediato, um primeiro benefício experimentado em sua utilização é a separação entre aplicações e a infraestrutura de rede. Na Internet atual, o *namespace* das aplicações é inteiramente ligado ao processo de endereçamento IP e a números de portas TCP/UDP [dLDG12]. Em RINA, as aplicações possuem nomes que são independentes das camadas inferiores e não contêm a semântica de identificação do seu ponto de conexão. Ao utilizar a estrutura provida pelo protótipo, as aplicações possuem nomes com escopo local na DIF em que estão registradas e só enxergam o nome da aplicação de destino. Isto fornece uma maior segurança à comunicação e mobilidade às aplicações.

Um cenário de comunicação entre sistemas em redes diferentes sem a necessidade direta de

[1] https://github.com/dana-i2cat/alba/wiki/Building%2C-installing-and-running-alba

middleboxes também pode ser experimentado com a utilização desta implementação do RINA. Uma vez em que uma DIF esteja estabelecida entre as redes, as aplicações em ambas as redes podem comunicar-se diretamente utilizando os serviços providos pelo IPC.

Este protótipo disponibiliza uma API que fornece acesso às principais funcionalidades da arquitetura e permite que aplicações possam ser desenvolvidas para se comunicar em rede utilizando RINA.

Visando auxiliar este desenvolvimento, o protótipo contém o código de aplicações exemplo desenvolvidas com esta API. A partir da observação da maneira como tais aplicações utilizam as funções disponibilizadas, pode-se obter um maior entendimento de como utilizá-las no contexto adequado.

O protótipo do RINA em JAVA oferece um console de gerenciamento onde o usuário pode observar os elementos presentes na RIB de um processo IPC, o que é útil pois permite verificar quais são os vizinhos do processo, outros processos inscritos na mesma DIF e as características dos fluxos alocados.

Além disto, a partir da sua inicialização, o protótipo exibe em tempo real o que está acontecendo no momento: alocação dos fluxos, registros na DIF, mensagens enviadas, mensagens recebidas, entre outros.

Uma aplicação de testes, chamada de RINABand, também está disponível. Ela permite que o usuário efetue testes de esforço na DIF, alocando um número de fluxos e enviando uma determinada quantidade de SDU's através deles. Como resultado, é exibido um relato da largura de banda vista por cada fluxo.

Toda implementação possui características particulares resultantes das decisões tomadas por seus desenvolvedores. Vamos mostrar a seguir algumas características deste protótipo JAVA.

3.1.1 Características da Implementação

A prioridade no desenvolvimento deste protótipo foi a implementação dos elementos essenciais da arquitetura. A criação de uma API que apresente o principal serviço provido por uma DIF, o estabelecimento de um fluxo entre aplicações com certos parâmetros de qualidade de serviço, é suficiente para a efetuação de uma primeira análise do funcionamento do RINA.

Seis operações precisam ser oferecidas por esta API: alocar e desalocar fluxo, escrever e ler dados, registrar e cancelar registro de aplicação [GTB+12]. Essas operações são providas pelos elementos do processo IPC, explanado no Capítulo 2.

Vamos expor a seguir as características da implementação dos principais elementos do processo IPC presentes neste protótipo.

3.1.1.1 O protocolo CDAP

A sintaxe do CDAP define o conteúdo que pode ser utilizado pelo protocolo em suas mensagens. São elas: conectar, criar, deletar, ler, escrever, iniciar e parar, além de suas respectivas respostas [GTB+12].

Várias codificações podem ser utilizadas em uma sintaxe CDAP. Nesta implementação, foi utilizado o GPB (*Google Protocol Buffers*).

O GPB foi escolhido por prover um processo eficiente de *encoding*, por ser largamente utilizado em produção pelo Google em sistemas distribuídos de grande escala e por possuir ferramentas de desenvolvimento abertas e de código livre para diversas linguagens de programação [GTB+12].

As mensagens CDAP consistem de uma sequencia de um ou mais campos onde o campo sempre presente, obrigatório, é o que indica o tipo da mensagem. Cada campo possui um nome identificador e seu valor [rin13].

Em um arquivo com a terminação *.proto* os valores e tipos dos campos do protocolo são descritos e, a partir desta definição, o compilador GPB é capaz de gerar o código que define o protocolo em diversas linguagens.

No protótipo que estamos analisando, o GPB foi utilizado para gerar a implementação do CDAP na linguagem JAVA.

3.1.1.2 *Resource Information Base*

O RIB é uma representação lógica, orientada a objetos, da informação conhecida por um processo de aplicação [GTB+12].

Para se comunicar, os processos trocam mensagens entre si se referindo aos objetos presentes no RIB por nome, o qual é uma árvore estruturada que possui os objetos em suas folhas.

Esta base de informações é implementada nesta versão do protótipo como um *hashtable* indexado pelos nomes dos objetos.

3.1.1.3 Inscrição

O procedimento de inscrição permite que o processo IPC obtenha informações de estado suficientes para se tornar um membro da DIF [GTB+12].

A implementação atual define a troca de objetos necessária para iniciar um novo processo IPC como membro da DIF. Esta troca de objetos inclui a verificação da validade do endereço e a atribuição de um endereço válido para o novo membro caso o existente seja inválido ou nulo. Informação sobre as classes de qualidade de serviço, conjunto de políticas e constantes de transferência de dados suportadas pela DIF, como o tamanho máximo de pacotes de dados e MPL, do inglês *Maximum Packet Lifetime*, são providos por esta troca.

3.1.1.4 Alocação de Fluxo

Este elemento do processo IPC é responsável por gerenciar o ciclo de vida de um fluxo.

É implementado, nesta versão, como um diretório distribuído simples que implementa uma estratégia de *broadcast* para disseminar entre os membros da DIF a ocorrência de atualizações no diretório.

Este diretório faz o mapeamento de nomes de aplicações para os processos IPC onde estão registrados. É atualizado sempre que um novo registro de aplicação ocorre e de tempos em tempos para descartar entradas que não estão sendo utilizadas.

A implementação atual é adequada somente para pequenas DIF's. Quanto maior a DIF, mais complexa precisa ser a estratégia de implementação do diretório como um banco de dados distribuído [rin13].

