Universidade Federal Fluminense

UBIRATAM CARVALHO DE PAULA JUNIOR

Posicionamento de Réplicas em Redes de Distribuição de Conteúdos

NITERÓI

2011

Universidade Federal Fluminense

UBIRATAM CARVALHO DE PAULA JUNIOR

Posicionamento de Réplicas em Redes de Distribuição de Conteúdos

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação. Área de Concentração: Algoritmos Distribuídos.

Orientadora:

Lúcia Maria Assumpção Drummond

NITERÓI

2011

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

D419 De Paula Junior, Ubiratam Carvalho
Posicionamento de réplicas em redes de distribuição de conteúdos / Ubiratam Carvalho de Paula Junior. – Niterói, RJ : [s.n.], 2011.
83 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2011.
Orientador: Lúcia Maria Assumpção Drummond.

1. Algoritmo. 2. Heurística. 3. Programação (Matemática). 4. Rede de distribuição de conteúdos. 5. Problemas de posicionamento de réplicas. I. Título.

CDD 005.136

Posicionamento de Réplicas em Redes de Distribuição de Conteúdos

Ubiratam Carvalho de Paula Junior

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação. Área de Concentração: Algoritmos Distribuídos.

Aprovada por:

Profa. Lúcia M. A. Drummond / IC-UFF(Presidente)

Prof. Eduardo Uchoa Barboza / EP-UFF

Prof. Luidi Simonetti / IC-UFF

Profa. Silvana Rossetto / UFRJ

Prof. Lucidio Cabral / UFPB

Niterói, 29 de agosto de 2011.

À minha família.

Agradecimentos

A Deus, primeiramente.

À minha querida família, principalmente meus pais, Ubiratam e Fátima e minha avó, Maria Eugênia, pelo amor, carinho e incentivo sempre presentes na minha vida.

À minha namorada Rafaelli, pelo seu amor, companheirismo e paciência em todos estes anos.

À professora Lúcia Drummond, pela sua importante orientação neste trabalho.

Aos professores Eduardo Uchoa, Luidi Simonetti e Yuri Abitbol, pelas contribuições realizadas neste trabalho.

Aos professores e funcionários do IC-UFF, pela competência e dedicação sempre presentes.

Aos amigos e colegas do IC-UFF e do LOGIS, principalmente ao Tiago Neves por sua colaboração neste trabalho.

À CAPES, pela bolsa de mestrado concedida.

Resumo

Uma Rede de Distribuição de Conteúdos (RDC) é uma rede sobreposta na qual servidores replicam conteúdos e distribuem requisições de clientes com o objetivo de reduzir o atraso, as cargas dos servidores e da rede, melhorando a Qualidade de Serviço (QoS) percebida pelos clientes. Neste cenário surge o Problema de Posicionamento de Réplicas (PPR) que consiste em posicionar as réplicas nos servidores, respeitando os recursos disponíveis, com o objetivo de minimizar os custos de replicação e comunicação. Neste trabalho são propostas uma formulação matemática, uma heurística centralizada e duas heurísticas distribuídas, considerando características mais realísticas do PPR do que as encontradas na literatura relacionada. Os resultados mostraram que as heurísticas distribuídas propostas apresentaram resultados satisfatórios para a versão dinâmica do problema quando comparada com a solução ótima global, a heurística ótima por período e uma heurística distribuída presente na literatura relacionada.

Palavras Chaves. Redes de Distribuição de Conteúdos, Problema de Posicionamento de Réplicas, Heurísticas Distribuídas

Abstract

A Content Distribution Network (CDN) is an overlay network in which servers replicate contents and distribute client requests in order to reduce the delay, the server and network loads, improving the quality of service (QoS) perceived by customers. In this scenario, the Replica Placement Problem (RPP) arises. It consists on position the replicas on servers, respecting the resources available so that the replication and communication costs are minimized. In this work a mathematical formulation, a centralized heuristic and two distributed heuristics are proposed, considering more realistic features of PPR than those found in related literature. The results showed that the proposed distributed heuristics presented satisfactory results for the dynamic version of the problem when compared with the global optimal solution, with the optimal heuristic for each period and with a distributed heuristic present in related literature.

Key Words. Content Distribution Network, Replica Placement Problem, Distributed Heuristics

Glossário

QoS	:	Quality of Service - Qualidade de serviço
RDC	:	Rede de Distribuição de conteúdos
PPR	:	Problema de Posicionamento de Réplicas em RDC
PAC	:	Problema de Atribuir Clientes a Servidores da RDC

Sumário

Lista de Figuras **ix**

Lista de Tabelas **xii**

1 Introdução **13**

1.1 Objetivos e Contribuições do Trabalho 14

1.2 Organização do Trabalho 14

2 Descrição do Problema e Trabalhos Relacionados **16**

2.1 Definição do Problema 16

2.2 Algoritmos Centralizados 18

2.3 Algoritmos Distribuídos 20

3 Formulações Matemáticas e Heurística Centralizada **24**

3.1 Formulação Matemática I 24

3.2 Heurística Ótima por Período 26

3.3 Formulação Matemática II 27

4 Algoritmos Distribuídos **30**

4.1 Sistema Distribuído 30

4.2 Algoritmo Distribuído RPA 31

4.2.1 Procedimentos 31

4.2.2 Exemplo 38

4.2.3 Complexidades 46

4.3 Algoritmo Distribuído RPA_Mod 47
4.3.1 Procedimentos 47
4.3.2 Exemplo 49
4.3.3 Complexidades 59

5 Resultados Computacionais **61**

5.1 Formulação Matemática I × Heurística Ótima por Período × RPA 61
5.2 Formulação Matemática II × RPA_Mod 64
5.3 RPA_Mod × Presti 68

6 Conclusão e Trabalhos Futuros **78**

Referências **80**

Lista de Figuras

2.1 Exemplo do problema de posicionamento de réplicas 17

2.2 Arquitetura da RDC 18

4.1 Dados do problema. 38

4.2 Um configuração inicial. 39

4.3 Cada servidor tenta atender as requisições localmente. 40

4.4 Cada servidor envia $STATIC$ com as requisições não atendidas e seus recursos disponíveis. 40

4.5 O servidor central envia $STATIC$ para os servidores. 41

4.6 Novas requisições surgem no período de tempo seguinte. 41

4.7 Cada servidor tenta atender as requisições localmente. 42

4.8 Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo. 42

4.9 Os servidores enviam $CHECKED$ com as requisições atendidas e não atendidas. 43

4.10 Fim de atendimentos locais. 43

4.11 Fim de atendimentos externos. 44

4.12 Cada servidor tenta atender as requisições localmente. 44

4.13 Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo. 45

4.14 Os servidores enviam $CHECKED$ com as requisições atendidas e não atendidas. 45

4.15 O servidor S_1 envia $ATTEND_CENTRAL$ para o servidor origem. . . . 46

4.16 Dados do problema. 51

4.17 Uma configuração inicial. 51

4.18 Cada servidor tenta atender as requisições localmente. 52

4.19 Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo. 53

4.20 Os servidores enviam CHECKED com as requisições atendidas e não atendidas. 53

4.21 O servidor S_2 envia $CHECK$ para o segundo servidor mais próximo que possui o conteúdo. 54

4.22 O servidor S_1 envia $CHECKED$ com as requisições atendidas e não atendidas. 54

4.23 Novas requisições surgem no período de tempo seguinte. 55

4.24 Cada servidor tenta atender as requisições localmente. 55

4.25 O servidor S_3 envia $CHECK$ para o servidor mais próximo que possui o conteúdo. 56

4.26 O servidor S_1 envia $CHECKED$ com as requisições atendidas e não atendidas. 56

4.27 Fim de atendimentos locais. 57

4.28 Fim de atendimentos externos. 57

4.29 Cada servidor tenta atender as requisições localmente. 58

4.30 Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo. 58

4.31 Os servidores enviam $CHECKED$ com as requisições atendidas e não atendidas. 59

4.32 O servidor S_1 envia $ATTEND_CENTRAL$ para o servidor origem. . . . 59

5.1 Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 10 servidores. 76

5.2 Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 20 servidores. 76

5.3 Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 30 servidores. 77

5.4 Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 50 servidores. 77

Lista de Tabelas

2.1 Trabalhos Relacionados - Abordagens Centralizadas 20

2.2 Trabalhos Relacionados - Abordagens Distribuídas 23

5.1 Formulação Matemática I × RPA × Heurística Ótima por Período - para 10 servidores 62

5.2 Formulação Matemática I × RPA × Heurística Ótima por Período - para 20 servidores 63

5.3 Formulação Matemática I × RPA × Heurística Ótima por Período - para 30 servidores 63

5.4 Formulação Matemática I × RPA × Heurística Ótima por Período - para 50 servidores 64

5.5 Formulação Matemática II × RPA_Mod - para 10 servidores 65

5.6 Formulação Matemática II × RPA_Mod - para 20 servidores 66

5.7 Formulação Matemática II × RPA_Mod - para 30 servidores 67

5.8 Formulação Matemática II × RPA_Mod - para 50 servidores 67

5.9 Primeira comparação entre Presti × RPA_Mod - para 10 servidores 71

5.10 Primeira comparação entre Presti × RPA_Mod - para 20 servidores 72

5.11 Primeira comparação entre Presti × RPA_Mod - para 30 servidores 72

5.12 Primeira comparação entre Presti × RPA_Mod - para 50 servidores 73

5.13 Segunda comparação entre Presti × RPA_Mod - para 10 servidores 73

5.14 Segunda comparação entre Presti × RPA_Mod - para 20 servidores 74

5.15 Segunda comparação entre Presti × RPA_Mod - para 30 servidores 75

5.16 Segunda comparação entre Presti × RPA_Mod - para 50 servidores 75

Capítulo 1

Introdução

Com o crescente número de pessoas que possuem acesso a Internet e com a maior velocidade proporcionada pela Internet banda larga, tem-se um grande aumento no acesso a conteúdos, principalmente os multimídias. Desse modo, uma estrutura de um único servidor web de conteúdo é insuficiente, pois o mesmo torna-se um gargalo.

Uma solução para este problema é o uso de múltiplos servidores que possuem réplicas dos conteúdos e estão, geralmente, mais próximos dos clientes (usuários). Grandes empresas atualmente disponibilizam esse serviço através de uma hierarquia de servidores distribuidamente localizados em todo o mundo, como por exemplo a *Akamai Technologies* [2].

A estratégia de utilização de múltiplos servidores caracteriza uma Rede de Distribuição de Conteúdos (RDC), que consiste de uma rede sobreposta [13] a Internet na qual servidores replicam conteúdos e distribuem requisições de clientes com o objetivo de reduzir o atraso, as cargas dos servidores e da rede, melhorando a Qualidade de Serviço (QoS) percebida pelos clientes.

Existem três problemas principais envolvendo uma RDC. O primeiro é o Posicionamento dos servidores de distribuição na rede em questão. O segundo é Problema de Posicionamento de Réplicas (PPR), que consiste em posicionar as réplicas dos conteúdos nestes servidores. O terceiro é o Problema de Atribuir Clientes a servidores (PAC), que consiste em associar as requisições dos clientes com servidores que possuem os conteúdos requeridos. O presente trabalho trata o segundo problema. Entretanto, para resolver o PPR é necessário considerar as requisições dos cliente. Desse modo, o PAC também é resolvido de forma aproximada e gulosa.

1.1 Objetivos e Contribuições do Trabalho

O objetivo deste trabalho é apresentar uma solução para o PPR utilizando técnicas de programação distribuída e de otimização matemática, tratando características tais como: limitações de recursos (largura de banda e capacidade de armazenamento) nos servidores, mudanças no estado da rede e banda necessária para QoS nas requisições que são apresentadas dinamicamente ao longo do tempo, normalmente presentes nas RDC reais.

Desta forma, neste trabalho são propostos dois algoritmos distribuídos, uma formulação matemática e uma heurística centralizada para a versão dinâmica do PPR. O segundo algoritmo distribuído é uma modificação do primeiro, incluindo outras características de RDCs reais, como por exemplo, mudança do estado dos enlaces da rede. A formulação matemática e a heurística centralizada são propostas e implementadas para realizar uma avaliação dos algoritmos distribuídos através de comparações.

A justificativa para o desenvolvimento de algoritmos distribuídos é que a natureza do problema é distribuída. Ou seja, as informações estão distribuídas geograficamente. Com isso, para solucionar o problema de forma centralizada, é adicionado um alto custo para coletar os dados distribuídos, resolver o problema e distribuir a solução. Vale ressaltar que para o caso real e dinâmico, na maioria das vezes isso se torna inviável em relação aos recursos computacionais necessários (CPU e memória) e também ao tempo gasto, pois o número de requisições e estado da rede variam muito em curtos períodos de tempo.

A principal contribuição deste trabalho em relação aos demais presentes na literatura é tratar a versão dinâmica do problema de forma geral, considerando diversos requisitos da RDC simultaneamente e de forma distribuída. Os algoritmos encontrados na literatura são específicos para determinadas tipos topologias, tratam apenas um conteúdo ou classe de conteúdo, ou são estáticos. Uma outra contribuição é que também são propostas uma formulação matemática e uma heurística centralizada para comparar e avaliar os algoritmos distribuídos.

1.2 Organização do Trabalho

O restante da dissertação está organizada da seguinte forma. No Capítulo 2, são apresentadas a definição do problema, a Formulação Matemática I e a heurística centralizada propostas nesta dissertação, além da Formulação Matemática II, proposta em [17]. No Capítulo 3, alguns trabalhos relacionados centralizados e distribuídos da literatura são

comentados. No Capítulo 4, são apresentados os pseudocódigos dos algoritmos distribuídos propostos e suas complexidades. No Capítulo 5, são expostos os resultados experimentais para as três comparações realizadas. Na primeira, o primeiro algoritmo distribuído proposto, chamado de RPA, é avaliado comparando seus resultados com os obtidos através da Formulação Matemática I e da heurística centralizada. Na segunda comparação, o segundo algoritmo distribuído, chamado de RPA_Mod, é avaliado comparando seus resultados com os obtidos através da Formulação Matemática II. Por fim, na terceira comparação, o algoritmo RPA_Mod também é avaliado comparando seus resultados com os obtidos através da heurística distribuída proposta em [22, 19], aqui chamada de Presti. Finalmente, no Capítulo 6, uma conclusão é apresentada, resumindo as contribuições deste trabalho e apontando os trabalhos futuros.

Capítulo 2

Descrição do Problema e Trabalhos Relacionados

Neste capítulo, é apresentada a descrição do problema de posicionamento de réplicas em uma RDC. Além disso, também são apresentados os trabalhos relacionados para o PPR e problemas correlatos. Estes trabalhos foram divididos em duas seções, sendo a primeira para aqueles que utilizam soluções centralizadas, e a segunda para aqueles que utilizam soluções distribuídas.

2.1 Definição do Problema

O PPR em RDC consiste basicamente em: dado um conjunto de servidores e requisições de clientes por conteúdos, posicionar as réplicas nos servidores de forma a minimizar os custos de comunicação, respeitando os recursos de banda e armazenamento disponíveis. Ou seja, com as informações das requisições por conteúdos e dos recursos dos servidores em cada período, réplicas são posicionadas nos servidores, de maneira que as requisições sejam satisfeitas com o menor custo de comunicação possível, bem como as replicações sejam realizadas minimizando o custo de replicação. É importante destacar que, tanto para o atendimento das requisições quanto para a replicação dos conteúdos, os recursos dos servidores (banda e armazenamento) não podem ser ultrapassados.

Para auxiliar a definição deste problema, a Figura 2.1 ilustra um exemplo simples. Na Figura 2.1a, tem-se os clientes 01, 02 e 03 com suas respectivas requisições de conteúdos no lado esquerdo e os servidores 01 e 02 com os seus conjuntos de réplicas inicialmente vazios do lado direito, ou seja, os dados de entrada para o problema. Na Figura 2.1b, tem-se os posicionamentos das réplicas dos conteúdos em cada servidor, de acordo com as requisições

dos clientes e limitações de recursos, e as atribuições (atendimentos) determinadas pela solução.

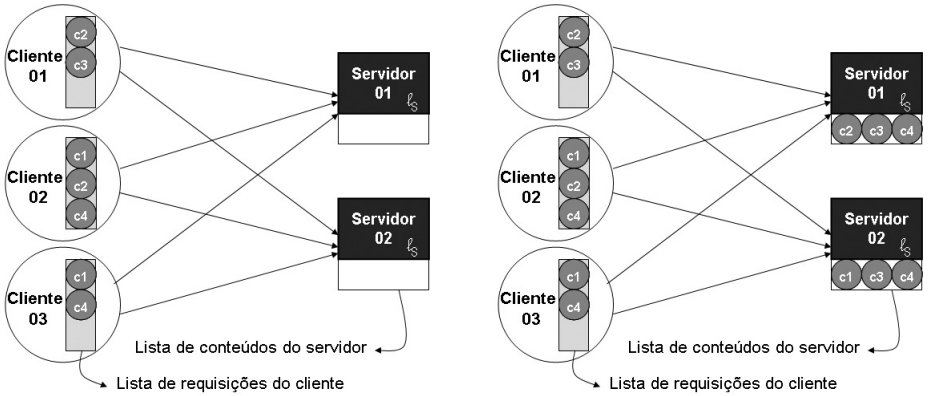


Figura 2.1: Exemplo do problema de posicionamento de réplicas

No PPR dinâmico, a cada período de tempo, surgem novas requisições de clientes. Se uma requisição não puder começar a ser atendida no seu período de chegada, esta será tratada no próximo período. Vale ressaltar que requisições não atendidas de períodos anteriores possuem prioridade maior que aquelas do período atual. Quando uma requisição começa a ser atendida por um servidor em um determinado período, esta continua sendo atendida pelo mesmo servidor nos períodos seguintes, até que seja completamente atendida. Desta forma, os recursos alocados pelos servidores para seu atendimento, continuam ocupados até o final do atendimento da requisição.

A arquitetura de RDC considerada é ilustrada na Figura 2.2. Nesta, é possível visualizar a hierarquia da RDC, onde os usuários finais se comunicam com os servidores de distribuição e os servidores de distribuição se comunicam entre si e com os servidores origem dos conteúdos. Neste trabalho é assumido, sem perda de generalidade, que as requisições dos clientes já foram direcionadas os servidores da RDC que estão mais próximos. Assim, estes servidores possuem armazenadas as requisições dos clientes mais próximos.

Duas abordagens também são consideradas para a versão On-Line (dinâmica) do problema de posicionamento de réplicas em servidores de uma RDC.

Na primeira, é assumido que: não existe penalização por atraso de atendimento de requisição (a requisição começa a ser atendida em um período posterior ao seu período de chegada), os custos de transportes entre os servidores não variam com o tempo e a replicação de um conteúdo é instantânea. Para esta abordagem, é proposto o algoritmo distribuído preliminar RPA, descrito na Seção 4.2. Para avaliar a qualidade das soluções obtidas através deste algoritmo, também são propostas a Formulação Matemática I, de-

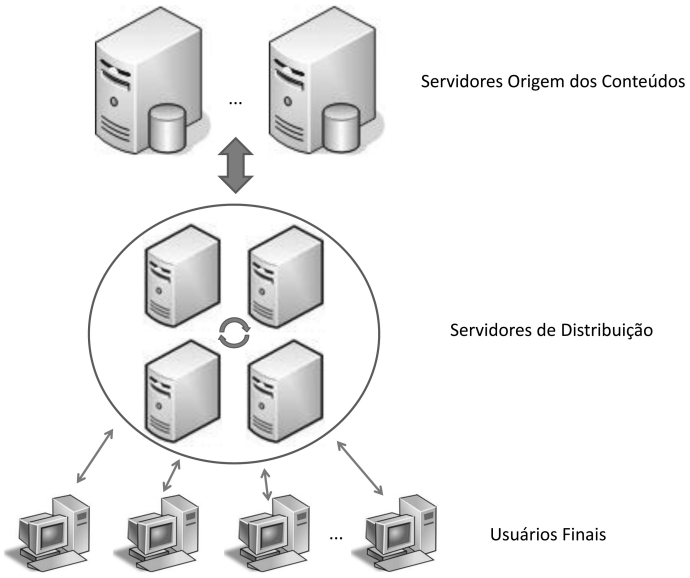


Figura 2.2: Arquitetura da RDC

scrita na Seção 3.1, e a heurística centralizada, apresentada na Seção 3.2.

Na segunda abordagem, com o objetivo de aproximação com a realidade, é assumido que: existe penalização por atraso de atendimento à requisições (*backlog*), os custos de transporte entre os servidores variam com o tempo (representando mudanças na rede), e as replicações dos conteúdos não são instantâneas (ou seja, duram um período de tempo). Para esta abordagem, foi realizado um aprimoramento e modificação do algoritmo preliminar, resultando no algoritmo distribuído RPA_Mod, descrito na Seção 4.3. Também para realizar uma avaliação deste algoritmo, é utilizada a Formulação Matemática II, proposta em [17] e exposta na Seção 3.3.