3.1.1.5 Alocação de Recursos

Este componente é responsável por decidir como os recursos no processo IPC serão alocados.

Na versão atual do protótipo, apenas a parte responsável por gerenciar a criação e remoção de fluxos na camada inferior está implementada.

O número de fluxos a serem criados é um valor estático definido no arquivo de configuração do protótipo.

3.1.1.6 DTP e DTCP

Os módulos de transferência e controle de dados do protocolo EFCP estão implementados neste protótipo através de um cabeçalho, chamado de PCI, do inglês, *Protocol Control Information*, que contém informações para controle da transmissão.

No caso do DTP, o PCI é concatenado aos dados do usuário e contém as informações requeridas para efetuar mecanismos de transferência de dados diretamente ligados aos dados transportados, como, por exemplo, endereçamento e sequenciamento.

Já no DTCP, nenhum dado do usuário é concatenado ao PCI e este é enviado para efetuar mecanismos de transferência de dados que estão fracamente ligados aos dados transportados, como controle de transmissão, retransmissão e fluxo por exemplo.

3.1.1.7 Repasse e Multiplexação

Uma implementação simples foi adotada para este módulo.

N filas FIFO (*First In, First Out*) de PDU's (*Protocol Data Unit*) são multiplexadas na ordem de chegada dos pacotes de acordo com suas regras de multiplexação e repassadas para outro processo IPC através do fluxo adequado baseado no endereço de destino.

3.1.2 Limitações

Apesar de bastante útil para o estudo e experimentação dos conceitos, o protótipo não fornece acesso a todas as funcionalidades da arquitetura. Como o objetivo principal era ter uma versão que fornecesse uma primeira experiência prática com os principais conceitos do RINA, algumas funcionalidades foram simplificadas e outras não foram implementadas, para que assim uma versão pudesse ser disponibilizada mais brevemente.

Como exemplos de funcionalidades que foram simplificadas temos os parâmetros de qualidade de serviços (QoS *Cubes*) e o endereçamento. No primeiro caso, somente dois tipos de QoS *Cubes* podem ser definidos nesta implementação: um equivalente aos serviços providos pelo TCP e outro equivalente ao UDP. Já o endereçamento foi simplificado para uma simples numeração sem qualquer informação topológica, informação esta que otimizaria o roteamento.

As funcionalidades de controle de congestionamento, autenticação e proteção de SDU's são exemplos de propriedades que não foram implementadas. Estas fogem do propósito inicial do protótipo de desenvolver de imediato o básico da arquitetura que é necessário para estabelecer a comunicação entre dois processos IPC.

3.1.3 Cenário simples de uso

A Figura 3.1 expõe um cenário simples de utilização do protótipo para a comunicação entre dois processos em rede.

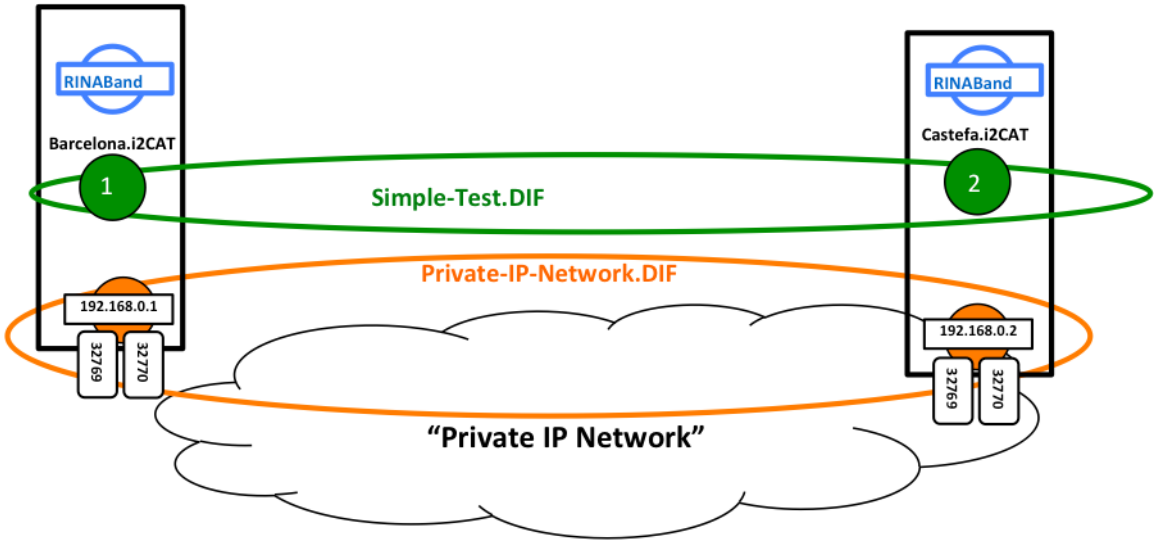

Figura 3.1 Experimentando em um cenário simples.
Fonte: https://github.com/dana-i2cat/alba/wiki/Experimenting-with-a-simple-scenario

Neste cenário, uma DIF é criada imediatamente sobre a rede IP para servir como uma camada base para a implementação de DIF's que sejam independentes de endereços IP. Esta camada, por servir como uma espécie de "calço" para outras, é chamada de *shim* DIF. Nela, duas portas específicas são alocadas em cada sistema para separar o fluxo de dados do fluxo de gerenciamento. Essas portas são definidas em um arquivo de configuração e, embora estejam ilustradas na Figura 3.1 como as portas 32769 e 32770, podem ser escolhidas conforme a preferência do usuário.

Esta DIF envolve a camada IP com a interface dos serviços providos por IPC, mapeando os nomes dos processos da camada acima para endereços IP e criando fluxos TCP e/ou UDP baseada nos parâmetros de qualidade de serviço solicitados na requisição de alocação de fluxo.

Uma vez estabelecida a *shim* DIF, outras camadas podem ser postas acima dela e utilizá-la para se comunicar.

A camada "Simple-Test.DIF", ilustrada na figura, é onde se pode desfrutar melhor das características do RINA. Nessa camada, o uso de endereço IP para encontrar o sistema de destino é transparente, de responsabilidade da camada inferior, e apenas uma visão do diretório da DIF é necessária para efetuar a comunicação.

Montado o cenário descrito acima, aplicações em cada sistema podem efetuar requisições de fluxo à DIF mais próxima e se comunicar utilizando a estrutura provida pelo protótipo. Na ilustração, dois processos da aplicação RINABand rodando em sistemas diferentes, um fazendo papel de cliente e o outro de servidor, se comunicam através da DIF "Simple-Test".

A criação dos processos IPC e o estabelecimento das DIF's é efetuado pelo protótipo a partir da configuração presente no arquivo de configuração do *software*. As instruções para o preenchimento deste arquivo também pode ser encontrado no guia prático de instalação do protótipo citado no início desta seção.

3.2 Outras Implementações

Outras abordagens de prototipagens foram implementadas para experimentar o RINA em diferentes ambientes. Nas subseções a seguir, iremos apresentar três desses protótipos.

3.2.1 TRIA

A organização *TRIA Network Systems* [2] desenvolveu sua implementação do RINA com os objetivos de prover um *framework* para testar os novos protocolos e experimentar o desenvolvimento de alguns módulos da arquitetura diretamente no núcleo de sistemas operacionais [rin13].

O *framework* para testes dos novos protocolos opera inteiramente no nível das aplicações de usuário para prover uma maior facilidade de análise. É implementado como uma máquina de estados simples, utilizando uma única *thread* para simplificar o bloqueio da execução, facilitando a análise passo-a-passo.

Os módulos movidos para o núcleo do sistema operacional são codificados na linguagem C e implementam as operações de gerenciamento de tempo, memória e *buffering* de maneira bastante similar às disponíveis no sistema operacional UNIX/Linux.

Esta implementação também provê uma API do RINA e a aplicação de testes como o RINABand.

3.2.2 RINA para GENI

GENI é um ambiente de testes de rede que permite a experimentação de novas arquiteturas [gen]. Um protótipo do RINA foi desenvolvido para este ambiente com o propósito de testar duas de suas características fundamentais: segurança e gerenciamento.

Estes testes foram efetuados a partir de dois cenários de experimentação: a autenticação de um processo IPC em uma DIF já existente para formar uma camada segura e a criação de uma nova DIF em um nível superior, acima de duas outras DIF's preexistentes.

Isto contribuiu para a elaboração de novos testes neste ambiente, visando comparar as características do RINA em relação a outras propostas para a Internet do futuro [WEM13].

3.2.3 IRATI

Este projeto tem como foco avançar o estado da arte do RINA, experimentando a arquitetura sobre a camada Ethernet e comparando seu desempenho com o da camada TCP/IP.

[2] http://www.trianetworksystems.com/

Seus principais objetivos são: melhorar o modelo de referência e as especificações do RINA focando na definição de DIF's sobre Ethernet, disponibilizar um protótipo de código livre para sistemas operacionais UNIX que implemente o RINA sobre Ethernet, validar experimentalmente a teoria do RINA e produzir um conjunto de requerimentos e testes de comparação com TCP/IP [Fig12].

O projeto teve seu início em Janeiro de 2013 e, apesar de não haver ainda disponibilizado publicamente uma versão do software, o site do projeto [3] já disponibiliza a documentação das especificações definidas até então.

[3] www.irati.eu

CAPÍTULO 4

Ferramenta para análise do protocolo CDAP

A análise de protocolos de uma arquitetura de redes é importante para sua compreensão. O entendimento de protocolos pode ser aprofundado através da visão deles em ação [KR13], observando a sequência de mensagens trocadas entre dois processos e os valores de cada campo do protocolo em questão. Neste capítulo iremos expor o desenvolvimento de uma ferramenta que, utilizada em conjunto com o protótipo JAVA descrito no Capítulo 3, auxilia a análise do protocolo CDAP da arquitetura RINA.

4.1 CDAP

O protocolo CDAP é o único protocolo de aplicação definido pelo RINA e é utilizado para efetuar conexão e operações em objetos.

Como ele define todas as possíveis operações que podem ser realizadas remotamente, este protocolo pode ser utilizado como base para efetuar as operações providas pelos protocolos de aplicação presentes na arquitetura atual.

Uma requisição HTTP do tipo *GET*, por exemplo, pode ser implementada com o uso da requisição *READ* do CDAP. Esta requisição lê o valor de um objeto de aplicação definido na mensagem e retorna, em uma mensagem do tipo *READ_R* (*read response*), o valor do objeto requisitado ou uma indicação de erro. De maneira similar, o método *POST* do HTTP pode ser também implementado em RINA utilizando a requisição *WRITE*, que escreve um valor especificado em um determinado objeto de aplicação, e sua resposta.

Como o CDAP é o único protocolo definido responsável pela troca de mensagens entre os processos na DIF, a sua análise e entendimento é de fundamental importância para a compreensão da arquitetura. Isto motivou o desenvolvimento de sua ferramenta de análise.

No protótipo JAVA, como dito no Capítulo 3, o CDAP foi implementado utilizando *Google Protocol Buffers* para serializar suas mensagens.

Os campos e valores de cada mensagem são serializados utilizando a técnica *varint*. Esta técnica representa inteiros utilizando um ou mais *bytes*, onde quanto menor o número, menor a quantidade de *bytes* para representá-lo.

Quando uma mensagem CDAP é codificada, os campos e valores são concatenados em um *stream* de bytes, onde cada campo, tipo e valor são representados em *varint*.

4.2 *Dissector* Wireshark

A ferramenta desenvolvida neste trabalho foi um módulo do conhecido software capturador de pacotes, Wireshark [1], para reconhecer as mensagens CDAP.

A Figura 4.1 exibe a interface do software Wireshark.