2.2 Algoritmos Centralizados

Embora o problema de posicionamento de réplicas seja tratado utilizando abordagem distribuída neste trabalho, nesta seção alguns trabalhos centralizados para este problema serão brevemente descritos. Ao final da seção, a Tabela 5.1 apresenta um resumo dos trabalhos comentados.

Em [5], o problema de posicionamento dinâmico de réplicas é abordado. Para a obtenção da solução ótima, este problema foi modelado como um processo de decisão de Markov[20] e resolvido com o MATLAB [16]. Entretanto, como este processo é computacionalmente caro, uma heurística também foi proposta. Os resultados apresentados mostraram que o comportamento da heurística é próximo do ótimo.

Em [3], é realizada um combinação de posicionamento de réplicas e técnicas de *caching* em RDC. Para isto, um modelo analítico e um algoritmo guloso híbrido (posicionamento de réplicas e *caching*) foram propostos. Os resultados encontrados mostraram que um esquema simples de *caching* sobre uma infraestrutura baseada em replicação pode melhorar o tempo de resposta das requisições.

Em [6], os problemas de posicionamento de réplicas e atribuição cliente a servidor são modelados através de uma formulação matemática inteira não-linear. Também foram propostos uma linearização para o modelo, um algoritmo para obtenção da solução exata (baseado em decomposição de Benders [7]) e uma heurística para o problema. Os resultados realizados em instâncias randômicas mostraram que a heurística proposta apresenta soluções próximas dos algoritmos de solução exata.

Em [25] é tratado o problema de replicação e posicionamento de réplicas de vídeos. Para estes problemas são propostos uma formulação matemática e algoritmos centralizados para replicação ótima, considerando posicionamento limitado, distribuição Zipf-Like e baseados em Simulated Annealing. Uma avaliação de desempenho dos algoritmos é realizada através de simulações sobre um conjunto de cargas de trabalho sintéticas.

Em [17], são propostas uma formulação matemática (FD) para a versão estática e heurísticas (HC, HCFK, PSH e OGHS) para versão dinâmica do problema de posicionamento de réplica e distribuição de requisições em uma RDC. Dentre estas heurísticas, duas (HCFK e PSH) foram usadas para justificar a diferença entre as soluções da formulação matemática FD e da heurística HC. Os resultados apresentados mostraram que a heurística HC apresenta boas soluções em tempo computacionais menores que a formulação FD e supera a heurística OGHS. A formulação matemática FD (aqui chamada de Formulação II) é descrita na Seção 3.3 e será utilizada para avaliação do algoritmo distribuído aqui proposto.

Tabela 2.1: Trabalhos Relacionados - Abordagens Centralizadas

Problema	Ano	Método Usado	Contribuição
Posicionamento dinâmico de réplicas em RDC. [5]	2003	Algoritmo Exato e Heurística Centralizados	Proposta de posicionamento ótimo baseado em modelo de decisão de Markov e uma heurística centralizada.
Posicionamento de réplica e caching em RDC. [3]	2005	Formulação Matemática e Heurística Centralizada	Proposta de formulação matemática para o problema e algoritmo híbrido (posicionamento de réplicas e *caching*) centralizado (guloso simples).
Posicionamento de réplicas e roteamento das requisições dos usuários para os servidores da RDC. [6]	2007	Formulação Matemática, Algoritmo Exato e Heurística Centralizados	Formulação matemática e proposta de uma heurística centralizada gulosa.
Replicação e posicionamento de réplicas de vídeos em clusters de armazenamento. [25]	2007	Formulação Matemática e Heurísticas Centralizadas	Formulação matemática e algoritmos centralizados para a replicação ótima, posicionamento limitado, distribuição Zipf-Like e baseado em Simulated Annealing.
Problema de posicionamento de réplica e distribuição de requisições em RDC. [17]	2010	Heurística Híbrida e Formulação Matemática	Proposta de uma formulação matemática para o problema Off-Line e de heurísticas para o problema On-Line.

2.3 Algoritmos Distribuídos

Nesta seção, são descritos alguns trabalhos relacionados que possuem abordagens distribuídas. No final desta seção, é apresentada a Tabela 5.5 com um resumo dos trabalhos relacionados.

Em [15], uma heurística distribuída simples é proposta para o PPR e PAC em RDC. Neste trabalho, cada requisição é tratada pelo servidor que a recebeu, chamado servidor origem. Se o servidor origem não possui o conteúdo requisitado, ao invés de encaminhar essa requisição para outro servidor que possua este conteúdo, este realiza a replicação do conteúdo localmente para então atender a requisição do usuário. Na ausência de recursos para a realização da replicação, é utilizado o esquema de *Least Recent Used (LRU)* para descartar réplicas, liberando recursos. Uma versão mais sofisticada desta heurística foi implementada em [17], apresentando resultados insatisfatórios em relação ao problema. Os algoritmos aqui propostos têm como objetivo posicionar as réplicas em servidores de modo a minimizar os custos totais de replicação e comunicação.

Em [12], uma RDC é projetada para prover acesso a conteúdos com restrição de tempo (latência). Para isto, é proposto um algoritmo distribuído baseado no Problema de Dominação em Grafos, que deve ser executado para cada classe de conteúdo (com a mesma restrição de tempo de acesso). Já os algoritmos aqui apresentados são executados apenas uma vez para todos os conteúdos.

Em [21], dois algoritmos, um centralizado e outro distribuído são apresentados para replicação de objetos em RDC, entretanto os algoritmos propostos são restritos a topologias em árvores. Os algoritmos propostos nesta dissertação consideram uma topologia de rede genérica.

Em [1], é considerado um modelo de RDC dinâmica e móvel, que considera variações de demandas para o posicionamento das réplicas, tentando fazer uma previsão das necessidades futuras para um número de passos seguintes. Entretanto, não são consideradas restrições de armazenamento e banda nos servidores. É considerado apenas um limite máximo de atendimentos a requisições externas.

Em [23, 24], são apresentados dois algoritmos distribuídos para o posicionamento de réplicas que podem ser adaptados para suportar balanceamento de carga. Entretanto, apenas a topologia de rede em anel é considerada. Já neste trabalho, como já mencionado anteriormente, não existe restrição quanto a topologia de rede utilizada. Além disso, o objetivo dos algoritmos propostos é minimizar os custos de replicação e transporte, e não realizar o balanceamento de carga no servidor central, como proposto neste trabalho relacionado.

Em [8], é apresentado um algoritmo híbrido para o posicionamento de réplicas para uma arquitetura de RDC auto-organizável. Este possui um módulo centralizado que fornece informações adicionais sobre o estado da rede (visão topológica global) para o módulo distribuído, diferentemente do trabalho aqui proposto.

Em [22, 19], uma heurística distribuída e um mecanismo de redirecionamento de tráfego são propostos em RDC para minimizar os custos de replicação. A heurística replica ou remove réplicas baseada em informações locais e de servidores vizinhos. Para o redirecionamento de tráfego, é resolvido um problema de otimização considerando o padrão de requisição de um servidor e a configuração das réplicas. Já o objetivo deste trabalho é, além de minimizar os custos de replicação, também minimizar os custos de comunicação dos atendimentos das requisições. Este trabalho é melhor descrito na Seção 5.3, onde uma comparação com o algoritmo aqui desenvolvido também é realizada. Este trabalho relacionado foi escolhido para comparação principalmente pelo fato de ser o mais

recente na bibliografia pesquisada. Além disso, as considerações assumidas também são próximas das aqui abordadas.

Em [11], algoritmos centralizados e distribuídos são propostos para solucionar o problema da auto-configuração (e.g., eleição de líder) em redes de sensores através do Problema de Localização de Facilidades (PLF). Entretanto, mesmo com o PPR podendo ser modelado também com um PLF, estes algoritmos não podem ser utilizados para resolver o PPR por duas razões principais: não consideram mais de um produto (conteúdo); e, também não consideram limitação de recursos nas facilidades (servidores).

Em [18], são descritos a estrutura e o funcionamento de uma das maiores empresas de distribuição de conteúdos, a Akamai. Devido a interesses comerciais, esta descrição não é profunda, mas muito importante para fornecer uma visão geral do serviço. O algoritmo de posicionamento adotado é bem simples: posicionar o mais perto possível do usuário. Como a rede e os servidores são proprietários, não existe preocupação em minimizar o número de réplicas existentes nem os dados trafegados (custos de replicação e transporte). Não há direcionamento de requisições, pois as requisições são sempre atendidas pelo servidor mais próximo (melhor servidor). A replicação do conteúdo é feita de forma eficiente e rápida através do sistema de transporte da Akamai, que utiliza uma rede sobreposta à Internet.

Tabela 2.2: Trabalhos Relacionados - Abordagens Distribuídas

Problema	Ano	Método Usado	Contribuição
Posicionamento de réplicas e atribuição cliente a servidor em RDC. [15]	2000	Heurística Distribuída	O algoritmo replica o conteúdo solicitado localmente. Se não existir recurso, o esquema de *Least Recent Used (LRU)* para descartar réplicas.
Rede de distribuição de conteúdo projetada para prover latência de acesso a conteúdo limitada. [12]	2004	Heurística Distribuída.	Problema de dominação em grafos distribuído aplicado a uma RDC.
Posicionamento de réplicas em RDC. [21]	2005	Heurística Centralizada e Distribuída.	Proposta de algoritmo centralizado guloso e distribuído para o problema.
Modelo de RDC dinâmica móvel que considera variações de demanda para as decisões de replicar um conteúdo e remover uma réplica, minimizando o tráfego total da rede.[1]	2005	Heurística Distribuída On-Line e Formulação Matemática.	Proposta de um algoritmo distribuído On-Line, que tenta fazer uma previsão das necessidades futuras para um número de passos seguintes.
Problema de posicionamento de réplica em RDC com balanceamento de carga. [23, 24]	2005 e 2006	Formulação Matemática e Heurísticas Distribuídas.	Proposta de adaptação de heurísticas distribuídas, apresentadas em trabalhos anteriores, para considerarem balanceamento de carga.
Problema de posicionamento de réplicas em servidores de uma RDC. [8]	2006	Heurística Distribuída Híbrida.	Proposta de uma heurística híbrida (centralizada e distribuída) para posicionar réplicas, adequado para uma arquitetura de RDC auto-organizável.
Posicionamento de réplicas e atribuição cliente a servidor em RDC. [22]	2007	Heurística Distribuída.	Proposta de heurística distribuída e mecanismo de redirecionamento de requisição (modelado como problema de otimização).
Problema de configuração própria em redes de sensores, onde nós da rede possuem diferentes funções para alcançar uma dada meta da aplicação. Ex. clustering (eleição de um nó líder do cluster). [11]	2007	Heurísticas Centralizada e Distribuída.	Proposta de uma heurística distribuída para o problema de localização de facilidades que pode ser implementada em redes de sensores multi-hop com comunicação local. Entretanto, não pode ser utilizado para posicionamento de réplicas em RDC.
Posicionamento de réplicas e atribuição cliente a servidor em RDC. [18]	2010	Heurística Distribuída.	Descrição da estrutura da RDC Akamai, explicando simplificadamente o funcionamento e os serviços fornecidos pela empresa.

Capítulo 3

Formulações Matemáticas e Heurística Centralizada

Neste capítulo são apresentadas duas formulações matemáticas e uma heurística centralizada para o problema, que serão utilizadas na comparação e avaliação dos algoritmos distribuídos propostos.

3.1 Formulação Matemática I

Para obtenção da solução ótima da versão On-Line do Problema de Posicionamento de Réplicas em servidores de uma RDC é proposta uma formulação matemática. Nesta, é considerado que uma requisição só pode ser atendida totalmente por um único servidor e que todas as requisições de todos os períodos de tempos são conhecidas a priori.

O motivo pelo qual esta formulação matemática foi proposta é a obtenção da solução ótima, utilizada na comparação e avaliação da qualidade da solução obtida pelo algoritmo RPA. A seguir, esta formulação é apresentada.

Conjuntos

S	Conjunto de servidores, indexado por s.
R	Conjunto de requisições, indexado por r.
C	Conjunto de conteúdos, indexado por k.

Dados

T	Conjunto $\{1, ..., T_f\}$ de períodos de tempo, indexado por t.
$c_{s_i s_j}$	Custo de transporte entre o servidor s_i e o servidor s_j (\$/Mb).
f_{sk}	Custo de replicação do conteúdo k no servidor s (\$/Mb).
AS_s	Capacidade de armazenamento do servidor s (Mb).
MB_s	Capacidade de banda do servidor s (Mb/t).
L_k	Tamanho do conteúdo k (Mb).
B_r	Banda necessária para a requisição r (Mb/t).
$g(r)$	Conteúdo da requisição r.
O_r	Servidor de origem da requisição r.
$Temp(r)$	Período da requisição r.
τ_r	$\lceil L_{g(r)}/B_{g(r)} \rceil$.
$Tmax(r)$	$T_f - \tau_r$

Variáveis

x_{skt}	Variável binária indicando se o conteúdo k foi replicado no servidor s no período t.
x'_{skt}	Variável binária indicando se o conteúdo k foi apagado do servidor s no período t.
w_{skt}	Variável binária indicando se o conteúdo k está replicado no no servidor s no período t.
y_{rst}	Variável binária indicando o período de inicio do atendimento da requisição r pelo servidor s no período t.

$$min \sum_{r \in R} \sum_{\substack{t \in T | \\ Temp(r) \leq t \\ t \leq Tmax(r)}} \sum_{s \in S} L_{g(r)} c_{O_r s} y_{rst} + \sum_{s \in S} \sum_{k \in C} \sum_{t \in T} L_k f_{sk} x_{skt} \tag{3.1}$$

satisfazendo:

$$\sum_{t=Temp(r)}^{Tmax(r)} \sum_{s \in S} y_{rst} = 1, \forall r \in R \tag{3.2}$$

$$w_{skt} = \sum_{t'=0}^{t} (x_{skt'} - x'_{skt'}), \forall s \in S, \forall k \in C, \forall t \in T \tag{3.3}$$

$$y_{rst} \leq w_{skt} - \sum_{t'=t}^{t+\tau_r} x'_{skt}, \forall r \in R, \forall s \in S,$$

$$\forall t \in T | Temp(r) \leq t \leq Tmax(r), k = cont(r) \tag{3.4}$$

$$\sum_{k \in C} w_{skt}.L_k \leq AS_s, \forall s \in S, \forall t \in T \tag{3.5}$$

$$\sum_{r \in R} \sum_{\substack{t'=max(0,\\ t-\tau_r+1)}}^{t} y_{rst'}.B_{g(r)} \leq MB_s, \forall s \in S, \forall t \in T, \tag{3.6}$$

$$x_{skt} \in \{0,1\}, \forall s \in S, \forall k \in C, \forall t \in T \tag{3.7}$$

$$y_{rst} \in \{0,1\}, \forall r \in R, \forall s \in S, \forall t \in T | Temp(r) \leq t \leq Tmax(r) \tag{3.8}$$

A função objetivo (3.1) minimiza o custo de transporte dos atendimentos às requisições e o custo de instalação dos conteúdos nos servidores da RDC. O grupo de restrições (3.2) garante que as requisições sejam atendidas em um intervalo de tempo válido. O conjunto de restrições (3.3) garante que as variáveis w_{skt} possuam valor 1, caso o conteúdo k esteja replicado no servidor s no período t, e 0, caso contrário. As restrições representadas em (3.4) garantem que as requisições sejam atendidas por servidores que possuem o conteúdo solicitado durante todos os períodos de atendimento. O conjunto de restrições (3.5) garante que os conteúdos replicados nos servidores em um dado período de tempo não ultrapassem a capacidade de armazenamento dos mesmos. O grupo de restrições (3.6) garante que a banda utilizada pelos atendimentos nos servidores não ultrapassem a capacidade de banda dos mesmos.

Note que a formulação matemática considera o conhecimento das requisições de todos os períodos de tempo, o que não ocorre na prática.

3.2 Heurística Ótima por Período

Também foi proposta uma heurística centralizada que utiliza uma versão simplificada da formulação acima descrita para encontrar a solução ótima por período. Esta foi criada para analisar a solução da formulação matemática, limitando o conhecimento das requisições por período. Assim, esta heurística também é utilizada para comparar e avaliar a solução obtida através do algoritmo RPA. Assim, é possível observar se a distância (gap) entre a solução ótima e a solução obtida pelo RPA é devido ao conhecimento de todas as requisições em todos os períodos de tempo ou a uma baixa qualidade do algoritmo proposto.

Desta forma, para cada período foi executado o modelo simplificado considerando apenas as requisições deste período. O controle dos recursos dos servidores foi realizado através dos dados de entrada.

Assim, quando uma requisição começa a ser atendida em um período, para os próximos períodos a banda utilizada por este atendimento é diminuída da banda total do servidor. Esta banda volta a ser somada novamente quando o atendimento é finalizado.

Quando um conteúdo é replicado em um dado período, nos períodos seguintes o custo de replicação deste conteúdo é zerado e é adicionada uma restrição forçando sua replicação, caso esteja em andamento o atendimento de alguma requisição que o utiliza neste servidor.

Como nas instâncias utilizadas nos testes o custo de replicação é significativamente maior que o custo de transporte, foi adicionada uma previsão considerando que nos próximos cinco períodos de tempo existirão requisições do mesmo conteúdo (este valor foi escolhido baseado na ocorrência das requisições nas instâncias utilizadas). Isto foi necessário pois se o modelo considerasse apenas as requisições do período em questão, sempre seria mais vantajoso atendê-las em outros servidores pagando o custo de transporte, do que replicar o conteúdo localmente e atendê-las com custo de transporte zero.

3.3 Formulação Matemática II

Com a consideração de novas características de RDCs reais no algoritmo distribuído RPA_Mod (proposto neste trabalho e descrito na Seção 4.3), a Formulação Matemática I não pode ser mais utilizada para comparação e avaliação da solução distribuída. Desta forma, uma outra formulação e uma nova comparação se fazem necessárias. Assim, nesta seção é descrita a formulação matemática proposta no trabalho [17] e que contempla estas novas características adicionadas.

Para facilitar o entendimento e a explicação desta formulação, a mesma será apresentada a seguir.

Conjuntos	
R	Conjunto de requisições, indexado por i.
S	Conjunto de servidores, indexado por j e l.
C	Conjunto de conteúdos, indexado por k.
T	Conjunto $\{1, ..., T_f\}$ de períodos de tempo, indexado por t.

Dados

L_k	Tamanho do conteúdo k.
B_k	Período em que o conteúdo k é disponibilizado na RDC.
E_k	Período em que o conteúdo k é removido da RDC.
O_k	Servidor origem do conteúdo k.
AS_j	Espaço em disco disponível no servidor j.
MB_j	Banda máxima disponível do servidor j.
D_{it}	Demanda da requisição i no período t.
BR_i	Banda mínima exigida pela requisição i.
BX_i	Banda máxima aceita pela requisição i.
$g(i)$	Conteúdo solicitado pela requisição i.
c_{ijt}	Custo de atendimento da requisição i no servidor j, no período t.
p_{it}	Penalidade por usar *backlog* da requisição i no período t.

Variáveis

x_{ijt}	Variável contínua indicando a fração do conteúdo solicitado pela requisição i entregue pelo servidor j no período t.
y_{kjt}	Variável binária indicando se o conteúdo k está replicado no servidor j no período t.
b_{it}	Variável inteira indicando o *backlog* da requisição i no período t.
w_{kjlt}	Variável binária indicando se o conteúdo k é copiado pelo servidor j a partir do servidor l no período t.

$$min \sum_{i\in R}\sum_{j\in S}\sum_{t\in T} c_{ijt}x_{ijt} + \sum_{r\in R}\sum_{t\in T} p_{it}b_{it} + \sum_{k\in C}\sum_{j\in S}\sum_{l\in S}\sum_{t\in T} L_k w_{kjlt} \tag{3.9}$$

satisfazendo:

$$\sum_{j\in S} L_{g(i)}x_{ijt} - b_{i(t-)} + b_{it} = D_{it}, \forall i \in R, \forall t \in [B_{g(i)}, E_{g(i)}] \tag{3.10}$$

$$\sum_{i\in R} L_{g(i)}x_{ijt} \leqslant MB_j, \forall j \in S, \forall t \in T \tag{3.11}$$

$$\sum_{j\in S} L_{g(i)}x_{ijt} \leqslant BX_i, \forall i \in R, \forall t \in T \tag{3.12}$$

$$\sum_{j\in S}\sum_{t\in T} x_{ijt} = 1, \forall i \in R \tag{3.13}$$

$$y_{g(i)jt} \geqslant x_{ijt}, \forall i \in R, \forall j \in S, \forall t \in T \tag{3.14}$$

$$\sum_{j \in S} y_{kjt} \geqslant 1, \forall k \in C, \forall t \in [B_k, E_k] \tag{3.15}$$

$$y_{kjt} = 0, \forall k \in C, \forall j \in S, \forall t \notin [B_k, E_k] \tag{3.16}$$

$$y_{kO_kB_k} = 1, \forall k \in C \tag{3.17}$$

$$y_{kjB_k} = 0, \forall k \in C, \forall j \in \{S | j \neq O_k\} \tag{3.18}$$

$$y_{kj(t+1)} \leqslant \sum_{l \in S} w_{kjlt}, \forall k \in C, \forall j \in S, \forall t \in T \tag{3.19}$$

$$y_{kjt} \geqslant w_{kjlt}, \forall k \in C, \forall j \in S, \forall l \in S, \forall t \in T \tag{3.20}$$

$$\sum_{k \in C} L_k y_{kjt} \leqslant AS_j, \forall j \in S, \forall t \in T \tag{3.21}$$

$$x_{ijt} \in [0, 1], \forall i \in R, \forall j \in S, \forall t \in T \tag{3.22}$$

$$y_{kjt} \in \{0, 1\}, \forall k \in C, \forall j \in S, \forall t \in T \tag{3.23}$$

$$b_{it} \geqslant 0, \forall i \in R, \forall t \in T \tag{3.24}$$

$$w_{ijlt} \in \{0, 1\}, \forall j, l \in S, \forall k \in C \forall t \in T \tag{3.25}$$

A função objetivo (3.9) minimiza o custo de atendimento das requisições, o número de *backlogs* e replicações realizados. As restrições (3.10) garantem que o atendimento no período atual mais o *backlog* para o próximo período é igual a demanda do período atual mais o *backlog* do período anterior. O conjunto de restrições (3.11) garante que a banda máxima dos servidores não seja ultrapassada. As restrições (3.12) garantem que a banda suportada pelos clientes não seja ultrapassada. O conjunto de restrições (3.13) garante que uma requisição seja completamente atendida. As restrições (3.14) garantem que uma requisição seja atendida por um servidor que possua o conteúdo em questão. Os conjuntos de restrições (3.15) e (3.16) garantem a existência de uma réplica de cada conteúdo durante o seu tempo de vida e que nenhuma réplica pode existir fora do seu tempo de vida. As restrições (3.17) e (3.18) garantem que no período de surgimento de um conteúdo, apenas o servidor origem deste conteúdo possua uma réplica. O conjunto de restrições (3.19) garante que toda replicação crie uma nova réplica. As restrições (3.20) garantem que uma replicação ocorra a partir de um servidor que possua o conteúdo. As restrições (3.21) garantem que a capacidade de armazenamento dos servidores não seja ultrapassada.

Capítulo 4

Algoritmos Distribuídos

Neste capítulo, primeiramente é definido, na Seção 4.1, o sistema distribuído considerado neste trabalho. Em seguida, são apresentados os algoritmos distribuídos propostos para a versão On-Line do Problema de Posicionamento de Réplicas em RDC. Como mencionado na Seção 2.1, duas abordagens são consideradas para o problema em questão. Desta forma, um algoritmo distribuído preliminar foi desenvolvido para a primeira abordagem, sendo este aprimorado e modificado para a segunda abordagem. A Seção 4.2 descreve o algoritmo distribuído preliminar RPA, e a Seção 4.3, o algoritmo distribuído aprimorado RPA_Mod.

4.1 Sistema Distribuído

O sistema distribuído assumido neste trabalho é assíncrono e definido como segue:

- Cada servidor da RDC consiste em um nó de processamento.
- Os canais de comunicação entre os servidores são confiáveis e FIFO (*First In, First Out*). Ou seja, toda mensagem enviada é recebida e uma mensagem enviada após uma outra em um mesmo canal não pode chegar antes da enviada anteriormente.
- O tempo local dos servidores é dividido em passos, chamados de períodos. Inicialmente, cada servidor começa a tratar as requisições que já estão armazenadas localmente (período p). Quando os servidores acabam de tratar todas as requisições originadas no período p, este então passa para o próximo período $(p + 1)$. Entretanto, durante o processo de atendimento das requisições originadas no período p, novas requisições podem surgir. Estas novas requisições serão armazenadas para

que sejam tratadas no próximo período (no caso, $p+1$).

- As requisições dos clientes surgem com o passar do tempo em todos os servidores.

Para realizar as comparações descritas no Capítulo 5 de maneira justa, os algoritmos distribuídos propostos foram simulados de maneira síncrona. Desse modo, ao invés de tempo local, foi considerado tempo global. Assim, todos os servidores estão sempre no mesmo período, e passam para o próximo quando o último servidor acabar de tentar tratar todas as sua requisições para aquele período.

Como trabalho futuro, tem-se o objetivo de simular os algoritmos distribuídos de maneira assíncrona, utilizando um servidor (processo) adicional para gerar e distribuir as requisições dos clientes. Dessa forma, para realizar as comparações descritas no Capítulo 5 de maneira justa, não existirá mais a necessidade de sincronizar os algoritmos distribuídos.

4.2 Algoritmo Distribuído RPA

Nesta seção é apresentado o algoritmo distribuído RPA guloso proposto para a primeira abordagem do PPR On-Line em RDC e sua análise de complexidade.

A abordagem distribuída é utilizada uma vez que a natureza do problema tratado é distribuída e dinâmica. Dessa forma, como os dados do problema estão distribuídos geograficamente, para resolvê-lo de forma centralizada seria necessário coletar os dados, resolver o problema, e distribuir o resultado a cada período tempo. Note que as operações de coleta de dados e distribuição da solução adicionam um alto custo e tempo a resolução do problema.

4.2.1 Procedimentos

Os procedimentos do algoritmo distribuído RPA, bem como seu funcionamento são descritos a seguir.

Estruturas de dados utilizadas:

meu_id	id do servidor.
$numServidores$	número de servidores.
$numConteudos$	número de conteúdos.
$numPeriodos$	número de períodos.
$conteudos$	conteúdos replicados localmente.
$tamCont_c$	tamanho do conteúdo c.
$numReq_t$	número de requisições locais do período t.
$contReq_{rt}$	conteúdo da r-ésima requisição do período t.
$bandReq_{rt}$	banda necessária da r-ésima requisição do período t.
$tamReq_{rt}$	tamanho do conteúdo da r-ésima requisição do período t.
$iniReq_{rt}$	início de atendimento da r-ésima requisição do período t.
$fimReq_{rt}$	término de atendimento da r-ésima requisição do período t.
$servReq_{rt}$	servidor que atendeu a r-ésima requisição do período t.
$numReqExt_t$	número de requisições externas atendidas no período t.
$contReqExt_{rt}$	conteúdo da r-ésima requisição externa do período t.
$bandReqExt_{rt}$	banda necessária da r-ésima requisição externa do período t.
$tamReqExt_{rt}$	tamanho do conteúdo da r-ésima requisição externa do período t.
$iniReqExt_{rt}$	início de atendimento da r-ésima requisição externa do período t.
$fimReqExt_{rt}$	término de atendimento da r-ésima requisição externa do período t.
$custoReplicacao_k$	custo de replicação do conteúdo k.
$custoTransporte_{s_i s_j}$	custo de transporte entre os servidores s_i e s_j.
$capTotal$	capacidade local disponível.
$bandaTotal$	banda local disponível.
$custoReplicacaoTotal$	custo de replicação local total.
$custoTransporteTotal$	custo de transporte local total.

Mensagens enviadas:

STATIC(requisições não atendidas, bandaTotal', capTotal')	Enviada de um servidor s para um servidor central com o conjunto de suas requisições não atendidas e seus recursos de banda e armazenamento.
STATIC_RESP(requisições locais atendidas, requisições externas atendidas, conteúdos, bandaTotal', capTotal')	Enviada por um servidor central para um servidor s com o conjunto de requisições locais atendidas, o conjunto de requisições externas atendidas, o conjunto de conteúdos replicados localmente e os recursos de banda e armazenamento.
CHECK(requisições não atendidas do conteúdo contMsg)	Enviada por um servidor s para um servidor s' com o conjunto de requisições do conteúdo $contMsg$ não atendidas.
LIBERA_BANDA(banda)	Enviada por um servidor s para um servidor s' com a banda que deve ser adicionada à banda total de s'.
CHECKED(requisições atendidas e não atendidas do conteúdo contMsg)	Enviada por um servidor s para um servidor s' com os conjuntos de requisições atendidas e não atendidas do conteúdo $contMsg$.
ATTEND_CENTRAL(requisições não atendidas)	Enviada por um servidor s para um servidor central com o conjunto de requisições que não puderam ser atendidas.

O algoritmo se inicia com cada servidor tentando atender localmente suas requisições do primeiro período (Algoritmos 1 e 2). Para isso, é calculada uma ordem de replicação de conteúdos através de uma função de lucro (custo replicação / número de requisições). Possíveis requisições não atendidas são enviadas para o servidor central (servidor origem dos conteúdos) juntamente com os recursos disponíveis localmente. Sem perda de generalidade e para fins de simplificação, neste trabalho é considerado um único servidor origem para todos os conteúdos.

O servidor central, ao receber as mensagens $STATIC$ de cada servidor (Algoritmo 3), primeiramente armazena localmente as informações recebidas. Após receber de todos

Algoritmo 1: Inicialização /* Executado por um servidor s */

```
1 periodo := 0;
2 atenderLocal(periodo);  /* Algoritmo 2*/
3 Enviar STATIC(requisições não atendidas, bandaTotal', capTotal') para o servidor central;
```

Algoritmo 2: atenderLocal(periodo) */* *Executado por um servidor s* */*

```
1  para p:=0 até período faça
2      para cada requisição r do período p em ordem crescente da função de lucro faça
3          se a requisição r ainda não foi atendida então
4              se contReq_rt está replicado localmente então
5                  se bandaTotal ⩾ bandReq_rt então
6                      Marcar a requisição r do período p como atendida;
7                      iniReq_rt := p;
8                      fimReq_rt := p + ⌈tamReq_rt/bandReq_rt⌉;
9                      servReq_rt := meu_id;
10                     bandaTotal := bandaTotal − bandReq_rt;
11             senão
12                 se capTotal ⩾ tamReq_rt e bandaTotal ⩾ bandReq_rt então
13                     Replicar localmente o conteúdo contReq_rt;
14                     iniReq_rt := p;
15                     fimReq_rt := p + ⌈tamReq_rt/bandReq_rt⌉;
16                     servReq_rt := meu_id;
17                     bandaTotal := bandaTotal − bandReq_rt;
18                     capTotal := capTotal − tamReq_rt;
19                     custoReplicacaoTotal := custoReplicacaoTotal + custoReplicacao_contReq_rt;
```

os servidores, o servidor central determina o direcionamento de requisições para serem atendidas externamente por outros servidores, e também determina possíveis replicações de conteúdos nos servidores (Algoritmo 4). Por fim, o servidor central envia mensagens $STATIC_RESP$ de volta para cada servidor, informando as requisições locais que serão atendidas, as requisições de outros servidores que atenderá, os conteúdos que serão replicados e recursos disponíveis atualizados do servidor em questão.

Algoritmo 3: Ao receber uma mensagem *STATIC(requisições não atendidas, bandaTotal',capTotal')* do servidor s */* *Executado por um servidor central* */*

```
1 Armazenar informações recebidas localmente do servidor s;
2 se Recebeu STATIC de todos os servidores então
3     verificaAtendimentos();  /* Algoritmo 4 */
4     para cada servidor s faça
5         Enviar STATIC_RESP(requisições locais atendidas,requisições externas
          atendidas, conteudos, bandaTotal', capTotal') para o servidor s;
```

Os servidores, ao receberem a mensagem $STATIC_RESP$ do servidor central (Algoritmo 5), atualizam as informações recebidas sobre as suas requisições que estão sendo atendidas localmente ou externamente, as requisições de outros servidores que estão sendo atendidas localmente e suas capacidades de armazenamento e banda (Algoritmo 6). Depois disso, os servidores passam para o próximo período de tempo. Assim, estes tentam

Algoritmo 4: verificaAtendimentos() */* Executado por um servidor central */*

```
1  para cada servidor s faça
2      para cada requisição não atendida r de s faça
3          se ∃ um ou mais servidores que possuem o conteúdo requisitado e recursos então
4              Atribuir a requisição r para o servidor j com menor custo de transporte para s;
5              Atualizar os recursos disponíveis em j;
6          senão
7              Replicar o conteúdo solicitado no servidor j com menor custo de transporte para s e
               com recursos suficientes (capacidade e banda);
8              Atribuir a requisição r para o servidor j;
9              Atualizar os recursos disponíveis em j;
```

Algoritmo 5: Ao receber uma mensagem $STATIC_RESP$(*requisições locais atendidas*, *requisições externas atendidas, conteúdos, bandaTotal', capTotal')* do servidor central */* Executado por um servidor s */*

```
1  atualizaLocal();   /* Algoritmo 6 */
2  periodo := periodo + 1;
3  atenderLocal(periodo);   /* Algoritmo 2*/
4  contMsg := 0;   /* primeiro conteúdo a ser tratado */
5  Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor com menor
   custo de transporte e que possua o conteúdo contMsg;
```

Algoritmo 6: *atualizaLocal()* */* Executado por um servidor s_j */*

```
1  para cada requisição local atendida r faça
2      Marcar a requisição r do período p como atendida;
3      iniReq_rt := periodo;
4      fimReq_rt := periodo + ⌈tamReq_rt/bandReq_rt⌉;
5      servReq_rt := id do servidor externo;
6  se requisições externas atendidas ≠ NIL então
7      capTotal := capTotal';
8      bandaTotal := bandaTotal';
9      para cada requisição externa atendida r faça
10         iniReqExt_rt := periodo;
11         fimReqExt_rt := periodo + ⌈tamReq_rt/bandReq_rt⌉;
12 para cada período t' ≤ periodo faça
13     para cada requisição atendida localmente r do período t' faça
14         se fimReq_rt' = periodo então
15             se servReq_rt' = meu_id então
16                 bandaTotal := bandaTotal + bandReq_rt';
17             senão
18                 Enviar LIBERA_BANDA(bandReq_rt') para o servidor servReq_rt';
                   /*Algoritmo 7*/
```

Algoritmo 7: Ao receber uma mensagem $LIBERA_BANDA$(*banda*) do servidor s */* Executado por um servidor s_j */*

```
1  bandaTotal := bandaTotal + banda;
```

atender localmente requisições que não foram atendidas nos períodos anteriores mais as requisições que surgiram neste período (Algoritmo 2). As possíveis requisições não atendidas não são mais enviadas para o servidor central. Estas são direcionadas para outros servidores através da mensagem $CHECK$, em ordem crescente de custo de transporte, que possuam o conteúdo solicitado pelas requisições (final do Algoritmo 5). Para isso, pressupõe-se que a RDC fornece um serviço de localização de conteúdos, similar ao *DNS*.

Algoritmo 8: Ao receber uma mensagem $CHECK$(*requisições não atendidas do conteúdo contMsg*) do servidor s /* *Executado por um servidor* s_j */

1 *atendeExternas*(); /*Algoritmo 9*/
2 Enviar $CHECKED$(*requisições atendidas e não atendidas do conteúdo contMsg*) para o servidor s;

Um servidor ao receber mensagem $CHECK$ (Algoritmo 8) verifica se é possível atender as requisições do servidor remetente através do Algoritmo 9. Assim, este verifica se possui banda disponível para o atendimento e marca as requisições que puderam ou não ser atendidas. Por fim, o servidor responde de volta informando as requisições que puderam ser atendidas e as que não puderam, através da mensagem $CHECKED$.

Um servidor ao receber mensagem $CHECKED$ (Algoritmo 10) atualiza as informações locais (Algoritmo 6) e verifica se o servidor para o qual as requisições foram direcionadas atendeu todas as requisições. Em caso afirmativo, as requisições do próximo conteúdo ou do primeiro conteúdo do próximo período serão tratadas (linhas 2 a 11 do Algoritmo 10). Entretanto, em caso negativo, estas são redirecionadas para o próximo servidor com menor custo de transporte. Se todos os servidores forem percorridos e nenhum puder atender todas as requisições, estas ficam para o próximo período de tempo. Se as requisições não conseguirem ser atendidas nos próximos períodos (definido como parâmetro), estas serão direcionadas para o servidor central com um custo mais alto, através da mensagem $ATTEND_CENTRAL$ (linhas 13 a 23 do Algoritmo 10).

Quando o Algoritmo 6 é chamado dentro do Algoritmo 10, as ações das linhas de 7 a 11 não são realizadas, pois o conjunto de *requisições externas atendidas* é igual a NIL. Entretanto, é verificado se alguma requisição já acabou de ser atendida, sendo então a banda utilizada para esse atendimento liberada. Se a requisição foi atendida localmente, basta atualizar a banda do servidor. Mas se a requisição foi atendida por outro servidor, uma mensagem $LIBERA_BANDA$ é enviada.

Um servidor ao receber uma mensagem $LIBERA_BANDA$ (Algoritmo 7), adiciona a banda recebida na mensagem em sua *bandaTotal*.

Algoritmo 9: *atendeExternas()* */* Executado por um servidor s_j */*

```
1  para cada requisição não atendida r do período t' recebida faça
2      se bandaTotal ≥ bandReq_{rt'} então
3          Inserir a requisição r do período t' em requisições atendidas;
4          iniReqExt_{rt'} := periodo;
5          fimReqExt_{rt'} := periodo + ⌈tamReq_{rt'}/bandReq_{rt'}⌉;
6      senão
7          Inserir a requisição r do período t' em requisições não atendidas;
```

Algoritmo 10: Ao receber uma mensagem $CHECKED$(*requisições atendidas e não atendidas do conteúdo contMsg*) do servidor s */* Executado por um servidor s_j */*

```
1  atualizaLocal();/*Algoritmo 6*/
2  se Todas as requisições do conteúdo contMsg foram atendidas então
3      contMsg := contMsg + 1;
4      se contMsg < numConteudos então
5          Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor com
           menor custo de transporte e que possua o conteúdo contMsg;
6      senão
7          periodo := periodo + 1;
8          se periodo < numPeriodos então
9              atenderLocal(periodo);
10             contMsg := 0;
11             Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor
               com menor custo de transporte e que possua o conteúdo contMsg;
12 senão
13     se ∃ servidor não percorrido para o conteúdo contMsg então
14         Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o próximo
           servidor com menor custo de transporte e que possua o conteúdo contMsg;
15     senão
16         periodo := periodo + 1;
17         se periodo < numPeriodos então
18             atenderLocal(periodo);
19             contMsg := 0;
20             Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor
               com menor custo de transporte e que possua o conteúdo contMsg;
21         senão
22             /* Requisições locais não foram atendidas */
23             Enviar ATTEND_CENTRAL(requisições não atendidas) para o servidor central;
```

Algoritmo 11: Ao receber uma mensagem $ATTEND_CENTRAL$(*requisições não atendidas*) do servidor s */* Executado por um servidor central */*

```
1  para cada requisições não atendidas r faça
2      Atender a requisição r;
3      Adicionar custo de transporte do servidor central;
```

O servidor central ao receber mensagem $ATTEND_CENTRAL$ (Algoritmo 11), terá que atender as requisições do servidor remetente.

4.2.2 Exemplo

Nesta seção, é descrita uma possível execução do algoritmo distribuído RPA, ilustrada com figuras para facilitar o entendimento. Na Figura 4.1 é definido o significado de cada campo das figuras utilizadas neste exemplo. Os campos B e C dos servidores representam, respectivamente, a banda e capacidades disponíveis. Os campos *Conteúdos* e *Requisições* representam, respectivamente, os conteúdos replicados localmente e as requisições locais originadas no período definido em *Período*.

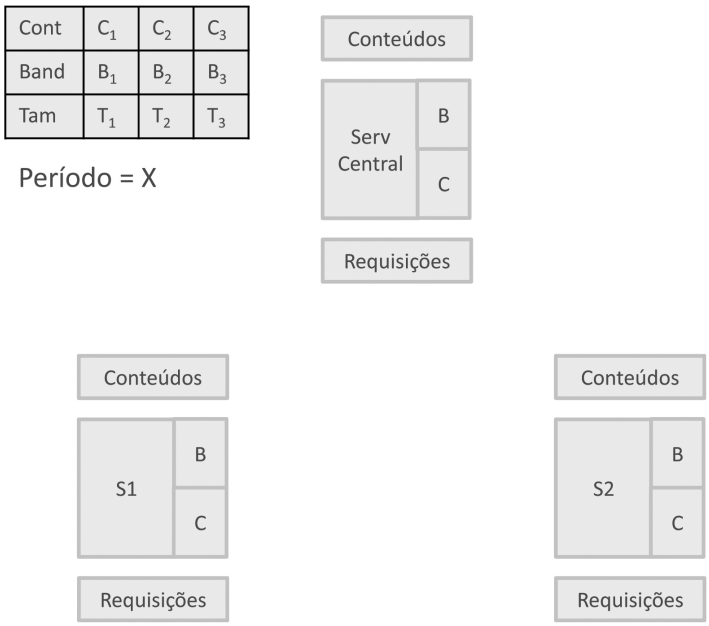


Figura 4.1: Dados do problema.