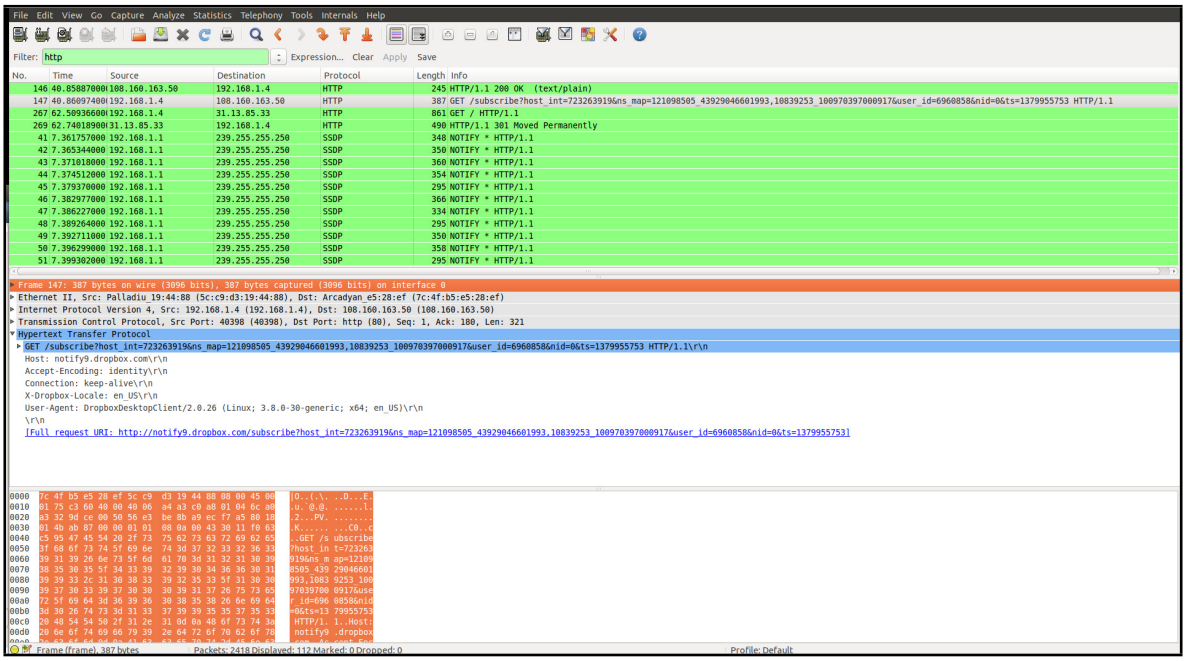

Figura 4.1 Interface Wireshark
Fonte: Imagem elaborada pelo autor.

Através do filtro localizado na parte superior e destacado em verde na imagem, podemos definir quais tipos de pacotes queremos observar. Ao digitar o nome do protocolo no filtro, a primeira seção da tela exibe os pacotes capturados em ordem de captura e algumas informações como fonte, destino e tamanho.

A segunda seção exibe, ao selecionarmos um pacote na primeira, a pilha de protocolos presente e permite que, ao expandir a seta localizada ao lado esquerdo do nome de cada protocolo, sejam visualizados os valores de seus campos. A terceira seção da tela, localizada na parte inferior, permite que observemos os bytes capturados na notação hexadecimal e ASCII.

O Wireshark exibe em sua interface o conteúdo de vários campos de protocolos das mensagens capturadas. Para que um protocolo seja reconhecido e tenha seus campos exibidos é necessário que haja um módulo no Wireshark responsável por "dissecá-lo", ou seja, reconhecer cada campo e cada valor do protocolo nos *bytes* das mensagens recebidas ou enviadas.

Este módulo se chama *dissector*. Cada *dissector* analisa os pacotes de uma porta específica. Portanto, como o protótipo JAVA define portas específicas na *shim* DIF para os fluxos de

[1] www.wireshark.org

gerenciamento e de dados, implementamos um *dissector* para capturar as mensagens CDAP trocadas no protótipo.

4.2.1 Implementação de um *Dissector* para o CDAP

O Wireshark é um software de código livre que permite a adaptação de *dissectors* já existentes ou a inclusão de novos. Novos *dissectors* podem ser implementados como parte do núcleo da aplicação ou como *plugins*.

A implementação de *plugins*, por sua característica modular, é mais simples de ser efetuada e, portanto, foi a modalidade escolhida para o desenvolvimento nesse trabalho.

Um série de funções para manipulação das mensagens capturadas é disponibilizada pelo Wireshark fazendo com que o desenvolvedor se concentre apenas no reconhecimento dos campos e valores do protocolo a ser analisado e no seu modo de exibição na interface.

Assim, o desenvolvedor é responsável por definir, no conjunto de *bytes* recebidos pelo plugin, *byte* a *byte*, o significado de cada campo e o seu valor para que o *software* apresente-os na tela para o usuário.

A análise de mensagens CDAP é dificultada pela característica do GPB de representar o conteúdo das mensagens na notação *varint*. Esta notação manipula o conteúdo da mensagem e faz com que o reconhecimento dos campos e valores exija uma série de operações no conjunto de *bytes* recebidos.

Com o objetivo de facilitar a decodificação de uma mensagem serializada com GPB por um *dissector* Wireshark, os desenvolvedores Dilip Joseph e Naveen Gogineni desenvolveram uma ferramenta, o protobuf-wireshark [2], que gera, a partir da definição do protocolo na sintaxe GPB, o seu *dissector*.

Esta ferramenta foi utilizada para gerar um *dissector* Wireshark a partir do arquivo de definição do protocolo CDAP contido no protótipo JAVA. Porém, foi necessário fazer uma adaptação no código gerado para que os dados do usuário, contendo a mensagem CDAP em si, fossem separados do cabeçalho criado pelo protocolo EFCP.

O *dissector* do CDAP permite que as mensagens sejam visualizadas com detalhe e na ordem em que foram capturadas. Esta ferramenta fornece ao usuário uma visão clara das mensagens que são trocadas, permitindo uma maior compreensão do protocolo.

As Figuras 4.2 e 4.3 ilustram duas mensagens capturadas na execução do protótipo JAVA no cenário descrito na seção 3.1.3 do Capítulo 3.

4.3 Exemplo de Utilização

Ao utilizar o protótipo no cenário descrito no Capítulo 3 e ilustrado na Figura 3.1 juntamente com o *dissector* CDAP desenvolvido, podemos ver as mensagens trocadas pela aplicação de testes RINABand.

Ao iniciarmos o RINABand no sistema “Castefa.i2cat”, com o papel de servidor, capturamos a mensagem ilustrada na Figura 4.4.