Este exemplo contém o seguinte cenário ilustrado na Figura 4.2:

- Dois servidores de distribuição da RDC (S_1 e S_2), com 50 e 70 unidades de banda e 25 e 30 unidades de armazenamento respectivamente.
- Um servidor central (*ServCentral*) com todos os conteúdos replicados e recursos de banda e capacidade infinitos e sem nenhuma requisição de conteúdos.
- Três conteúdos (1,2 e 3) presentes na RDC.
- As requisições são apresentadas na notação R_c, onde c define o conteúdo solicitado.

- Na tabela exposta no canto superior esquerdo, estão as informações de banda e tamanho necessários para cada conteúdo.

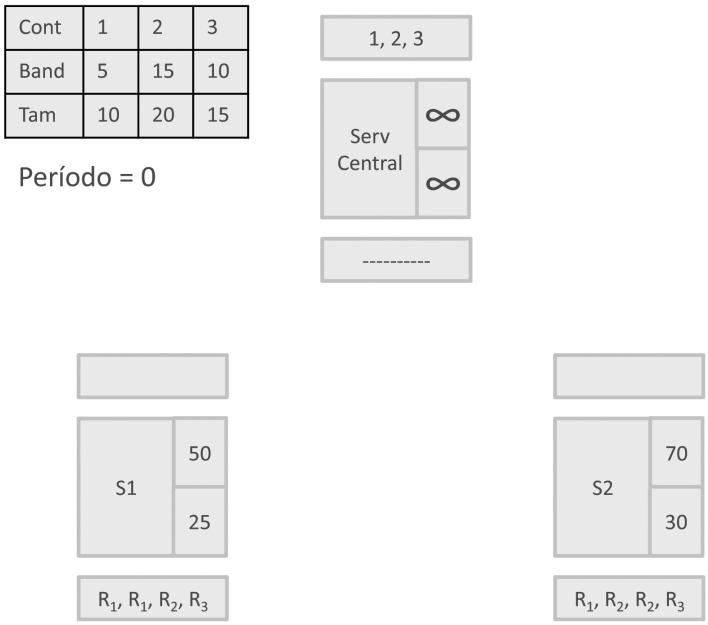


Figura 4.2: Um configuração inicial.

Inicialmente, cada servidor tenta atender suas requisições locais, replicando os conteúdos em ordem da função de lucro. Desta forma, na Figura 4.3, pode-se observar que o servidor S_1 replica os conteúdos 1 e 3 e atende localmente as duas requisições R_1 e a requisição R_3. Entretanto, este não consegue atender a requisição R_2, pois mesmo possuindo a banda necessária, este não possui capacidade de armazenamento para replicar o conteúdo 2. Já o servidor S_2 replica os conteúdos 1 e 2 e atende localmente a requisições R_1 e as duas requisições R_2. De maneira similar ao servidor S_1, o servidor S_2 não consegue atender a requisição R_3 por falta de capacidade de armazenamento para o conteúdo 3.

Em seguida, na Figura 4.4, cada servidor envia uma mensagem $STATIC$ para o servidor central, com as requisições não atendidas localmente e os recursos de banda e capacidade disponíveis.

O servidor central, ao receber mensagem $STATIC$ dos dois servidores, determina os direcionamentos e replicações necessários, enviando um outra mensagem $STATIC$ para cada servidor. Nesta mensagem estão as requisições locais que foram atendidas por outros servidores, as requisições externas atendidas localmente e os recursos de banda e capacidade atualizados (Figura 4.5). Os servidores ao receberem esta mensagem, atualizam suas informações e passam para o próximo período de tempo.

No período de tempo seguinte, novas requisições surgem em cada servidor (Figura

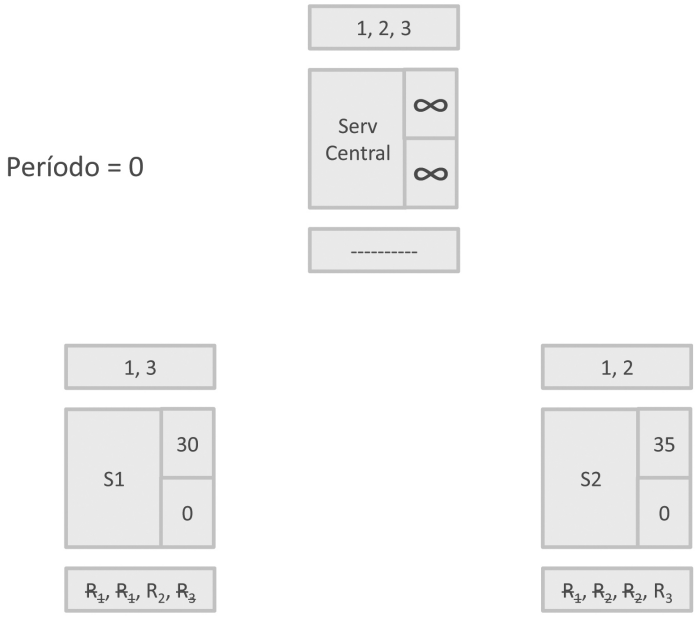


Figura 4.3: Cada servidor tenta atender as requisições localmente.

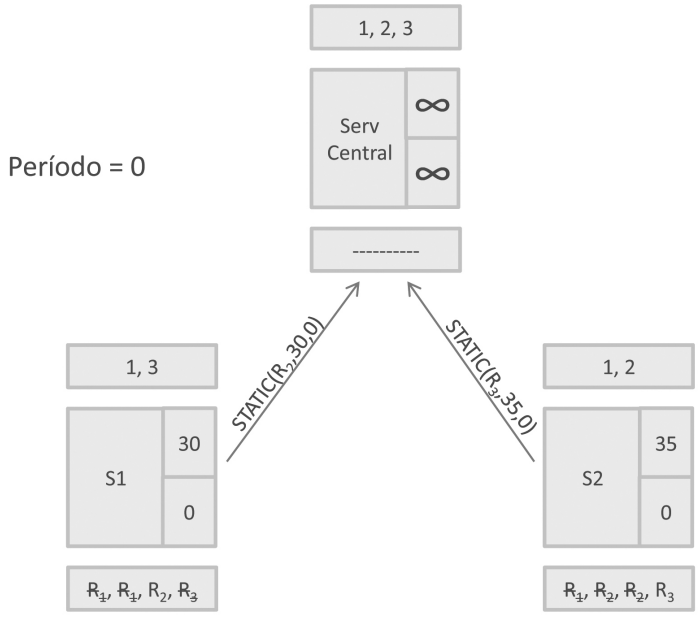


Figura 4.4: Cada servidor envia $STATIC$ com as requisições não atendidas e seus recursos disponíveis.

4.6).

Novamente, cada servidor tenta atender localmente suas novas requisições. Na Figura 4.7, pode-se observar que o servidor S_1 consegue atender localmente as duas requisições R_1 e o servidor S_2, a requisição R_1.

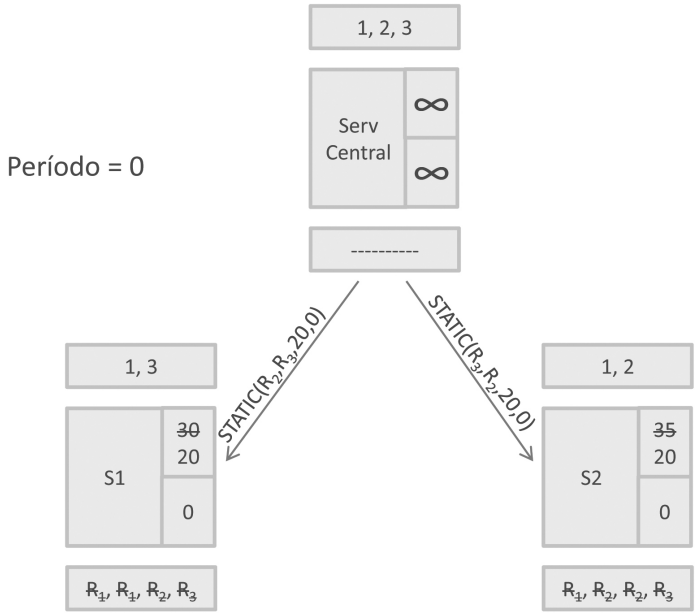


Figura 4.5: O servidor central envia $STATIC$ para os servidores.

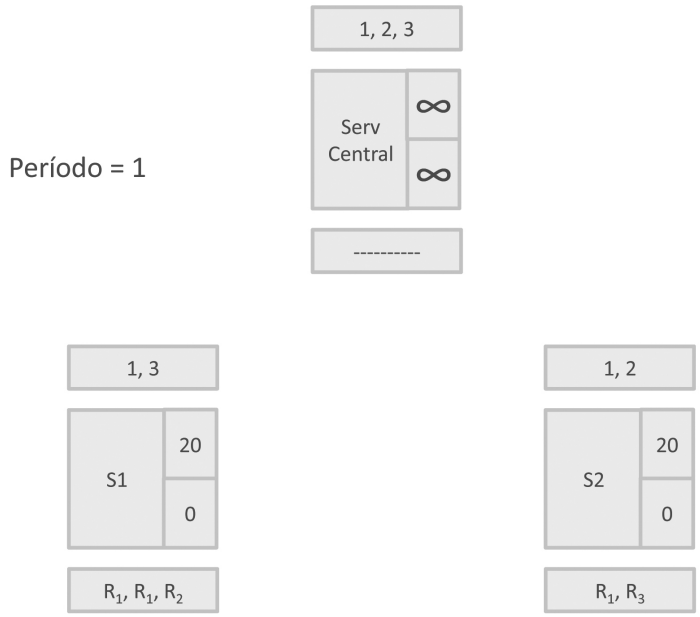


Figura 4.6: Novas requisições surgem no período de tempo seguinte.

Em seguida, cada servidor direciona suas requisições não atendidas diretamente para o servidor mais próximo que possua o conteúdo solicitado replicado através da mensagem $CHECK$. Desta forma, na Figura 4.8, o servidos S_1 envia $CHECK(R_2)$ para o servidor S_2 e o servidor S_2 envia $CHECK(R_3)$ para o servidor S_1.

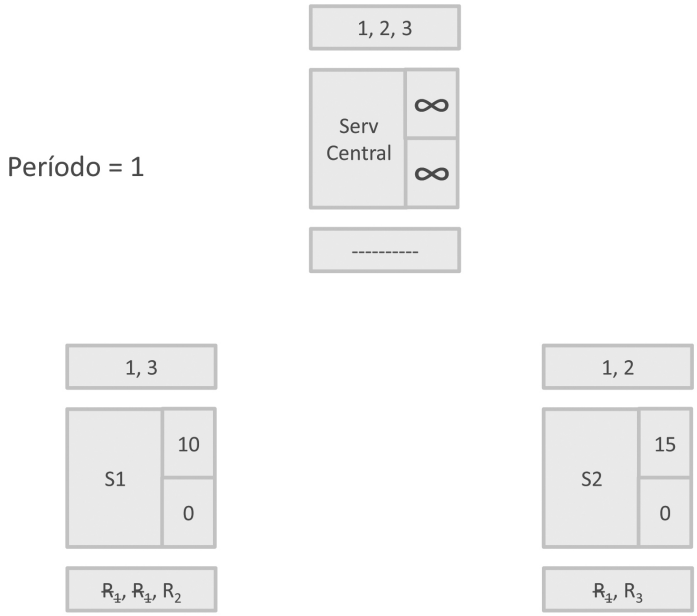


Figura 4.7: Cada servidor tenta atender as requisições localmente.

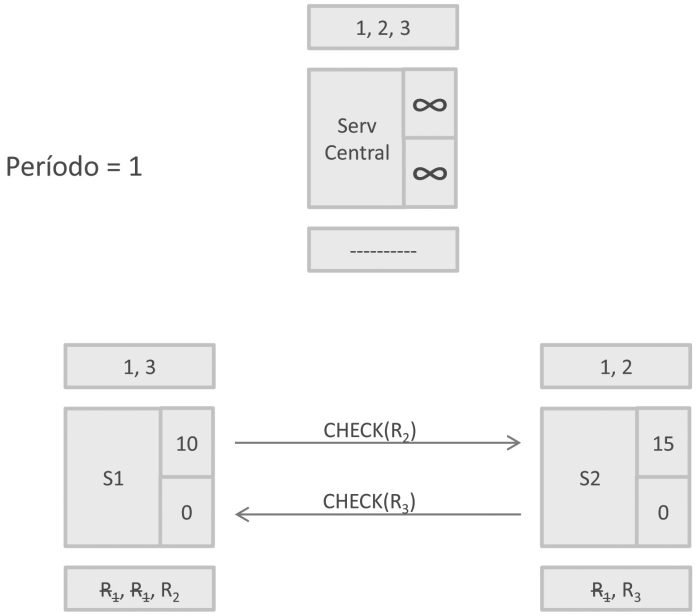


Figura 4.8: Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo.

Cada servidor, ao receber uma mensagem $CHECK$ verifica se possui recursos necessários para atender as requisições direcionadas. Em seguida, envia uma mensagem $CHECKED$ com as requisições que puderam e que não puderam ser atendidas. Neste exemplo, na Figura 4.9, ambos os servidores conseguiram atender as requisições direcionadas.

No período de tempo seguinte, as requisições que começaram a ser atendidas localmente no período 0 em ambos servidores, S_1 e S_2, completam seu atendimento, liberando a

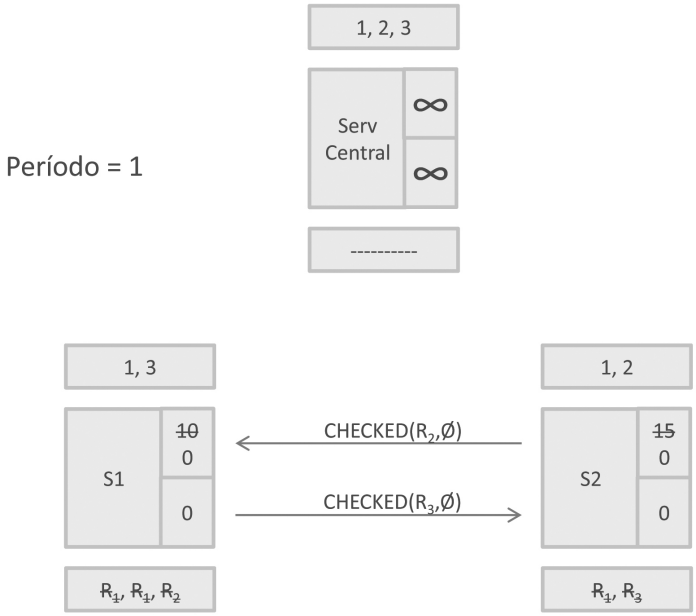


Figura 4.9: Os servidores enviam $CHECKED$ com as requisições atendidas e não atendidas.

banda utilizada (Figura 4.10). Além disso, como estes dois servidores também possuem requisições externas completamente atendidas, o servidor S_1 envia mensagem $LIBERA_BANDA$ com o valor da banda utilizada nos atendimentos para o servidor S_2, e vice-versa (Figura 4.11).

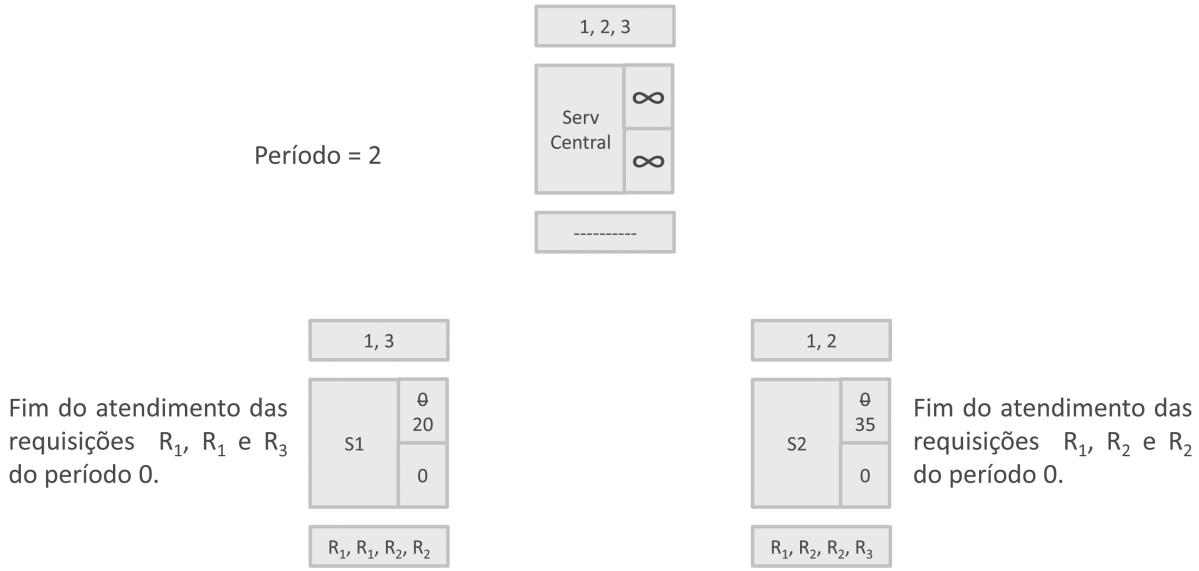


Figura 4.10: Fim de atendimentos locais.

Novamente, cada servidor tenta atender localmente suas novas requisições. Na Figura 4.12, pode-se observar que o servidor S_1 consegue atender localmente as duas requisições R_2 e o servidor S_2, a requisição R_3.

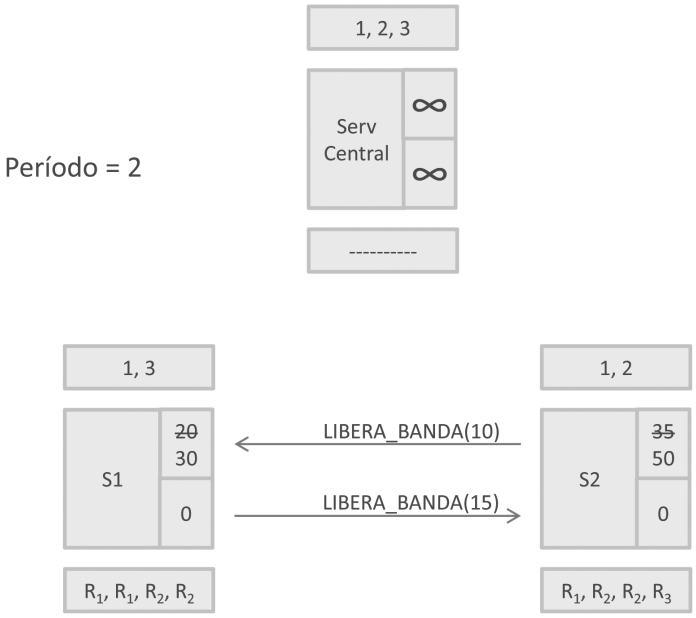


Figura 4.11: Fim de atendimentos externos.

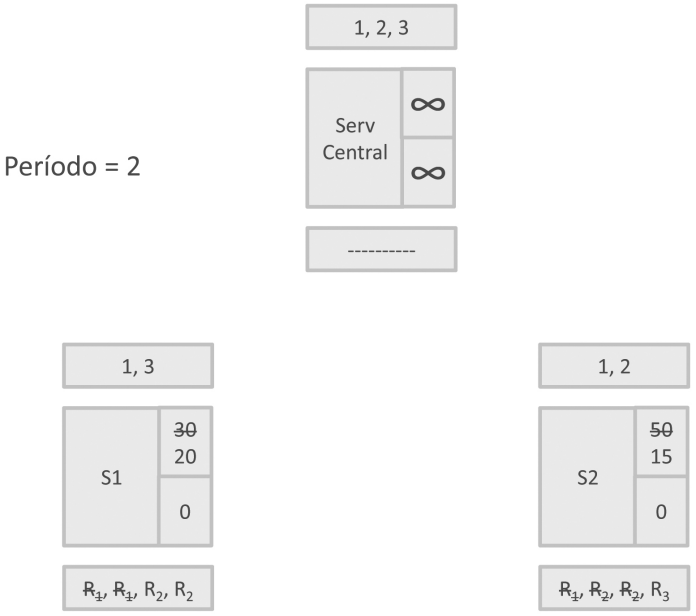


Figura 4.12: Cada servidor tenta atender as requisições localmente.

Assim, cada servidor direciona suas requisições não atendidas diretamente para o servidor mais próximo que possua o conteúdo solicitado replicado através da mensagem $CHECK$. Desta forma, na Figura 4.13, o servidos S_1 envia $CHECK(R_2, R_2)$ para o servidor S_2 e o servidor S_2 envia $CHECK(R_3)$ para o servidor S_1.

Conforma já descrito, cada servidor, ao receber uma mensagem $CHECK$ verifica se possui recursos necessários para atender as requisições direcionadas. Em seguida, envia uma mensagem $CHECKED$ com as requisições que puderam e que não puderam ser

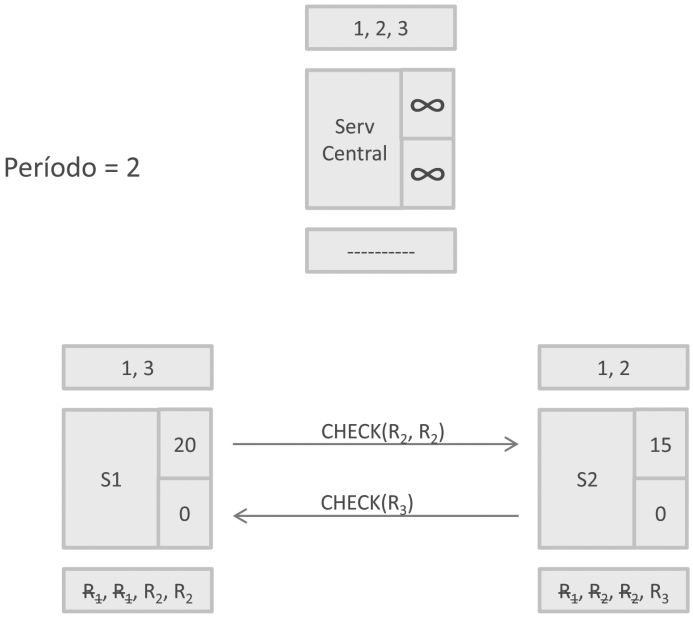


Figura 4.13: Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo.

atendidas. Neste exemplo, na Figura 4.14, o servidor S_1 conseguiu atender a requisição direcionada e o servidor S_2 não conseguiu atender uma requisição R_2.

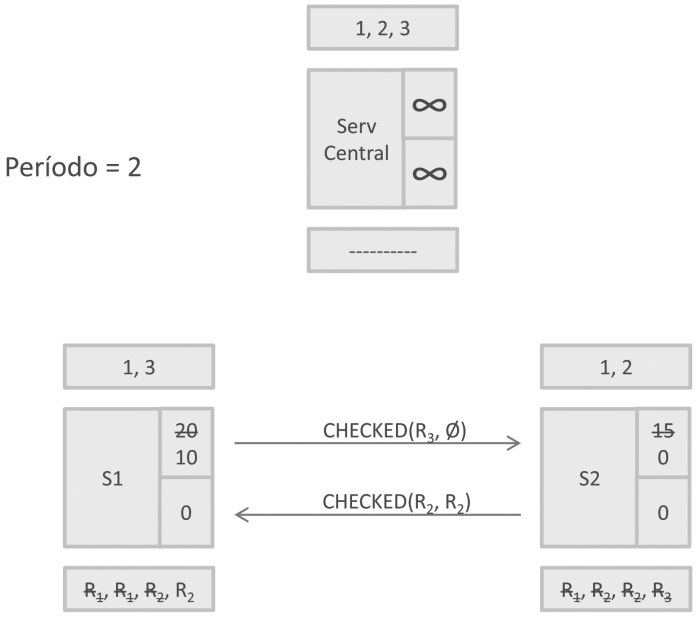


Figura 4.14: Os servidores enviam $CHECKED$ com as requisições atendidas e não atendidas.

Considerando que este é o último período de tempo da execução, o servidor S_1 não conseguiu atender sua requisição R_2, nem localmente nem através de um outro servidor. Dessa forma, na Figura 4.15, este envia uma mensagem $ATTEND_CENTRAL$ para que o servidor origem deste conteúdo atenda esta requisição (com um custo de comunicação

elevado).

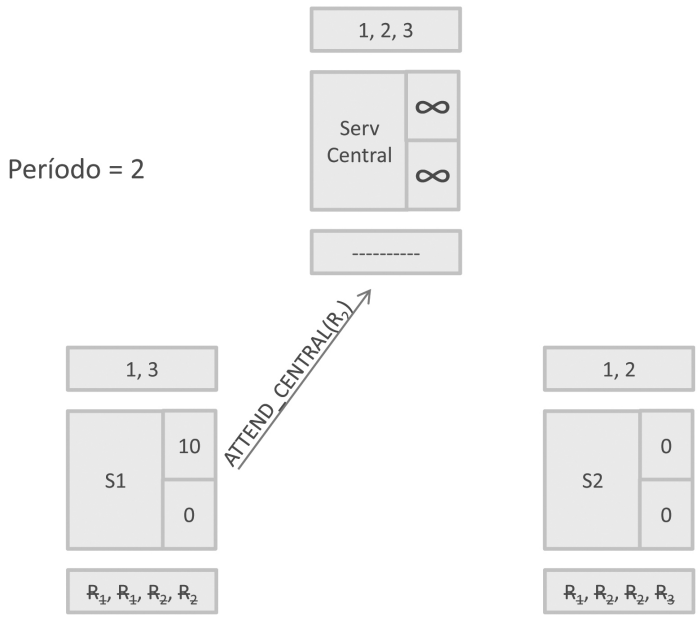


Figura 4.15: O servidor S_1 envia $ATTEND_CENTRAL$ para o servidor origem.

4.2.3 Complexidades

Nesta seção, as complexidades do algoritmo distribuído RPA são descritas. Para a versão dinâmica, inicialmente, cada servidor envia uma mensagem $STATIC$ para o servidor central, que a responde com outra mensagem $STATIC$. A seguir, no pior caso, para cada conteúdo, cada servidor (exceto o central) envia uma mensagem $CHECK$ para os demais servidores a cada período. Em resposta a mensagem $CHECK$, uma mensagem $CHECKED$ também é enviada. E ainda, para cada requisição completamente atendida externamente, cada servidor envia uma mensagem $LIBERA_BANDA$ para o servidor que a atendeu. Por fim, como as requisições não atendidas após um determinado número de períodos são direcionadas para o servidor origem do conteúdo, são enviadas x mensagens $ATTEND_CENTRAL$, o x é o número de conteúdos de requisições que não foram atendidas. Então, o número total de mensagens enviadas é: $(2+2\times(numConteudos\times(numServidores-1)\times(numPeriodos-1))\times numServidores)+numRequisicoes+x$. Desta forma, a complexidade de mensagens é $O(MAX((numConteudos\times(numServidores-1)\times(numPeriodos-1))\times numServidores), numRequisicoes, x)$.

A complexidade de tempo global é definida como a maior cadeia de causalidade do tipo "recebeu uma mensagem e enviou outra em consequência"[4] que ocorre na execução do algoritmo. No pior caso, para o atendimento de uma requisição é necessário percorrer

todos os demais servidores em todos os p períodos (definido como parâmetro, sendo no máximo igual a $numPeriodos$). Dessa forma, a complexidade de tempo global para o atender uma requisição é $O(numServidores \times p)$.

A complexidade de tempo local é determinada pelo tempo de computação local entre envio e recebimentos de mensagens. Assim, a complexidade de tempo local é, no pior caso, o máximo entre o tempo de percorrer as requisições originadas localmente ou as requisições externas que estão sendo atendidas. Portanto, a complexidade de tempo local é $O(MAX(numReq_t, \sum_{t'=0}^{t} numReqExt_{t'}))$.

Para determinar o tamanho máximo de mensagem enviada pelo algoritmo, é necessário observar apenas três tipos de mensagens: $STATIC$, $CHECK$ e $ATTEND_CENTRAL$. Isto é devido ao fato das mensagens $CHECK$ e $CHECKED$ possuírem, no pior caso, o mesmo tamanho. O tamanho das mensagens $STATIC$ é no máximo igual a quantidade de *bytes* para representar o conjunto de requisições locais atendidas mais o conjunto de requisições externas atendidas no primeiro período de tempo. O maior tamanho das mensagens $CHECK$ é igual ao maior conjunto de requisições de um mesmo conteúdo. O tamanho da mensagem $ATTEND_CENTRAL$ é definida pelo maior conjunto de requisições não atendidas em cada servidor. Dessa forma, o maior tamanho de mensagem enviada é obtida através do máximo dos tamanhos das mensagens $STATIC$, $CHECK$ e $ATTEND_CENTRAL$.

4.3 Algoritmo Distribuído RPA_Mod

Nesta subseção é apresentado o algoritmo distribuído RPA_Mod para a segunda abordagem do PPR On-Line em RDC. Este algoritmo é uma modificação do algoritmo distribuído RPA (Seção 4.2) para tratar as considerações presentes na segunda abordagem do PPR (mais próxima da realidade) e também para aprimorar o RPA. No algoritmo distribuído RPA_Mod foi retirado o ponto de centralização (com a participação do servidor central, visto como um gargalo) e também foi diminuído o tamanho das mensagens enviadas. Dessa forma, pode-se afirmar que o RPA_Mod é um algoritmo distribuído melhor, podendo ser considerado totalmente distribuído.

4.3.1 Procedimentos

Como este algoritmo é uma modificação do RPA, nesta subseção apenas os procedimentos e funções que foram adicionados, removidos ou que sofreram alteração serão descritos. As

estruturas de dados utilizadas são as mesmas utilizadas pelo algoritmo distribuído RPA.

O algoritmo se inicia de maneira semelhante ao anterior. Entretanto, as possíveis requisições não atendidas localmente são direcionadas para o servidor mais próximo que possua o conteúdo solicitado e os conteúdos replicados só estarão disponíveis para atendimento no próximo período. Se uma replicação não puder ser realizada por falta de espaço de armazenamento, verifica-se a existência de alguma réplica que não esteja sendo utilizada. Em caso positivo, esta réplica é removida liberando recursos no servidor em questão (Algoritmo 14). As alterações dos algoritmos 1 e 2 são apresentadas nos algoritmos 12 e 13.

Algoritmo 12: Inicialização /* Executado por um servidor s */

```
1 periodo := 0;
2 atenderLocal(periodo);
3 contMsg := 0;
4 Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor com menor
  custo de transporte e que possua o conteúdo contMsg;
```

Algoritmo 13: atenderLocal(periodo) /* *Executado por um servidor s* */

```
1  para p:=1 até periodo faça
2      para cada requisição r do período p em ordem crescente da função de lucro faça
3          se a requisição r ainda não foi atendida então
4              se contReq_rt está replicado localmente no período periodo então
5                  se bandaTotal ⩾ bandReq_rt então
6                      Marcar a requisição r do período p como atendida;
7                      iniReq_rt := p;
8                      fimReq_rt := p + ⌈tamReq_rt/bandReq_rt⌉;
9                      servReq_rt := meu_id;
10                     bandaTotal := bandaTotal − bandReq_rt;
11             senão
12                 se capTotal < tamReq_rt e bandaTotal ⩾ bandReq_rt então
13                     removerConteudosNaoUtilizados(); /* Algoritmo 14 */
14                 se capTotal ⩾ tamReq_rt e bandaTotal ⩾ bandReq_rt então
15                     Replicar localmente o conteúdo contReq_rt no período periodo + 1;
16                     capTotal := capTotal − tamReq_rt;
17                     custoReplicacaoTotal := custoReplicacaoTotal + custoReplicacao_contReq_rt;
```

Um servidor ao receber mensagem $CHECK$ realiza as mesmas ações do algoritmo distribuído RPA. Ou seja, verifica se é possível atender as requisições do servidor remetente. Desta forma, atualiza seus recursos locais e responde de volta informando as requisições que puderam ser atendidas, através da mensagens $CHECKED$ (Algoritmo 8).

De maneira semelhante ao RPA, se o servidor para qual a requisição foi direcionada não possui banda suficiente para o atendimento, esta é redirecionada para o próximo servidor com menor custo de transporte. Se todos os servidores forem percorridos e nenhum

puder atender a requisição, esta fica para o próximo período de tempo e é adicionado o custo de *backlog* sobre a fração de requisição que deixou de ser atendida.

Quando um servidor não consegue atender suas requisições até o último período (dado como parâmetro do sistema), este envia uma mensagem $ATTEND_CENTRAL$ para o Servidor Central com suas requisições não atendidas. Estas ações são descritas no Algoritmo 15 e são disparadas com o recebimento de uma mensagem $CHECKED$. Como não existe mais a participação do Servidor Central nesta segunda abordagem, o algoritmo 6 foi substituído pelo algoritmo 16.

Algoritmo 14: *removerConteudosNaoUtilizados*() /* *Executado por um servidor* s_j */

```
1  para cada período t' ≤ periodo faça
2      para cada conteúdo c faça
3          semAtend := true;
4          para cada requisição r do período t' tal que contReq_rt' = c faça
5              se servReq_rt' = meu_id e fimReq_rt' < periodo então
6                  semAtend := false;
7          para cada requisição externa r do período t' tal que contReqExt_rt' = c faça
8              se fimReq_rt' < periodo então
9                  semAtend := false;
10         se semAtend = true então
11             Remover o conteúdo c;
12             capTotal := capTotal + tamCont_c;
```

Nesta segunda abordagem do PPR, a cada mudança de período os custos de transporte entre os servidores são atualizados, recebendo os valores definidos para o período atual.

4.3.2 Exemplo

Nesta seção é exposto um exemplo de uma possível execução do algoritmo distribuído RPA_Mod.

Na Figura 4.16 é definido o significado de cada campo das figuras utilizadas neste exemplo. De forma similar ao exemplo anterior, os campos B e C dos servidores representam, respectivamente, a banda e capacidades disponíveis. Os campos *Conteúdos* e *Requisições* representam, respectivamente, os conteúdos replicados localmente e as requisições locais originadas no período definido em *Período*. Diferentemente do exemplo da Seção 4.2.2, neste exemplo não existe servidor central. Em seu lugar, mais um servidor de distribuição foi adicionando.

Este exemplo contém o seguinte cenário ilustrado na Figura 4.17:

- Três servidores de distribuição da RDC (S_1, S_2 e S_3), com 50, 70 e 40 unidades de

Algoritmo 15: Ao receber uma mensagem $CHECKED$(*requisições atendidas e não atendidas do conteúdo contMsg*) do servidor s /* *Executado por um servidor* s_j */

```
1  atualizaLocal(); /*Algoritmo 16*/
2  se Todas as requisições do conteúdo contMsg foram atendidas então
3      contMsg := contMsg + 1;
4      se contMsg < numConteudos então
5          Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor com
           menor custo de transporte e que possua o conteúdo contMsg;
6      senão
7          periodo := periodo + 1;
8          se periodo < numPeriodos então
9              atenderLocal(periodo);
10             contMsg := 0;
11             Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor
               com menor custo de transporte e que possua o conteúdo contMsg;
12 senão
13     se ∃ servidor não percorrido para o conteúdo contMsg então
14         Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o próximo
           servidor com menor custo de transporte e que possua o conteúdo contMsg;
15     senão
16         periodo := periodo + 1;
17         se periodo < numPeriodos então
18             Adicionar custo de backlog sobre as requisições não atendidas no seu período de
               origem;
19             atenderLocal(periodo);
20             contMsg := 0;
21             Enviar CHECK(requisições não atendidas do conteúdo contMsg) para o servidor
               com menor custo de transporte e que possua o conteúdo contMsg;
22         senão
23             /* Requisições locais não foram atendidas */
24             Enviar ATTEND_CENTRAL(requisições não atendidas) para o servidor central;
```

Algoritmo 16: *atualizaLocal*() /* *Executado por um servidor* s_j */

```
1  para cada requisição atendidas r faça
2      Marcar a requisição r do período p como atendida;
3      iniReq_rt := periodo;
4      fimReq_rt := periodo + ⌈tamReq_rt / bandReq_rt⌉;
5      servReq_rt := s;
6  para cada período t' ≤ periodo faça
7      para cada requisição atendida r do período t' faça
8          se fimReq_rt' = periodo então
9              se servReq_rt' = meu_id então
10                 bandaTotal := bandaTotal + bandReq_rt';
11             senão
12                 Enviar LIBERA_BANDA(bandReq_rt') para o servidor
                   servReq_rt';/*Algoritmo 7*/
```

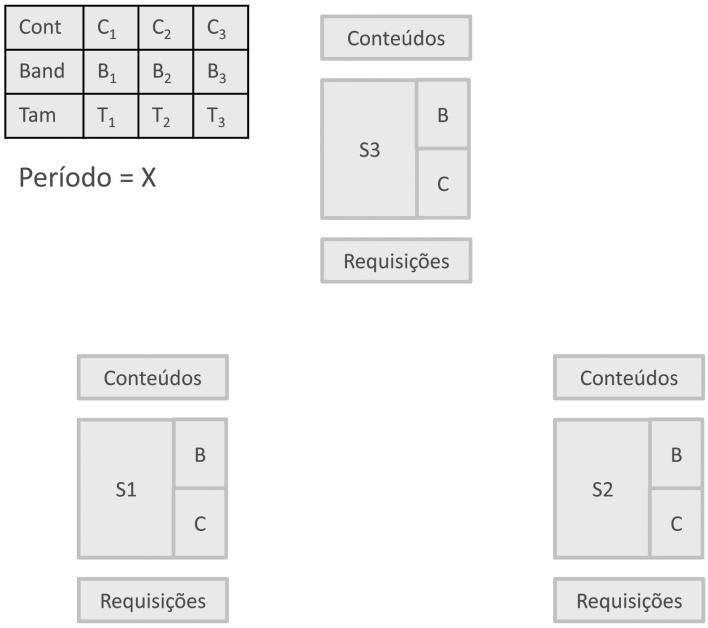


Figura 4.16: Dados do problema.

banda e 25, 30 e 35 unidades de armazenamento respectivamente.

- Três conteúdos (1,2 e 3) presentes na RDC.
- As requisições são apresentadas na notação R_c, onde c define o conteúdo solicitado.
- Na tabela exposta no canto superior esquerdo, estão as informações de banda e tamanho necessários para cada conteúdo.

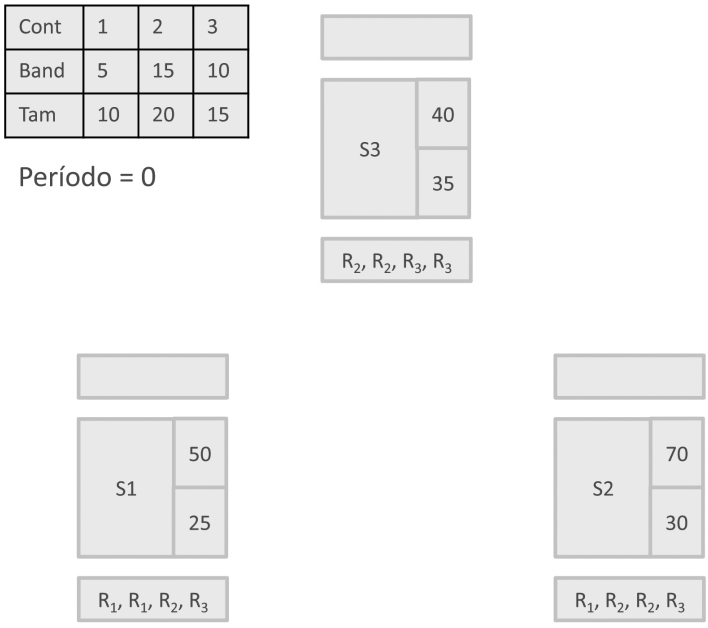


Figura 4.17: Uma configuração inicial.

Inicialmente, cada servidor tenta atender suas requisições locais, replicando os conteúdos em ordem da função de lucro. Desta forma, na Figura 4.18, pode-se observar que o servidor S_1 replica os conteúdos 1 e 3 e atende localmente as duas requisições R_1 e a requisição R_3. Entretanto, este não consegue atender a requisição R_2, pois mesmo possuindo a banda necessária, este não possui capacidade de armazenamento para replicar o conteúdo 2.

Já o servidor S_2 replica os conteúdos 1 e 2 e atende localmente a requisições R_1 e as duas requisições R_2. De maneira similar ao servidor S_1, o servidor S_2 não consegue atender a requisição R_3 por falta de capacidade de armazenamento para o conteúdo 3. O servidor S_3 replica os conteúdos 2 e 3 e atende localmente a requisição R_2 e as duas requisições R_3. Entretanto, por falta de banda disponível, uma requisição R_2 não pode ser atendida.