[2] https://code.google.com/p/protobuf-wireshark/

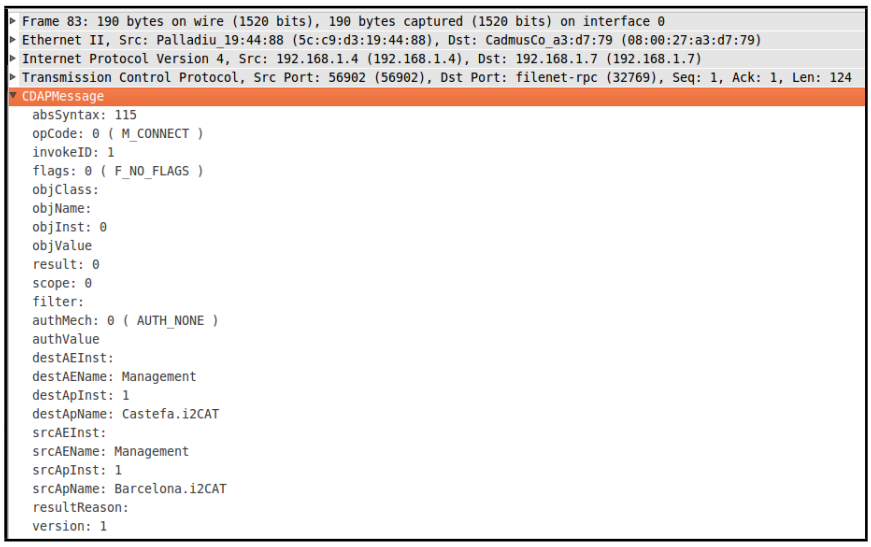

Figura 4.2 Mensagem M_CONNECT enviada de Barcelona.i2CAT para Castefa.i2CAT
Fonte: Imagem elaborada pelo autor.

Podemos observar nos valores exibidos pelo *dissector* que a classe, ou tipo, do objeto é uma entrada no diretório (*directory forwarding table entry*) e que seu valor contém o nome da aplicação (*rina.utils.apps.rinaband*). Também vemos que o tipo da operação a ser realizada (*opCode*) é M_CREATE. Logo, concluímos que esta mensagem está sendo enviada para solicitar a criação de uma entrada no diretório da DIF para a aplicação.

Ao iniciarmos, por sua vez, a aplicação no sistema "Barcelona.i2cat" com o papel de cliente, capturamos a mensagem ilustrada na Figura 4.5.

Podemos observar que o tipo da operação a ser realizada é também M_CREATE, que a classe do objeto é do tipo fluxo (*flow*) e que o valor do objeto da mensagem contém o nome da aplicação que solicita o fluxo (*rina.utils.apps.rinabandclient*). A partir desta análise, podemos concluir que a mensagem é enviada pela aplicação cliente para solicitar a criação de um fluxo de comunicação.

Ao receber a solicitação de criação de um fluxo de comunicação, a aplicação servidor envia uma mensagem M_CREATE_R como resposta, estabelecendo assim o canal de comunicação entre as aplicações cliente e servidor. A partir deste ponto, os processos podem começar a trocar dados entre si. A Figura 4.6 exibe a mensagem M_CREATE_R capturada.

Ao finalizar a comunicação entre os processos servidor e cliente, o servidor envia uma mensagem M_DELETE retirando a aplicação do diretório da DIF. Esta mensagem está ilustrada na Figura 4.7.

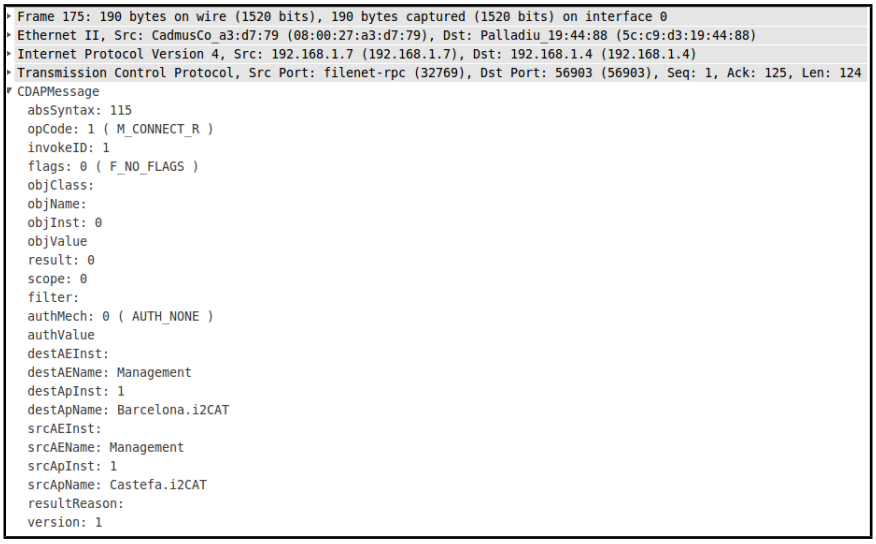

Figura 4.3 Mensagem M_CONNECT_R enviada de Castefa.i2CAT para Barcelona.i2CAT
Fonte: Imagem elaborada pelo autor.