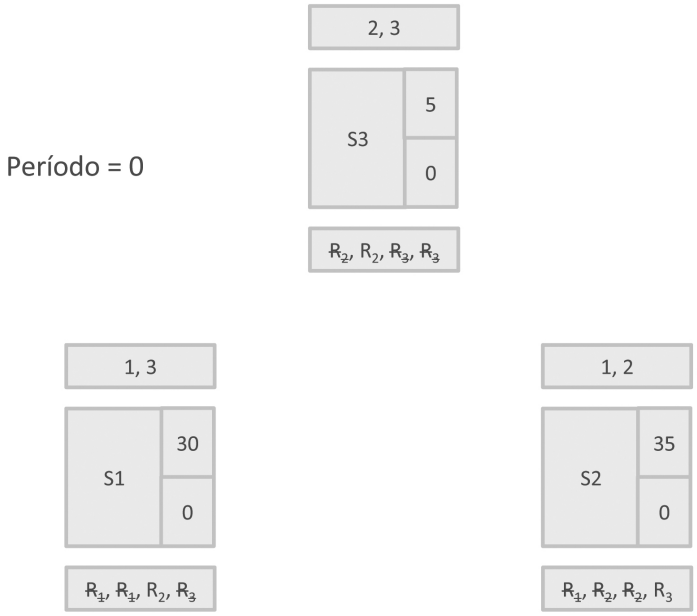


Figura 4.18: Cada servidor tenta atender as requisições localmente.

Em seguida, cada servidor direciona suas requisições não atendidas diretamente para o servidor mais próximo que possua o conteúdo solicitado replicado através da mensagem $CHECK$. Desta forma, na Figura 4.19, considerando que o servidor mais próximo do servidor S_1 é o servidor S_2 e que o servidor mais próximo de S_2 é o servidor S_3, o servidor S_1 envia $CHECK(R_2)$ para o servidor S_2, o servidor S_2 envia $CHECK(R_3)$ para o servidor S_3 e o servidor S_3 envia $CHECK(R_2)$ para o servidor S_2.

Cada servidor, ao receber uma mensagem $CHECK$ verifica se possui recursos necessários para atender as requisições direcionadas. Em seguida, envia uma mensagem $CHECKED$ com as requisições que puderam e que não puderam ser atendidas. Neste

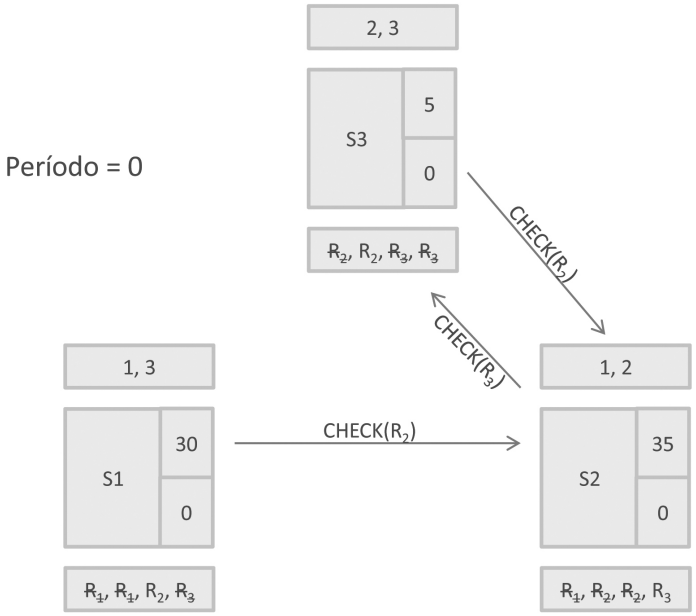


Figura 4.19: Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo.

exemplo, na Figura 4.20, ambos os servidores S_1 e S_3 conseguiram atender as requisições direcionadas. Entretanto, o servidor S_2 não conseguiu atender sua requisição R_3.

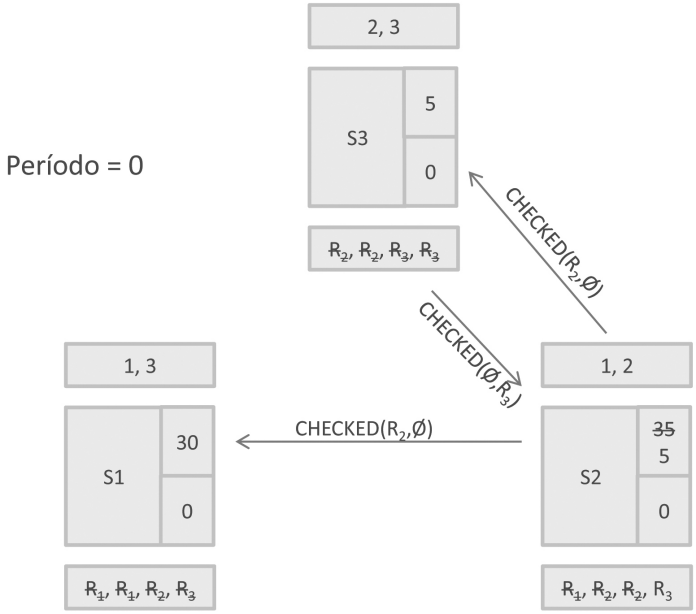


Figura 4.20: Os servidores enviam CHECKED com as requisições atendidas e não atendidas.

Desse modo, o servidor S_2 envia novamente mensagem $CHECK(R_3)$, agora para o servidor S_1 (Figura 4.21).

O servidor S_1 ao receber a mensagem $CHECK$ verifica que pode atender a requisição direcionada e envia uma mensagem $CHECKED$ com as requisições que puderam e que

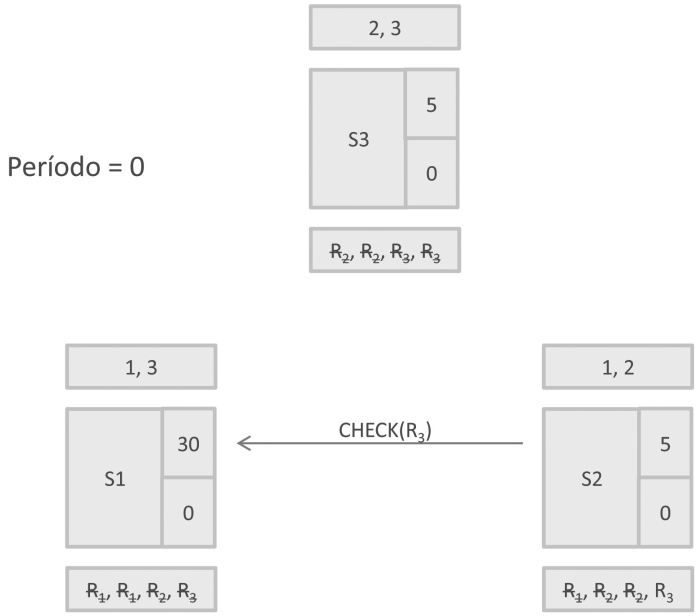


Figura 4.21: O servidor S_2 envia $CHECK$ para o segundo servidor mais próximo que possui o conteúdo.

não puderam ser atendidas (Figura 4.22).

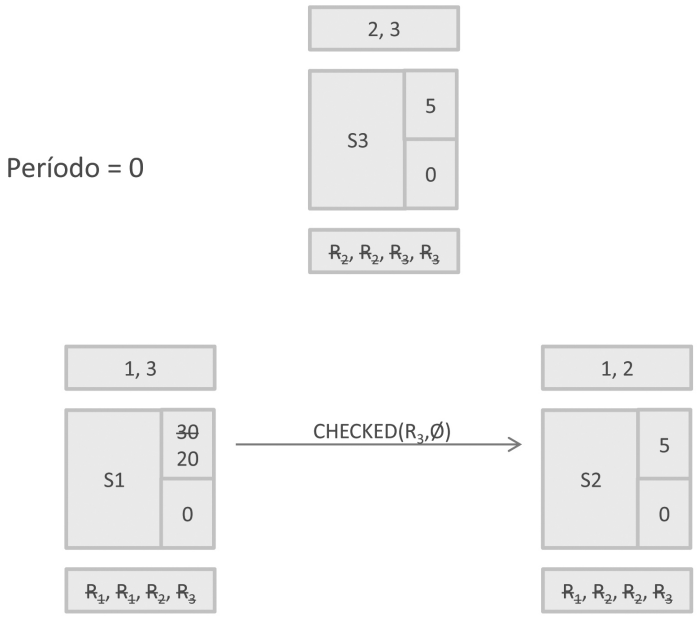


Figura 4.22: O servidor S_1 envia $CHECKED$ com as requisições atendidas e não atendidas.

Como todos os servidores tentaram atender todas as suas requisições deste período, estes passam para o próximo período. Assim, novas requisições surgem em cada servidor (Figura 4.23).

Novamente, cada servidor tenta atender suas requisições localmente. Na Figura 4.24, apenas o servidor S_3 não atende todas as suas requisições por falta de banda disponível.

Figura 4.23: Novas requisições surgem no período de tempo seguinte.

Figura 4.24: Cada servidor tenta atender as requisições localmente.

Em seguida, o servidor S_3 envia mensagem $CHECK$ para o servidor mais próximo que possui o conteúdo solicitado, que no exemplo é o servidor S_1 (Figura 4.25).

Assim, o servidor S_1 verifica que pode atender a requisição direcionada e envia uma mensagem $CHECKED$ com as requisições que puderam e que não puderam ser atendidas (Figura 4.26).

De maneira similar ao exemplo do RPA, no período de tempo seguinte, as requisições que começaram a ser atendidas localmente no período 0 em ambos servidores, S_1 e S_2,

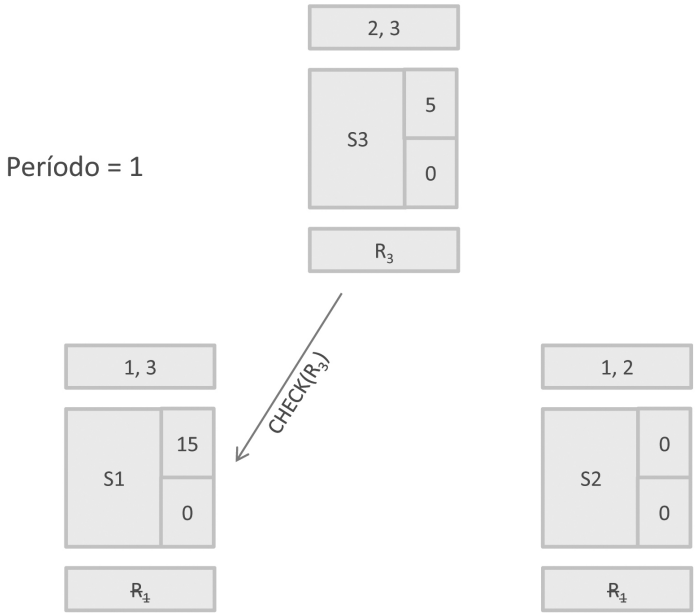


Figura 4.25: O servidor S_3 envia $CHECK$ para o servidor mais próximo que possui o conteúdo.

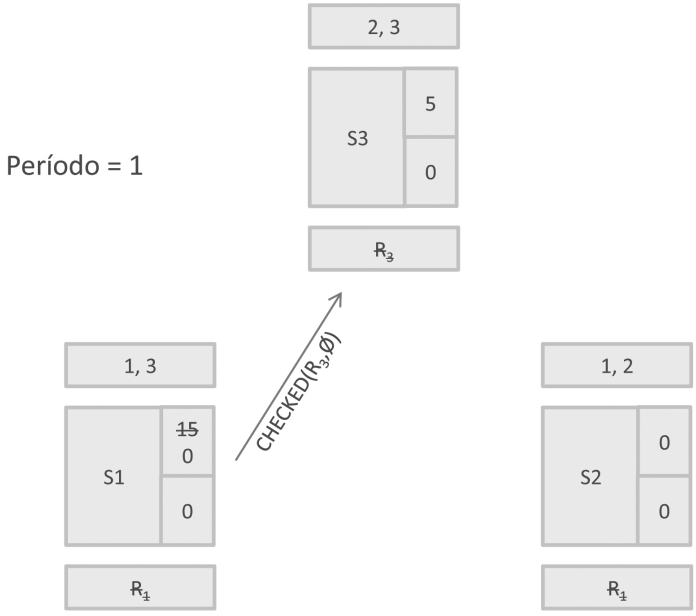


Figura 4.26: O servidor S_1 envia $CHECKED$ com as requisições atendidas e não atendidas.

completam seu atendimento, liberando a banda utilizada (Figura 4.27). Além disso, como estes dois servidores também possuem requisições externas completamente atendidas, o servidor S_1 envia mensagem $LIBERA_BANDA$ com o valor da banda utilizada nos atendimentos para o servidor S_2, e vice-versa (Figura 4.28).

Novamente, cada servidor tenta atender localmente suas novas requisições. Na Figura 4.29, pode-se observar que o servidor S_1 consegue atender localmente as duas requisições

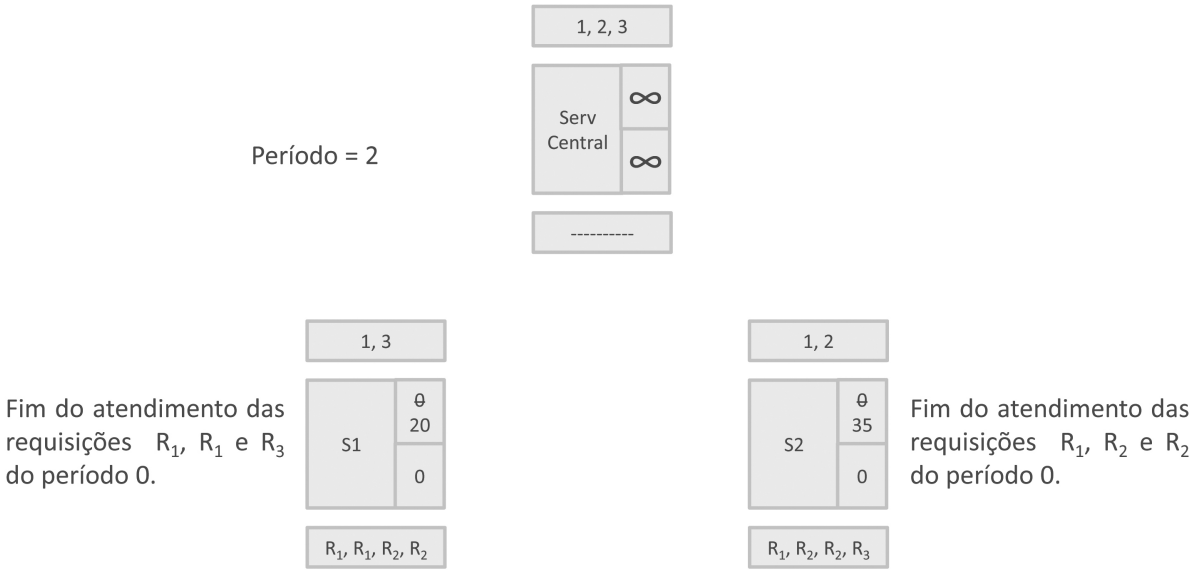


Figura 4.27: Fim de atendimentos locais.

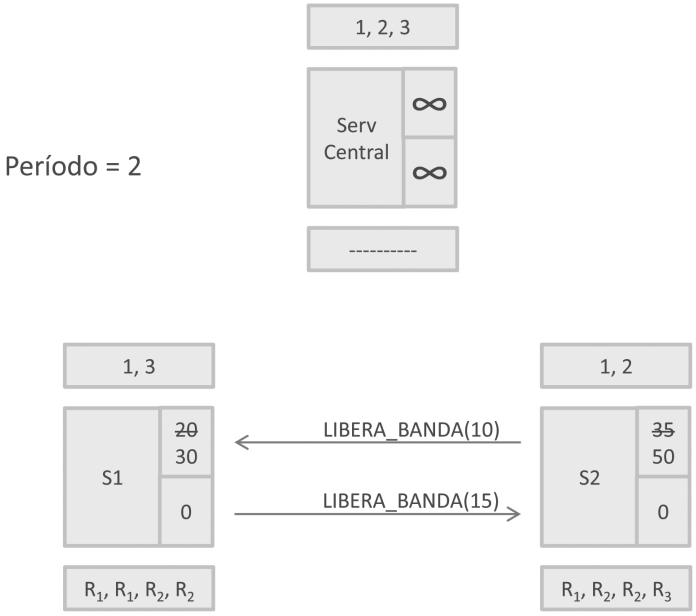


Figura 4.28: Fim de atendimentos externos.

R_2 e o servidor S_2, a requisição R_3.

Assim, cada servidor direciona suas requisições não atendidas diretamente para o servidor mais próximo que possua o conteúdo solicitado replicado através da mensagem $CHECK$. Desta forma, na Figura 4.30, o servidos S_1 envia $CHECK(R_2, R_2)$ para o servidor S_2 e o servidor S_2 envia $CHECK(R_3)$ para o servidor S_1.

Conforma já descrito, cada servidor, ao receber uma mensagem $CHECK$ verifica se possui recursos necessários para atender as requisições direcionadas. Em seguida, envia uma mensagem $CHECKED$ com as requisições que puderam e que não puderam ser

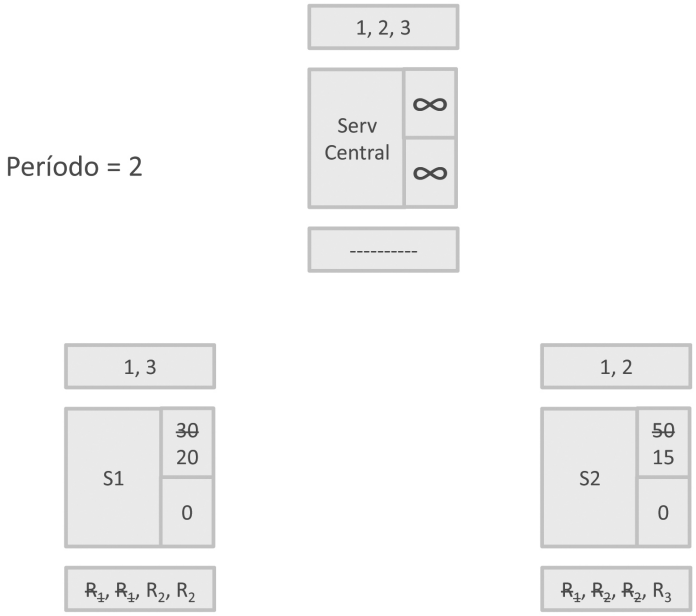


Figura 4.29: Cada servidor tenta atender as requisições localmente.

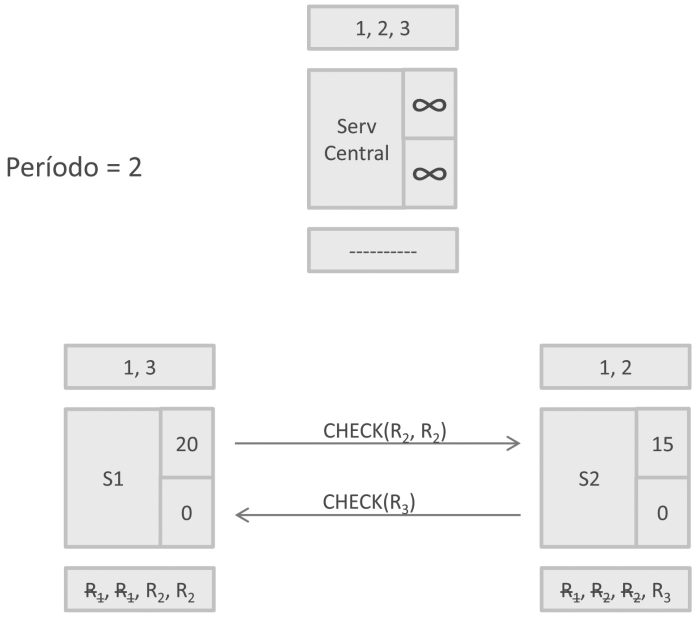


Figura 4.30: Cada servidor envia $CHECK$ para o servidor mais próximo que possui o conteúdo.

atendidas. Neste exemplo, na Figura 4.31, o servidor S_1 conseguiu atender a requisição direcionada e o servidor S_2 não conseguiu atender uma requisição R_2.

Considerando que este é o último período de tempo da execução, o servidor S_1 não conseguiu atender sua requisição R_2, nem localmente nem através de um outro servidor. Dessa forma, na Figura 4.32, este envia uma mensagem $ATTEND_CENTRAL$ para que o servidor origem deste conteúdo atenda esta requisição (com um custo de comunicação elevado).