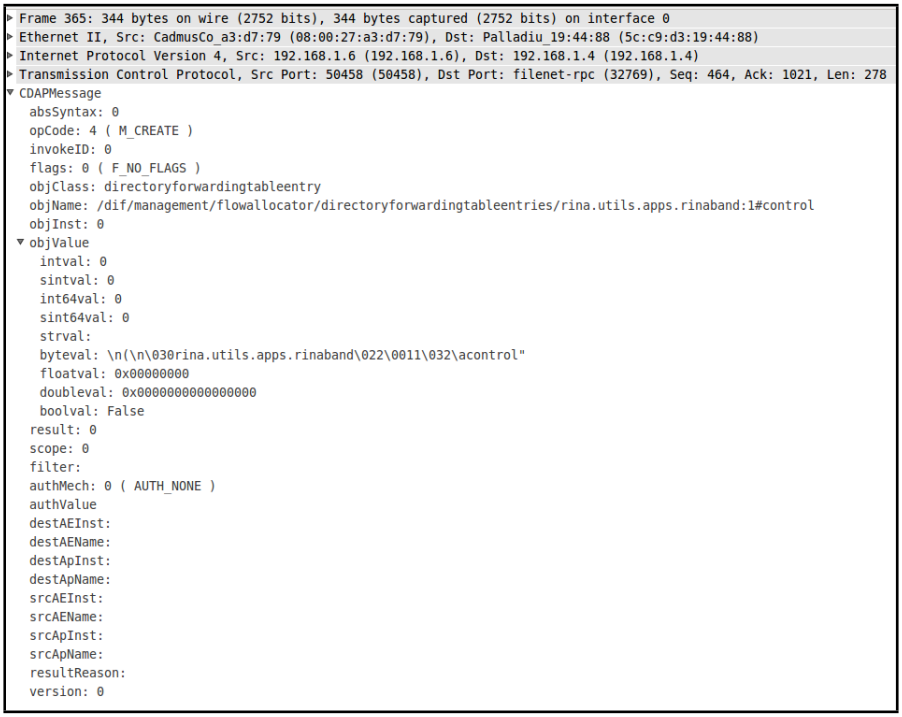

Figura 4.4 Mensagem M_CREATE enviada de Castefa.i2CAT para Barcelona.i2CAT
Fonte: Imagem elaborada pelo autor.

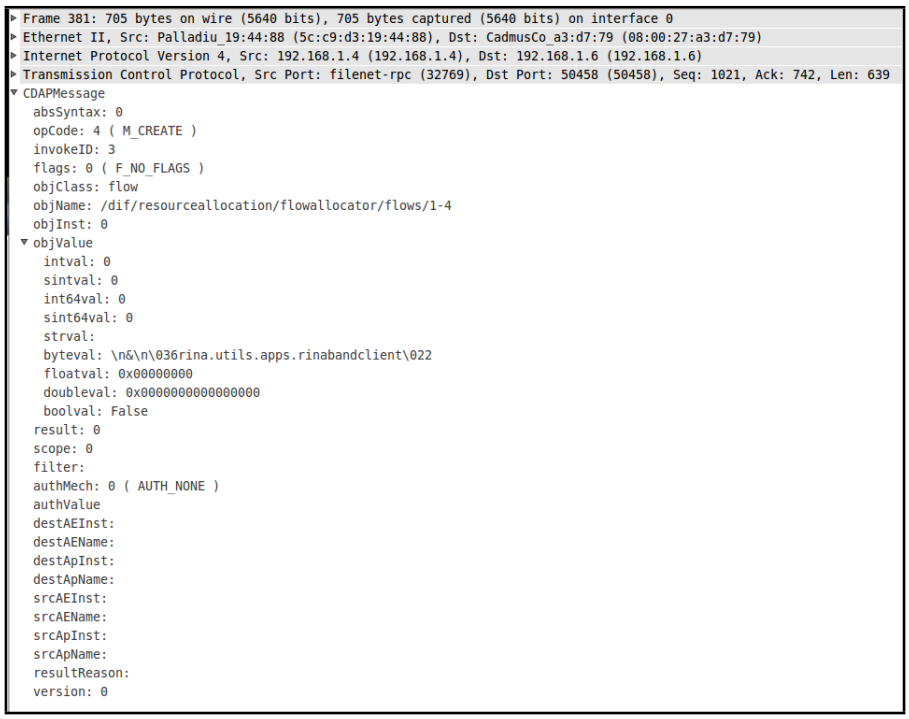

Figura 4.5 Mensagem M_CREATE enviada de Barcelona.i2CAT para Castefa.i2CAT
Fonte: Imagem elaborada pelo autor.

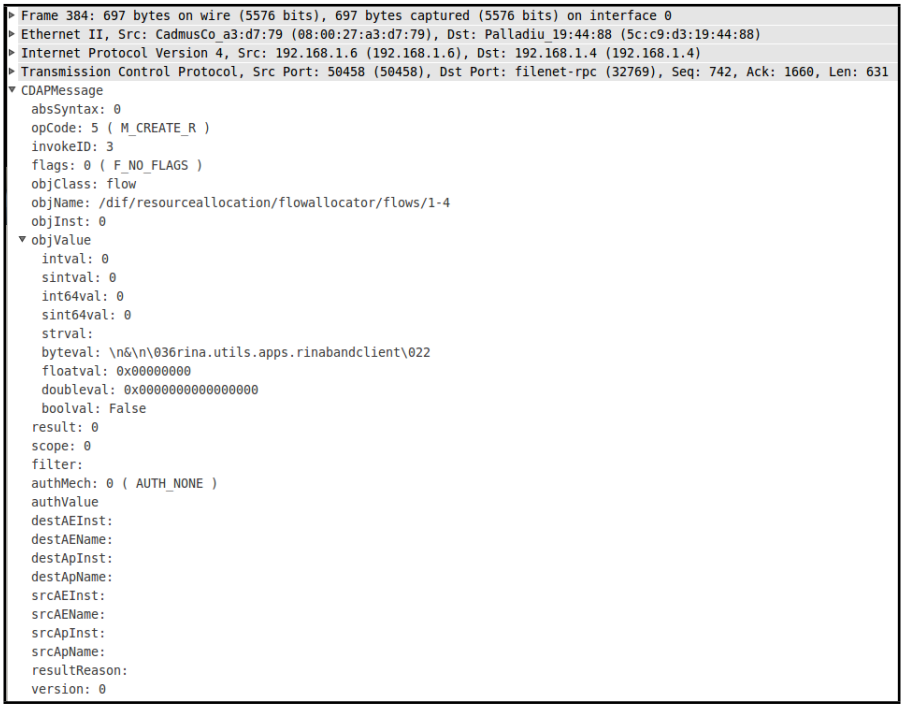

Figura 4.6 Mensagem M_CREATE_R enviada de Barcelona.i2CAT para Castefa.i2CAT
Fonte: Imagem elaborada pelo autor.

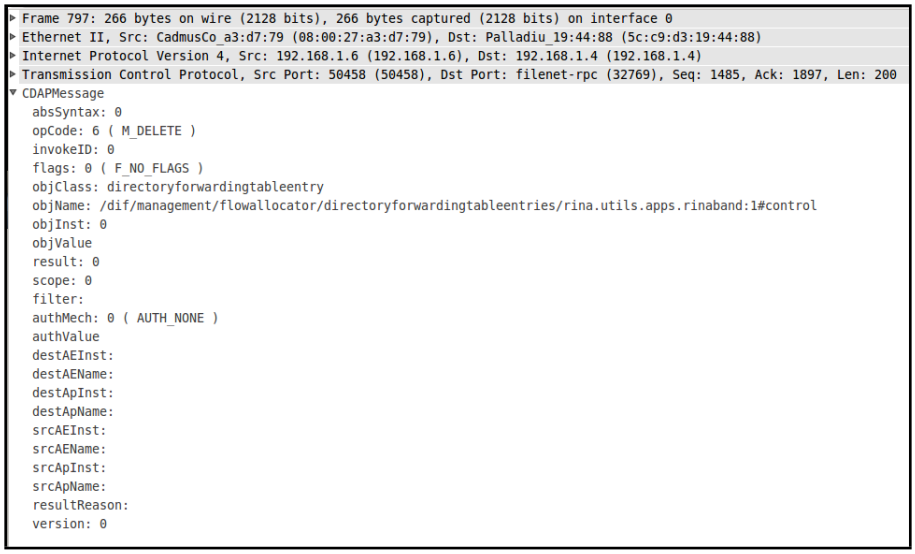

Figura 4.7 Mensagem M_DELETE enviada de Castefa.i2CAT para Barcelona.i2CAT
Fonte: Imagem elaborada pelo autor.

CAPÍTULO 5

Conclusão e Trabalhos Futuros

RINA é uma proposta de arquitetura para a Internet que busca criar uma rede que atenda naturalmente a demanda atual de serviços. Para avaliar os conceitos dessa nova proposta e para auxiliar sua aprendizagem, a necessidade de prototipagem surgiu e alguns protótipos foram desenvolvidos ou estão em desenvolvimento.

Este trabalho experimentou um desses protótipos, o protótipo JAVA desenvolvido pela organização i2CAT, e viu, na prática, processos se comunicando em rede utilizando IPC. Além disso, para auxiliar na compreensão da arquitetura, foi adaptado um analisador de protocolos para capturar e exibir os campos e valores das mensagens CDAP trocadas no cenário onde foi experimentado o protótipo.

5.1 Contribuições

O estudo da arquitetura RINA e a sua experimentação através do uso do protótipo JAVA e do *dissector* Wireshark contribui para o entendimento dos princípios básicos da comunicação em rede.

Após o estudo teórico da proposta, o processo de instalação e utilização do protótipo permitem uma visão prática, realizando efetivamente uma comunicação entre processos em sistemas diferentes através da arquitetura estudada. Isto permite a observação de resultados palpáveis dos benefícios propostos pelo modelo de John Day.

Este trabalho teve um papel de desbravamento na utilização deste protótipo, utilizando-o para a solidificação dos conceitos aprendidos e para a compreensão de princípios básicos para a troca de dados entre sistemas.

Além disso, o desenvolvimento de um *dissector* para o protocolo CDAP forneceu uma ferramenta para auxiliar no processo de construção de uma maturidade não só no entendimento do RINA mas também na análise de protocolos.

Através da construção de um *plugin* para o Wireshark, foi obtido um contato direto com os dados trocados entre os processos IPC instanciados pelo protótipo para manipulá-los e entender seus campos e valores, o que contribuiu para o entendimento do processo de análise de protocolos do *software* Wireshark e, de maneira mais específica, para a compreensão do funcionamento do protocolo CDAP.

Para o estudante de redes acostumado com o modelo tradicional da Internet atual, estudar RINA amplia seus horizontes. Apesar de poder parecer estranho no início, esta nova visão focada nos princípios fundamentais de comunicação muda paradigmas e permite a visualização de soluções para problemas atuais que eram dificultadas pelo conhecimento único da arquite-

tura em uso.

Este trabalho contribuiu para a obtenção de um entendimento sólido da estrutura necessária para o funcionamento de uma internet, dando uma nova visão e ampliando os conceitos daqueles que até então eram acostumados com a arquitetura atual.

5.2 Trabalhos Futuros

Para solidificar ainda mais os conhecimentos adquiridos com o estudo da nova arquitetura, algumas atividades podem ser efetuadas. As seguir são apresentadas algumas sugestões.

5.2.1 Funcionalidades do protótipo JAVA

Na versão atual do protótipo JAVA, todo pedido de inscrição na DIF é aceito independente se as políticas de fluxo são aceitáveis ou não. Pode-se implementar a autenticação da fonte da requisição e, se a aplicação tiver acesso ao destino, a verificação da aceitabilidade das políticas de fluxo requisitadas.

Toda requisição de fluxo é mapeada para a primeira DIF do tipo *shim* DIF encontrada. Pode-se implementar como exercício a alocação de fluxo à DIF da camada inferior sem, necessariamente, precisar ser esta do tipo *shim*.

Outra possibilidade é a limitação de fluxos alocados por entidades de transferência de dados da aplicação, AE's, do inglês *Application Entities*, de um processo IPC. Atualmente toda requisição é aceita, enquanto que o correto é que só seja permitido um fluxo para cada AE de um processo IPC por vez.

Também é assumido nesta versão do protótipo que o nome de cada aplicação encontra-se no diretório local da DIF, logo, se a aplicação não estiver presente, nunca será encontrada, mesmo que esteja em uma outra DIF existente. O estudante precisará implementar neste caso o IDD, do inglês *Inter-DIF Directory*, o que será um excelente exercício para a solidificação do entendimento do processo de nomeação (*naming*) da arquitetura.

Ao adicionar o IDD ao protótipo, será possível adicionar mais uma DIF no cenário explanado na seção anterior e uma comunicação entre processos em DIF's distintas poderá ser efetuada e observada.

Além das atividades citadas acima, o aluno pode aproveitar a estrutura modular do código do protótipo e implementar a sua própria versão de uma parte da arquitetura. Módulos como o processo IPC, RIB Daemon e o próprio CDAP podem ser substituídos facilmente por outra implementação em JAVA.

5.2.2 Protocolos e serviços

É interessante a implementação de um outro *dissector* Wireshark para a análise do protocolo EFCP. Como este protocolo não é implementado utilizando *Google Protocol Buffers*, o seu dissector não precisa ser gerado pela ferramenta protobuf-wireshark. Os campos do EFCP são estáticos e todos eles estão sempre presentes no cabeçalho do pacote. Isto torna simples a implementação de um *dissector*.