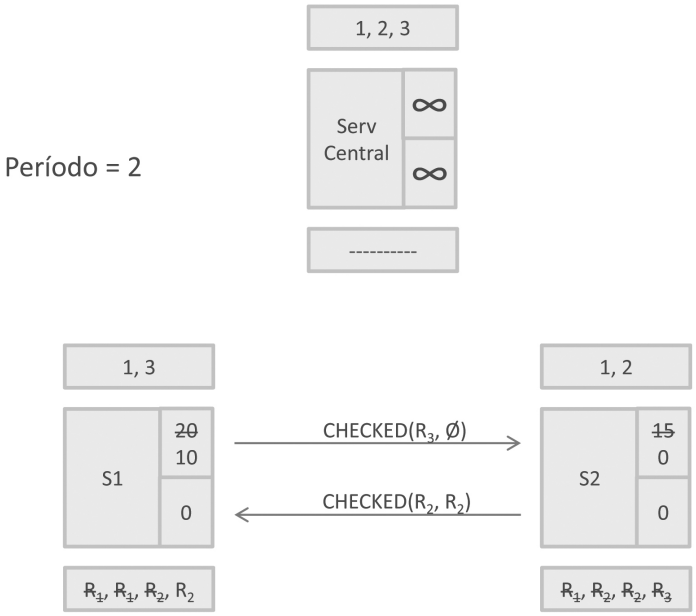


Figura 4.31: Os servidores enviam $CHECKED$ com as requisições atendidas e não atendidas.

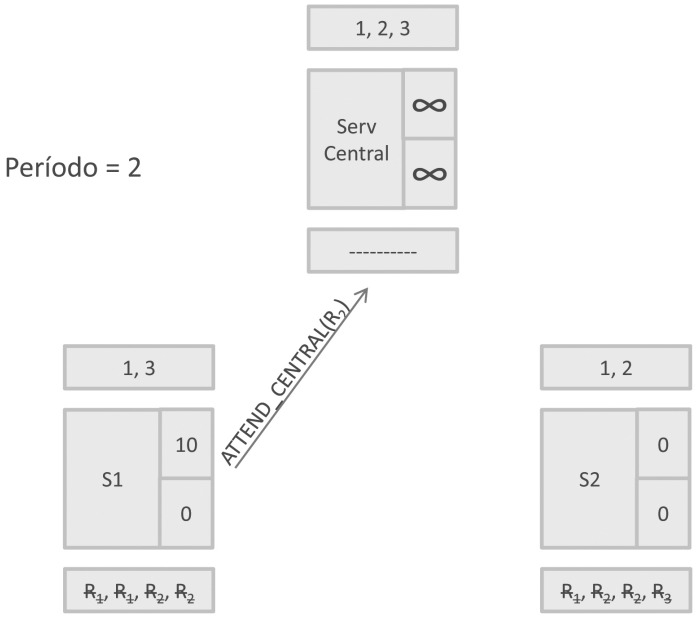


Figura 4.32: O servidor S_1 envia $ATTEND_CENTRAL$ para o servidor origem.

4.3.3 Complexidades

Nesta seção as complexidades do algoritmo distribuído RPA_Mod são descritas. Ao contrário do primeiro, este algoritmo inicialmente não envia mensagens para o Servidor Central, redirecionando possíveis requisições não atendidas para os servidores mais próximos. Desta forma, cada servidor, para cada conteúdo envia, no pior caso uma mensagem $CHECK$ para os demais servidores a cada período. Cada mensagem $CHECK$ é

respondida por uma mensagem $CHECKED$. Ademais, para cada requisição completamente atendida externamente, cada servidor envia uma mensagem $LIBERA_BANDA$ para o servidor que a atendeu. Por fim, considerando também que x é o número de mensagens $ATTEND_CENTRAL$ enviadas, o número total de mensagens enviadas é: $(2 \times (numConteudos \times (numServidores - 1) \times numPeriodos) \times numServidores) + numRequisicoes + x$. Desta forma, a complexidade de mensagens é $O(MAX(numConteudos \times (numServidores - 1) \times numPeriodos) \times numServidores), numRequisicoes, x)$.

A complexidade de tempo global é a mesma do algoritmo distribuído RPA: $O(numServidores \times p)$, onde p é o número de períodos (definido como parâmetro) de tentativas que a requisição tem para começar a ser atendida antes de ser direcionada para o Servidor Central.

A complexidade de tempo local é, no pior caso, o tempo de percorrer todas as requisições originadas localmente. Portanto, esta é $O(numReq_t)$.

Para determinar o tamanho máximo de mensagem enviada pelo algoritmo, é necessário observar apenas dois tipos de mensagens: $CHECK$ e $ATTEND_CENTRAL$, pois no pior caso, as mensagens $CHECK$ e $CHECKED$ possuem o mesmo tamanho. Neste algoritmo não existe envio de mensagem $STATIC$. O maior tamanho da mensagens $CHECK$ é igual ao maior conjunto de requisições de um mesmo conteúdo. O tamanho da mensagem $ATTEND_CENTRAL$ é definida pelo maior conjunto de requisições não atendidas em cada servidor. Dessa forma, o maior tamanho de mensagem enviada é obtida através do máximo dos tamanhos das mensagens $CHECK$ e $ATTEND_CENTRAL$.

Capítulo 5

Resultados Computacionais

Neste capítulo serão apresentados os resultados experimentais encontrados nas simulações e testes realizados. Este capítulo está dividido em três seções. Na primeira seção, a Formulação Matemática I, a Heurística Ótima por Período e o algoritmo distribuído RPA são comparados. Na segunda seção, são apresentados os resultados da comparação entre a Formulação Matemática II e o algoritmo distribuído RPA_Mod. Na terceira seção, o algoritmo distribuído RPA_Mod é comparado com o trabalho relacionado proposto em [22, 19], chamado aqui de Presti.

5.1 Formulação Matemática I × Heurística Ótima por Período × RPA

Para comparar os resultados da Formulação Matemática I, da Heurística Ótima por Período e do algoritmo distribuído RPA propostos, testes foram realizados em uma máquina com a seguinte configuração: processador Phenom X4 9600 de 2.4Ghz e 4Gb de memória. Para a Formulação Matemática I, foi utilizado o resolvedor *CPLEX* 11.2 [9]. O RPA foi implementado em linguagem ANSI C, utilizando o MPICH2 para implementar trocas de mensagens, na distribuição Ubuntu 10.04 do sistema operacional Linux.

As instâncias utilizadas nos testes foram obtidas em [17] e estão disponíveis em [14]. De acordo com [17], as instâncias são divididas em três tipos: fáceis de resolver (x-1 até x-5), intermediárias (x-06 até x-10) e difíceis(x-10 até x-15), onde x é o numero de servidores. Como neste trabalho não é considerado que os conteúdos possuem tempo de vida (períodos que estão disponíveis) e que os enlaces não sofrem variações com o tempo, estas informações presentes nas instâncias não foram utilizadas. Desta forma, a

partir do período de surgimento de um conteúdo, este permanece disponível até o final da simulação, e as características dos enlaces são definidas a partir das informações do primeiro período.

Os resultados obtidos nos testes são apresentados nas Tabelas 5.1, 5.2, 5.3 e 5.4, para instâncias com 10, 20, 30 e 50 servidores, respectivamente. Na última linha da Tabela 5.4 são apresentadas as médias para o tempo e o gap de todas as instâncias. A primeira coluna contém o nome da instância. A segunda e a terceira, o valor e o tempo da solução ótima, obtida com a Formulação Matemática I. A quarta, quinta e sexta, contêm o valor, o número de mensagens e o tempo da simulação do algoritmo distribuído RPA. Na sétima, é apresentada o *gap* entre a solução distribuída e o ótimo. A oitava e a nona contêm o valor e o tempo de simulação da Heurística Ótima por Período. Por fim, a décima coluna apresenta o *gap* entre a solução da Heurística Ótima por Período e a solução ótima.

Tabela 5.1: Formulação Matemática I × RPA × Heurística Ótima por Período - para 10 servidores

Instâncias	Num Req	Solução Ótima	Tempo (s)	Solução Distribuída	Mensagens	Tempo (s)	Gap (%)	Heur. Ótima por período	Tempo (s)	Gap (%)
10-1	613	540,00	0,30	540,00	1765,00	0,00	0,00	540,00	0,00	0,00
10-2	526	860,00	0,25	860,00	2616,00	0,00	0,00	860,00	1,00	0,00
10-3	519	751,00	0,24	751,00	2192,00	0,10	0,00	751,00	0,00	0,00
10-4	632	760,00	0,31	760,00	2614,00	0,00	0,00	760,00	0,00	0,00
10-5	445	588,00	0,19	588,00	2186,00	0,10	0,00	588,00	0,00	0,00
10-06	627	30162,00	0,92	38302,00	12395,00	0,40	26,99	36871,00	1,00	22,24
10-07	675	29303,00	0,95	33702,00	11363,00	0,20	15,01	35330,00	0,00	20,57
10-08	641	29548,00	0,98	36776,00	13415,00	0,30	24,46	34811,00	0,00	17,81
10-09	659	29107,00	1,73	38716,00	14441,00	0,10	33,01	35186,00	1,00	20,89
10-10	649	32662,00	1,11	40875,00	15472,00	0,40	25,15	40887,00	0,00	25,18
10-11	627	30162,00	0,95	37810,70	12387,60	0,10	25,36	36409,00	0,00	20,71
10-12	675	29303,00	0,96	33844,20	11360,00	0,30	15,50	35378,00	1,00	20,73
10-13	641	29548,00	0,99	36299,30	13404,40	0,30	22,85	34175,00	0,00	15,66
10-14	659	29107,00	1,76	36005,70	14418,00	0,40	23,70	35407,00	1,00	21,64
10-15	649	32681,00	1,26	39982,60	15456,00	0,30	22,34	40508,00	0,00	23,95

Em todas as simulações realizadas, o tamanho máximo das mensagens foi de 136 *bytes* (17 variáveis *double*). Além disso, para a Formulação Matemática I e a heurística centralizada o início de atendimento das requisições foi fixado para o período que estas surgem na RDC.

O RPA apresentou resultados satisfatórios, ficando em média 19.5% do ótimo, visto que a heurística ótima por período ficou em média 15% do ótimo. Além disso, os tempos de simulação do RPA foram significativamente menores que os gastos pelo resolvedor e pela heurística centralizada e o número de mensagens enviadas foi, em média, 83.4% menor que o limite dado pela complexidade teórica.

Em algumas instâncias, como por exemplo 20-06, 50-13 e 50-14, pode-se observar

Tabela 5.2: Formulação Matemática I × RPA × Heurística Ótima por Período - para 20 servidores

Instâncias	Num Req	Solução Ótima	Tempo (s)	Solução Distribuída	Mensagens	Tempo (s)	Gap (%)	Heur. Ótima por período	Tempo (s)	Gap (%)
20-1	715	760,00	1,19	760,00	3510,00	0,00	0,00	760,00	1,00	0,00
20-2	1525	1280,00	3,04	1280,00	3506,00	0,00	0,00	1280,00	1,00	0,00
20-3	1602	1372,00	3,06	1372,00	4381,00	0,10	0,00	1372,00	1,00	0,00
20-4	1224	1488,00	2,09	1488,00	4376,00	0,10	0,00	1488,00	2,00	0,00
20-5	1366	1010,00	2,40	1010,00	3522,00	0,00	0,00	1010,00	1,00	0,00
20-06	1289	66046,00	9,37	95377,00	30961,00	0,70	44,41	81395,00	1,00	23,24
20-07	1356	56339,00	4,40	70410,00	22723,00	0,60	24,98	69970,00	2,00	24,19
20-08	1314	60378,00	4,11	82295,00	26838,00	0,50	36,30	73630,00	1,00	21,95
20-09	1352	63559,00	19,59	85903,00	28903,00	0,60	35,15	80787,00	2,00	27,11
20-10	1367	56090,00	10,71	71141,00	24773,00	0,60	26,83	68660,00	2,00	22,41
20-11	1289	66060,00	9,52	88969,50	30895,00	0,50	34,68	80020,00	1,00	21,13
20-12	1356	56339,00	4,53	71311,40	22707,00	0,50	26,58	69958,00	2,00	24,17
20-13	1314	60475,00	7,03	80110,60	26794,00	0,60	32,47	72706,00	1,00	20,22
20-14	1352	63560,00	14,88	80903,10	28861,80	0,50	27,29	80159,00	2,00	26,12
20-15	1367	56090,00	9,57	70843,40	24759,10	0,60	26,30	68550,00	2,00	22,21

que o custo da solução ficou mais distante do ótimo do que as demais. Isto é justificado pela utilização dos recursos dos servidores por conteúdos que possuem requisições que, se atendidas com maiores custos por outros servidores, permitiriam o atendimento de requisições de outro conteúdo com custos menores, reduzindo de forma global os custos totais.

Tabela 5.3: Formulação Matemática I × RPA × Heurística Ótima por Período - para 30 servidores

Instâncias	Num Req	Solução Ótima	Tempo (s)	Solução Distribuída	Mensagens	Tempo (s)	Gap (%)	Heur. Ótima por período	Tempo (s)	Gap (%)
30-1	2190	2010,00	9,30	2010,00	6530,00	0,30	0,00	2010,00	3,00	0,00
30-2	1793	1996,00	4,75	1996,00	7831,00	0,20	0,00	1996,00	2,00	0,00
30-3	1845	2484,00	6,02	2484,00	7797,00	0,20	0,00	2484,00	3,00	0,00
30-4	1896	2526,00	5,17	2526,00	7840,00	0,20	0,00	2526,00	3,00	0,00
30-5	2384	2430,00	8,56	2430,00	7858,00	0,20	0,00	2430,00	3,00	0,00
30-06	2007	87312,00	46,97	112257,00	37176,00	1,00	28,57	104999,00	3,00	20,26
30-07	1963	82600,00	18,20	108046,00	37176,00	0,80	30,81	101629,00	4,00	23,04
30-08	2021	82370,00	26,57	103030,00	34096,00	0,90	25,08	99820,00	4,00	21,18
30-09	1991	80877,00	37,20	104749,00	34081,00	0,80	29,52	98719,00	4,00	22,06
30-10	1998	80030,00	16,74	101120,00	34089,00	0,50	26,35	99488,00	4,00	24,31
30-11	2007	87312,00	46,59	113369,10	37139,00	0,90	29,84	104786,00	4,00	20,01
30-12	1963	82600,00	17,00	108534,30	37159,00	0,70	31,40	101828,00	3,00	23,28
30-13	2021	82370,00	27,26	104274,70	34082,30	0,70	26,59	100212,00	3,00	21,66
30-14	1991	80877,00	34,68	102921,60	34065,00	0,90	27,26	98556,00	4,00	21,86
30-15	1998	80030,00	22.33	102093,20	34074,00	0,80	27,57	98398,00	4,00	22,95

Entretanto, mesmo apresentando alguns casos particulares onde a solução distribuída ficou mais distante da ótima, vale ressaltar que esta foi obtida a partir de uma heurística distribuída construtiva gulosa. Além disso, a Formulação Matemática I conhece todas as requisições da instâncias, o que não é suposição realista na prática. Já o algoritmo distribuído RPA, considera apenas as requisições originadas no período atual e possíveis

Tabela 5.4: Formulação Matemática I × RPA × Heurística Ótima por Período - para 50 servidores

Instâncias	Num Req	Solução Ótima	Tempo (s)	Solução Distribuída	Mensagens	Tempo (s)	Gap (%)	Heur. Ótima por período	Tempo (s)	Gap (%)
50-1	3660	2839,00	45,23	2839,00	8807,00	0,40	0,00	2839,00	12,00	0,00
50-2	3231	3674,00	42,73	3674,00	13091,00	0,30	0,00	3674,00	9,00	0,00
50-3	3534	3580,00	38,18	3580,00	13021,00	0,50	0,00	3580,00	12,00	0,00
50-4	3646	2464,00	44,61	2464,00	8807,00	0,30	0,00	2464,00	12,00	0,00
50-5	3762	3200,00	51,61	3200,00	10917,00	0,40	0,00	3200,00	12,00	0,00
50-06	3391	136961,00	84,30	177074,00	56808,00	1,60	29,29	170238,00	12,00	24,30
50-07	3329	130875,00	48,42	160020,00	56808,00	1,90	22,27	158361,00	11,00	21,00
50-08	3214	155719,00	253,32	223552,00	77377,00	2,40	43,56	199304,00	11,00	27,99
50-09	3303	160044,00	255,28	225977,00	77365,00	2,20	41,20	200063,00	12,00	25,00
50-10	3295	143799,00	128,35	188945,00	67080,00	2,40	31,40	181674,00	11,00	26,34
50-11	3391	136961,00	80,47	177459,40	56775,00	1,70	29,57	168987,00	12,00	23,38
50-12	3329	130875,00	47,09	160332,00	56795,10	1,60	22,51	157591,00	12,00	20,41
50-13	3214	155748,00	189,89	221007,10	77235,40	2,30	41,90	192748,00	11,00	23,76
50-14	3303	160044,00	190,00	229452,50	77198,80	2,60	43,37	195287,00	12,00	22,02
50-15	3295	143799,00	86,80	187936,30	67019,00	2,00	30,69	180073,00	13,00	25,23
Média Total		139,28			0,68	19,47		4,20	15,04	

não atendidas de períodos anteriores.

Pode-se também observar nestes resultados que o número de mensagens enviadas não cresce em relação ao tamanho das instâncias, mas sim em relação à dificuldade das mesmas (limitação de recursos disponíveis nos servidores). Assim, nas instâncias onde a folga entre os recursos disponíveis nos servidores e os necessários para o atendimento das requisições for grande, menos mensagens são enviadas. Entretanto, se esta folga for pequena, mais mensagens são enviadas para tentar atender todas as requisições.

Os resultados apresentados nesta seção, bem como o algoritmo distribuído RPA, a Formulação Matemática I e a heurística ótima por período, estão no artigo [10] aceito para publicação no XLIII Simpósio Brasileiro de Pesquisa Operacional.

5.2 Formulação Matemática II × RPA_Mod

Para considerar novas características de RDCs reais (e.g., mudança do estado dos enlaces da rede) e também aprimorar o algoritmo distribuído RPA, um outro algoritmo também foi proposto, o RPA_Mod. Com a inclusão dessas características, a Formulação Matemática I não pode ser mais utilizada para comparação, sendo então utilizada a Formulação Matemática II.

Para comparar a Formulação Matemática II com o algoritmo distribuído RPA_Mod, também foram realizados testes utilizando a mesma configuração de máquina, o mesmo resolvedor, a mesma biblioteca para troca de mensagens e as instâncias utilizadas na seção

5.1. Entretanto, desta vez as informações por período dos custos de transporte entre os servidores foram utilizadas. Desse modo, apenas as informações de final de tempo de vida de conteúdos não foram utilizadas, considerando que todos os conteúdos estão disponíveis a partir do período de sua origem até o final da simulação.

Para realizar uma comparação mais justa, nas instâncias da Formulação Matemática II, foi adicionado mais um servidor que contém todos os conteúdos, mas com nenhuma banda para atendimento. Isto foi necessário, devido a existência das restrições (3.15) e (3.16) da formulação que exigem a existência de pelo menos uma réplica de conteúdo na RDC. Sem este servidor, a solução da formulação poderia ser prejudicada, uma vez que se não existir mais requisições para um determinado conteúdo, este tenha que continuar replicado, impedindo a utilização de recursos para outras réplicas que poderiam diminuir os custos totais. No algoritmo distribuído, isto é satisfeito pela existência do Servidor Central.

Os resultados encontrados nos teste estão nas Tabelas 5.5, 5.6, 5.7 e 5.8. A primeira coluna possui o nome da instância. As duas próximas colunas possuem o custo total e o custo de replicação da solução ótima obtida através da Formulação Matemática II. As próximas quatro colunas contêm o custo total, custo de replicação, número de mensagens e tempo global da solução encontrada com o RPA_Mod. As duas últimas colunas, apresentam os *gaps* em relação ao custo total e também ao custo de replicação. Em todas as simulações realizadas, o tamanho máximo das mensagens foi de 64 *bytes* (08 variáveis *double*).

Tabela 5.5: Formulação Matemática II × RPA_Mod - para 10 servidores

Instâncias	Num Req	Solução Ótima		Solução Distribuída				Gap (%)	
		Custo Total	Custo Repli	Custo Total	Custo Repli	Mensagens	Tempo Global	Custo Total	Custo Repli
10-1	613	5260864,00	540	5295312,40	540	1889,1	2	0,65	0,00
10-2	526	5995209,00	860	6016235,40	860	2714,8	2	0,35	0,00
10-3	519	5887866,00	751	5921011,10	751	2276,8	2	0,56	0,00
10-4	632	7989603,00	760	8002396,20	760	2713,4	2	0,16	0,00
10-5	445	6225835,00	588	6245160,73	588	2258,8	2	0,31	0,00
10-06	627	714439143,00	38302	730417613,26	38302	12789,8	2	2,24	0,00
10-07	675	714581914,00	33702	742397625,50	33702	11784	2	3,89	0,00
10-08	641	639981893,00	36776	643464077,48	36776	13720,2	2	0,54	0,00
10-09	659	618636768,00	38716	669638928,81	38716	14188	2	8,24	0,00
10-10	649	702294001,00	40875	761442739,66	40875	15127,1	2	8,42	0,00
10-11	627	* 714544924,48	43016	754454822,88	48598,3	12883,1	6	5,59	12,98
10-12	675	* 714604087,56	34610	726301194,40	34782	11800,6	2	1,64	0,50
10-13	641	* 640062792,03	40665	645139140,52	41926	13788,7	4	0,79	3,10
10-14	659	* 618779238,46	45271	672573867,47	44070,3	14280,2	2	8,69	-2,65
10-15	649	* 702435116,85	47169	727331834,94	46343,6	15244,1	4	3,54	-1,75

O algoritmo RPA_Mod apresentou resultados satisfatórios ficando, em média, 1,33%

do ótimo. Observe que, mesmo sendo uma porcentagem pequena, esta pode representar uma grande diferença de custos, uma vez que a solução ótima possui muitos dígitos. O custo total é formado em grande parte pelo custo de *backlog*, seguido pelo custo de transporte e replicação. Deste modo, pode-se concluir que a diferença encontrada é, na maioria, causada pelo custo de transporte. Isto fica mais evidente analisando o *gap* em relação ao custo de replicação. Este, em média, é -0,17%, mostrando que a solução distribuída realiza menos replicação que a solução ótima. Justificando que a diferença entre as soluções é devida, na maioria, ao custo de transporte.