Também pode ser utilizada como exercício a criação de aplicações com serviços equivalentes aos providos pelos protocolos de aplicação da Internet, como HTTP, FTP e SMTP, fazendo uso do protocolo CDAP provido pela API RINA.

5.2.3 Projeto IRATI

É bastante recomendável que sejam analisadas as especificações já disponibilizadas pelo projeto IRATI, como a especificação de uma DIF sobre Ethernet, por exemplo, para que, fazendo um contraponto com o que foi visto na utilização do protótipo JAVA, o conhecimento sobre o estado da arte da arquitetura seja solidificado.

5.3 Considerações Finais

RINA é uma proposta para a Internet do futuro que promete trazer simplicidade à rede e, ao mesmo tempo, sanar as dificuldades encontradas na arquitetura atual para suprir demandas de serviços como *multihoming*, mobilidade e segurança.

Por esse motivo, vários investimentos têm sido disponibilizados para pesquisar mais a fundo essa arquitetura e trazer dados concretos de comparação de desempenho com o modelo vigente.

Essas pesquisas realizadas, muito provavelmente, irão deixar um legado que permitirá com que aplicações que necessitem das características dessa nova arquitetura possam utilizá-las convivendo lado a lado com a Internet atual. Isto se deve ao modelo de adoção proposto e é um dos principais fatores que levam a crer que RINA estará presente, de maneira completa ou parcial, na rede mundial de computadores.

Referências Bibliográficas

[BFH02] Bob Braden, Ted Faber, and Mark Handley. From protocol stack to protocol heap – role-based architecture (RBA). ACM HotNets I, Princeton University, October 2002.

[CPRW03] David D. Clark, Craig Partridge, J. Christopher Ramming, and John T. Wroclawski. A knowledge plane for the internet. In *Conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–10, 2003.

[Day] John Day. How PNA works: The future of networking. Disponível em: http://pouzin.pnanetworks.com/images/HowPNAWorks100506.pdf.

[Day08] John Day. *Patterns in Network Architecture, A Return to Fundamentals*. Pearson Education, 2008.

[Day10a] John Day. An introduction to RINA or how I learned to stop worrying and love the internet. FutureNet Tutorial Part I, May 2010.

[Day10b] John Day. An introduction to RINA or how I learned to stop worrying and love the internet. FutureNet Tutorial Part III, May 2010.

[Day11] John Day. How to clean a slate. Waterford Institute of Technology, Waterford, May 2011.

[dLDG12] Miguel Ponce de Leon, John Day, and Eduard Grasa. RINA java prototype demo and development plans. PhD Course on Future Network Architectures and Experimentation, March 2012.

[DMM08] John Day, Ibrahim Matta, and Karim Mattar. Networking is IPC: a guiding principle to a better internet. In *CoNEXT '08 Proceedings of the 2008 ACM CoNEXT Conference*, number 67, 2008.

[dTeA09] Grupo de Teleinformática e Automação. Projeto Horizon. International workshop: New architectures for future Internet, September 2009.

[Fig12] Sergi Figuerola. IRATI - investigating RINA as an alternative to TCP/IP. FIRE Engineering Workshop, Ghent, Belgium, November 2012.

[gen] GENI. http://www.geni.net. Acessado em 25 de Setembro de 2013.

[GL06] Guan-Qun Gu and Jun-Zhou Luo. Some issues on computer networks: Architecture and key technologies. *J. Comput. Sci and Technol.*, 21(5):708–722, September 2006.

[GTB+12] Eduard Grasa, Eleni Trouva, Steve Bunch, Peter DeWolf, and John Day. Developing a RINA prototype over UDP/IP using TINOS. In *Proceedings of the 7th International Conference on Future Internet Technologies*, pages 31–36, 2012.

[GTP+11] Eduard Grasa, Eleni Trouva, Patrick Phelan, Miguel Ponce de Leon, John Day, Ibrahim Matta, Lubomir T. Chitkushev, and Steve Bunch. Design principles of the recursive internetwork architecture (RINA). FIArch: Call for Position Papers on Internet Design Principles, Brussels, May 2011.

[i2C13] i2CAT. i2cat fundació. http://www.i2cat.net/en, 2013. Acessado em 19 de Setembro.

[IRA13a] IRATI. Introduction to RINA. http://irati.eu/introduction-to-rina/, 2013. Acessado em 18 de Setembro.

[IRA13b] IRATI. Investigating RINA as an alternative to TCP/IP. http://irati.eu/, 2013. Acessado em 18 de Setembro.

[Jai06] Raj Jain. Internet 3.0: Ten problems with current internet architecture and solutions for the next generation. *Proceedings IEEE Military Communications Conference*, October 2006.

[KR09] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet*. Pearson Education do Brasil, fifth edition, 2009.

[KR13] James F. Kurose and Keith W. Ross. *Computer Networking A Top-Down Approach*. Pearson Education Limited, sixth edition, 2013.

[MFCD09] Marcelo D. D. Moreira, Natalia C. Fernandes, Luís Henrique M. K. Costa, and Otto Carlos M. B. Duarte. Internet do futuro: Um novo horizonte, 2009. Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2009. : , 2009, v. 1, p. 1-59.

[Nol11] John Nolan. The last waltz and moving beyond TCP/IP. *The Journal of the Institute of Telecommunications Professionals*, 5(3):42–50, 2011.

[rin13] RINA detailed overview and implementation discussions, January 2013. RINA Workshop. Barcelona.

[TGD+11] Eleni Trouva, Eduard Grasa, John Day, Ibrahim Matta, Lubomir T. Chitkushev, Patrick Phelan, Miguel Ponce de Leon, and Steve Bunch. Is the internet an unfinished demo? Meet RINA! TERENA Networking Conference, Prague, Czech Republic., May 2011.

[Wat81] R. W. Watson. Delta-t protocol specification. Technical report, Lawrence Livermore National Laboratory, 1981.

[WEM13] Yuefeng Wang, Flavio Esposito, and Ibrahim Matta. Demonstrating RINA using the GENI testbed. The 16th GENI Engineering Conference, Salt Lake City., March 2013.

[wir13] Wireshark. http://www.wireshark.org/, 2013. Acessado em 19 de Setembro.