Tabela 5.6: Formulação Matemática II × RPA_Mod - para 20 servidores

Instâncias	Num Req		Solução Ótima Custo Total	Solução Ótima Custo Repli	Solução Distribuída Custo Total	Solução Distribuída Custo Repli	Solução Distribuída Mensagens	Solução Distribuída Tempo Global	Gap (%) Custo Total	Gap (%) Custo Repli
20-1	715		11676179,00	760	11787485,00	760	3285	2	0,95	0,00
20-2	1525		25470098,00	1280	25475006,00	1280	3844,6	2	0,02	0,00
20-3	1602		20795036,00	1372	20819276,00	1372	4506,8	2	0,12	0,00
20-4	1224		19012702,00	1488	19071243,60	1488	4706,3	2	0,31	0,00
20-5	1366		19614181,00	1010	19629643,00	1010	3780	2	0,08	0,00
20-06	1289		1385507913,00	95377	1413940612,91	95377	31811,1	2	2,05	0,00
20-07	1356		1437476593,00	70410	1456126928,35	70410	23747,1	2	1,30	0,00
20-08	1314		1827464729,00	82295	1852806150,31	82295	27767,6	2	1,39	0,00
20-09	1352		1511384285,00	85903	1551812674,83	85903	29777,6	2	2,67	0,00
20-10	1367		1484408787,00	71141	1489793930,55	71141	25671,5	2	0,36	0,00
20-11	1289	*	1386095411,30	110023	1445042651,34	111266	32048	8	4,25	1,13
20-12	1356	*	1437577588,60	76331	1450530430,55	75087	23841,2	2	0,90	-1,63
20-13	1314	*	1827710902,10	93086	1847749683,99	95009	27992,4	2,2	1,10	2,07
20-14	1352	*	1511673978,90	99742	1574211739,99	96146,6	29990,4	4	4,14	-3,60
20-15	1367	*	1484502630,60	76787	1490727075,29	72713	25713,2	4	0,42	-5,31

Também é possível observar que nas instâncias fáceis e intermediárias, não existe *gap* entre o custos de replicação. Isto acontece porque nestas instâncias existe capacidade de armazenamento em todos os servidores para todos os conteúdos. Assim, é mais vantajoso replicar sempre localmente e atender com um menor custo de transporte.

Nas Tabelas 5.5, 5.6, 5.7 e 5.8, as linhas que possuem um '*' significam que a formulação encontrou uma solução ótima dentro da tolerância padrão do *CPLEX* 11.2, que é $1e-6$ para a diferença entre a melhor solução inteira e o valor ótimo da relaxação do LP (tolerância absoluta) e 0.0001 para a tolerância relativa.

Nestes resultados, também é possível observar, assim como no RPA, o número de mensagens enviadas não cresce em relação ao tamanho da instância (número de servidores e requisições). O número de mensagens está relacionado com a dificuldade da instância. Isto é, em relação aos recursos disponíveis (armazenamento e banda) nos servidores. Dessa forma, se os recursos disponíveis nos servidores for maior que os recursos necessários para atender as requisições, menos mensagens são enviadas. Caso estes recursos sejam

Tabela 5.7: Formulação Matemática II × RPA_Mod - para 30 servidores

Instâncias	Num Req		Solução Ótima Custo Total	Solução Ótima Custo Repli	Solução Distribuída Custo Total	Solução Distribuída Custo Repli	Solução Distribuída Mensagens	Solução Distribuída Tempo Global	Gap (%) Custo Total	Gap (%) Custo Repli
30-1	2190		35828992,00	2010	35843200,00	2010	6961,4	2	0,04	0,00
30-2	1793		25491826,00	1996	25551019,90	1996	8237,6	2	0,23	0,00
30-3	1845		36027964,00	2484	36080662,60	2484	8351,9	2	0,15	0,00
30-4	1896		30485266,00	2526	30541989,40	2526	8336,2	2	0,19	0,00
30-5	2384		32640948,00	2430	32692591,95	2430	8289,7	2	0,16	0,00
30-06	2007		2515010389,00	112257	2532584578,64	112257	38718,1	2	0,70	0,00
30-07	1963		2176171902,00	108046	2188827140,78	108046	38230,2	2	0,58	0,00
30-08	2021		2268449317,00	103030	2287275487,00	103030	35472	2	0,83	0,00
30-09	1991		2632095921,00	104749	2667034579,93	104749	35317,1	2	1,33	0,00
30-10	1998		2287937137,00	101120	2318048362,58	101120	35326,2	2	1,32	0,00
30-11	2007	*	2515187350,20	118773	2524434915,14	118095	38876,8	2	0,37	-0,57
30-12	1963	*	2176233795,40	112281	2189023957,59	107281	38264,4	4	0,59	-4,45
30-13	2021	*	2268524582,40	109396	2282885184,74	112664	35589,7	4	0,63	2,99
30-14	1991	*	2632167699,60	108597	2647787585,63	103771	35351	4	0,59	-4,44
30-15	1998	*	2287997279,30	107052	2340314945,48	110540	35438	4	2,29	3,26

Tabela 5.8: Formulação Matemática II × RPA_Mod - para 50 servidores

Instâncias	Num Req		Solução Ótima Custo Total	Solução Ótima Custo Repli	Solução Distribuída Custo Total	Solução Distribuída Custo Repli	Solução Distribuída Mensagens	Solução Distribuída Tempo Global	Gap (%) Custo Total	Gap (%) Custo Repli
50-1	3660		49656161,00	2839	49773440,30	2839	9437,2	2	0,24	0,00
50-2	3231		48441249,00	3674	48703996,20	3674	13371,8	2	0,54	0,00
50-3	3534		68863770,00	3580	68940963,00	3580	13522,5	2	0,11	0,00
50-4	3646		50010688,00	2464	50034598,00	2464	9442,8	2	0,05	0,00
50-5	3762		55802654,00	3200	55868576,90	3200	10127,4	2	0,12	0,00
50-06	3391		4951676071,00	177074	4959232683,84	177074	58885,8	2	0,15	0,00
50-07	3329		4283881456,00	160020	4289145457,87	160020	58911,1	2	0,12	0,00
50-08	3214		4327189712,00	223552	4351598558,37	223552	78583,6	2	0,56	0,00
50-09	3303		5292600515,00	225977	5306202420,37	225977	79119,5	2	0,26	0,00
50-10	3295		5040165310,00	188945	5052405371,00	188945	68829	2	0,24	0,00
50-11	3391	*	4951840962,10	186663	4968269833,74	186361	59041,3	4	0,33	-0,16
50-12	3329	*	4283935518,70	162845	4287195550,27	162042	58959,8	2	0,08	-0,49
50-13	3214	*	4328315298,80	252467	4361498362,46	238074,2	79115,6	4	0,77	-5,70
50-14	3303	*	5293650554,80	265089	5316450785,44	256053,8	79766,1	4	0,43	-3,41
50-15	3295	*	5040465298,30	206993	5057548834,28	202699	69086,6	4	0,34	-2,07
Média Total									1,33	-0,17

próximos, mais mensagens são enviadas.

Vale ressaltar, assim como mencionado no final da seção 5.1, que a Formulação Matemática II possui o conhecimento de todas as informações em todos os períodos de tempo. Portanto, mesmo sendo importante para avaliar a RDC em um dado momento, não representa a realidade. Ademais, existe um grande custo de centralização das informações, sendo na maioria das vezes inviável. Então, é razoável a diferença entre os custos das soluções, visto que na abordagem distribuída os servidores só têm o conhecimento das suas requisições do período atual e não conhecem os recursos disponíveis dos demais servidores.

5.3 RPA_Mod × Presti

Antes de apresentar os resultados das comparações realizadas, é necessário detalhar mais a proposta de solução do trabalho relacionado apresentado em [22, 19], aqui chamado de Presti.

Como brevemente mencionando na Seção 2.3 do capítulo de trabalhos relacionados, em [22, 19] uma heurística distribuída é apresentada para replicação e remoção de conteúdos e um mecanismo de distribuição de tráfego é proposto para a atribuição de requisições de clientes a servidores. Algumas considerações assumidas neste trabalho relacionado são importantes para o seu entendimento. São elas:

- A RDC é formada por servidores de acesso e de réplicas.
- Cada servidor de réplica possui um número limite de réplicas que pode armazenar (limite de espaço).
- Cada réplica pode atender 100% de uma requisição (limite de banda).
- A distância entre um servidor de acesso e um servidor de réplica representa a qualidade percebida pelo usuário.
- Cada servidor de réplica conhece o conjunto de servidores de acesso que estão a no máximo uma distância d_{max} (número de saltos) dele e o conjunto de servidores de réplicas que estão a no máximo d_{max} de seus servidores de acesso vizinhos (a uma distância $\leqslant d_{max}$).

A função para replicar um conteúdo é chamada por um servidor j de réplica quando a carga de uma de suas réplicas de c ultrapassa um determinado limiar $Umax$. A função

retorna o servidor onde a réplica deve ser adicionada. Quando a função é chamada, o servidor j primeiro calcula o número de requisições do conteúdo em questão que ele atualmente atende. Se a carga média das réplicas, definida como o número de requisições sendo atendidas sobre o número de réplicas alocadas deste conteúdo, for maior que o limiar $Umax$, então uma nova réplica do conteúdo será adicionada na rede com o objetivo de diminuir esta carga. A localização j' da nova réplica é escolhida baseada no número de requisições atualmente atendidas por réplicas armazenadas localmente que poderiam ter sua carga diminuída para a nova réplica a ser adicionada. Quanto maior este número, mais adequado j' para armazenar a nova réplica.

Se um servidor adequado $best_vr$ para armazenar a nova réplica é encontrado, o nó j envia uma mensagem para este solicitando a adição da réplica do conteúdo. O nó j também calcula a quantidade máxima de requisições que atualmente atende que podem ser transferidas para $best_vr$ sem sobrecarregar a nova réplica. Este então verifica se as requisições que j está servindo e que não podem ser transferidas para $best_vr$ ainda são muitas, ou seja, se elas ainda sobrecarregam as réplicas do nó j. Em caso afirmativo, uma nova réplica é adicionada a rede e o procedimento é reexecutado sobre as réplicas que não puderam ser transferidas.

Pode acontecer de todo o sistema ou uma parte dele estar sobrecarregado e nenhum servidor onde a réplica possa ser adicionada seja identificado. Neste caso, o algoritmo não pode melhorar a situação e o procedimento é finalizado.

Uma réplica pode ser removida se e somente se esta não está atendendo nenhuma requisição. Para evitar situações na qual a diminuição de tráfego temporária dispare a remoção de uma réplica que será necessária rapidamente de novo, a remoção de réplicas é baseada na média exponencial do passado da réplica e carga atual. Ou seja, uma réplica é removida somente quando não estiver atendendo requisições por um tempo suficiente o bastante para levar a média exponencial para zero.

Para realizar o direcionamento de tráfego com balanceamento de carga de um determinado conteúdo, é formulado um problema de otimização que utiliza informações sobre o número de requisições existentes para este conteúdo e a configuração de réplicas (número de réplicas nos servidores). Os autores afirmam que este problema convexo simples com restrições lineares pode ser resolvido através de técnicas padrões, e que considerariam o método de projeção de gradiente. Os autores também destacam que a solução para este problema pode ser encontrada de maneira distribuída.

Conhecendo agora mais em detalhes o trabalho relacionado, pode-se apresentar as

comparações feitas entre os trabalhos. Para realizar uma comparação justa, foram realizadas duas comparações. Na primeira, a solução algorítmica proposta em [22, 19] foi adaptada e executada com as considerações assumidas aqui neste trabalho. Ou seja, não há restrição de vizinhança, podendo uma requisição ser atendida por qualquer um dos servidores. Na segunda, o algoritmo distribuído RPA_Mod foi adaptado e executado com as considerações assumidas pelo trabalho relacionado. Ou seja, existe restrição de vizinhança, podendo uma requisição ser atendida apenas por servidores a uma distância determinada. Em ambas as comparações as instâncias foram as mesmas utilizadas nas seções anteriores. Desta forma, um mapeamento entre as considerações assumidas e as informações presentes nas instâncias foi realizado.

- Todos os servidores são considerados servidores de acesso e réplica, uma vez que as requisições dos clientes mais próximos já se encontram nestes.
- A capacidade de armazenamento é considerada ao invés do número máximo de réplicas possíveis armazenadas nos servidores.
- Cada servidor possui no máximo uma réplica de um determinado conteúdo. A banda disponível é utilizada para determinar o número de requisições que podem ser atendidas, e não o número de réplicas.
- Dois servidores são considerados vizinhos se o custo de transporte entre eles é $\leqslant$ d_{max}, agora dado em custo.
- Para fins de simplificação, os custos de *backlog* e custos de transporte variantes no tempo não serão considerados.
- Uma réplica é considerada sobrecarregada se o servidor onde esta se encontra estiver com a capacidade de banda ultrapassada.

Os resultados desta primeira comparação estão nas Tabelas 5.9, 5.10, 5.11 e 5.12. A primeira coluna contém o nome da instância. As três próximas colunas possuem os custos de replicação, de transporte e a carga dos servidores da heurística distribuída proposta em [22, 19]. As próximas cinco colunas contêm custos de replicação, de transporte, número de mensagens, tempo global e a carga dos servidores do RPA_Mod. A carga dos servidores é calculada sobre a média da carga de cada servidor, sendo a carga de cada servidor calculada sobre a média da carga de cada período.

Os resultados mostram que, a heurística apresentada em [22, 19] replica menos e possui um balanceamento de carga melhor, já que seu objetivo é minimizar o número

de réplicas e balancear a carga entre os servidores. Entretanto, o algoritmo distribuído RPA_Mod possui um custo de transporte significativamente menor, visto que seu objetivo é minimizar os custos de transporte e replicação. Pode-se concluir que, o RPA_Mod obteve resultados satisfatórios, ressaltando que este não necessita das informações de seus vizinhos e apresentou, em média, uma carga superior de apenas 4% e no máximo de 6,6%.

Os gráficos das figuras 5.1a, 5.2a, 5.3a e 5.4a ilustram melhor a diferença entre as cargas. Nestes gráficos, pode-se observar que o RPA_Mod apresenta cargas maiores do que as cargas obtidas com a heurística Presti para todas as instâncias, entretanto relativamente próximas. Um outro ponto interessante é que as cargas não variam em relação ao tamanho das instâncias (número de servidores e requisições).

Tabela 5.9: Primeira comparação entre Presti × RPA_Mod - para 10 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
10-1	613	54	2417973	0,04	540	0	1960	0	0,07
10-2	526	92	2104956	0,04	860	0	2815	0	0,06
10-3	519	83	2931390	0,04	751	0	2355	0	0,06
10-4	632	77	2273313	0,04	760	0	2819	0	0,08
10-5	445	71	1822620	0,03	588	0	2349	0	0,05
10-06	627	4034	64630077	0,16	38302	0	12948	0	0,19
10-07	675	4614	60202959	0,17	33702	0	11922	0	0,21
10-08	641	4612	52466520	0,16	36776	0	13892	0	0,19
10-09	659	4162	52067709	0,16	38716	0	14369	0	0,19
10-10	649	5443	67410726	0,16	40875	0	15328	0	0,19
10-11	627	4034	64630077	0,16	34247	3151746	12960	2	0,19
10-12	675	4614	60202959	0,17	33362	466137	11938	2	0,21
10-13	641	4612	52466520	0,16	34644	2479209	13928	2	0,19
10-14	659	4162	52067709	0,16	33962	3246138	14398	2	0,19
10-15	649	5443	67410726	0,16	36215	3953214	15305	8	0,19

Os resultados da segunda comparação são apresentados nas Tabelas 5.13, 5.14, 5.15 e 5.16, que possuem a mesma estrutura das tabelas anteriores. Para esta comparação, o valor de d_{max} para a definição da vizinhança foi encontrado através de experimentos, de forma a não tornar as instâncias inviáveis. Também nesta comparação, o algoritmo distribuído RPA_Mod foi modificado para tentar realizar um balanceamento de carga, passando a atender requisições localmente somente se a carga do servidor em questão for menor que 70%, visto que o custo de transporte não é relevante neste caso.

Os resultado mostram que, a heurística apresentada em [22, 19] continua replicando menos e possuindo um balanceamento de carga melhor. Entretanto, o RPA_Mod continua possuindo um menor custo de transporte, mesmo que este não esteja sendo considerado relevante nesta comparação.

Pode-se observar também, que a carga média dos servidores encontrada nas simu-

Tabela 5.10: Primeira comparação entre Presti × RPA_Mod - para 20 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
20-1	715	54	2417973	0,04	760	0	3380	0	0,05
20-2	1525	92	2104956	0,04	1280	0	4000	0	0,10
20-3	1602	83	2931390	0,04	1372	0	4676	0	0,09
20-4	1224	77	2273313	0,04	1488	0	4872	0	0,07
20-5	1366	71	1822620	0,03	1010	0	3917	0	0,08
20-06	1289	4034	64630077	0,16	95377	0	32180	0	0,20
20-07	1356	4614	60202959	0,17	70410	0	24029	0	0,22
20-08	1314	4612	52466520	0,16	82295	0	28106	0	0,20
20-09	1352	4162	52067709	0,16	85903	0	30139	0	0,20
20-10	1367	5443	67410726	0,16	71141	0	25982	0	0,20
20-11	1289	4034	64630077	0,16	73554	12910287	32202	6	0,20
20-12	1356	4614	60202959	0,17	66913	2213490	24067	2	0,22
20-13	1314	4612	52466520	0,16	74966	5353704	28203	2	0,20
20-14	1352	4162	52067709	0,16	76040	6538767	30242	2	0,20
20-15	1367	5443	67410726	0,16	68481	1669359	26018	2	0,20

Tabela 5.11: Primeira comparação entre Presti × RPA_Mod - para 30 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
30-1	2190	103	12625371	0,06	2010	0	7243	0	0,09
30-2	1793	94	7474366	0,04	1996	0	8557	0	0,07
30-3	1845	99	10565370	0,05	2484	0	8669	0	0,09
30-4	1896	92	10278405	0,04	2526	0	8627	0	0,07
30-5	2384	159	9601173	0,06	2430	0	8580	0	0,09
30-06	2007	22227	231577261	0,17	112257	0	39167	0	0,21
30-07	1963	12751	220849261	0,16	108046	0	38680	0	0,19
30-08	2021	13775	238512963	0,17	103030	0	35866	0	0,21
30-09	1991	18021	226818368	0,16	104749	0	35746	0	0,21
30-10	1998	12822	214843147	0,16	101120	0	35748	0	0,20
30-11	2007	22227	231577261	0,17	106943	3602958	39241	2	0,21
30-12	1963	12751	220849261	0,16	106733	1465113	38718	2	0,19
30-13	2021	13775	238512963	0,17	101898	1571196	35897	2	0,21
30-14	1991	18021	226818368	0,16	100724	1778748	35762	2	0,21
30-15	1998	12822	214843147	0,16	99884	1487289	35771	2	0,20

Tabela 5.12: Primeira comparação entre Presti × RPA_Mod - para 50 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
50-1	3660	186	19108350	0,05	2839	0	9797	0	0,08
50-2	3231	188	19001694	0,04	3674	0	13866	0	0,07
50-3	3534	252	20280583	0,05	3580	0	14052	0	0,09
50-4	3646	174	19198859	0,05	2464	0	9797	0	0,08
50-5	3762	252	24480714	0,06	3200	0	10450	0	0,08
50-06	3391	29392	484890245	0,18	177074	0	59574	0	0,22
50-07	3329	23896	438785796	0,16	160020	0	59576	0	0,20
50-08	3214	32713	430854775	0,16	223552	0	79519	0	0,19
50-09	3303	31832	441980359	0,16	225977	0	80049	0	0,21
50-10	3295	31521	437402339	0,16	188945	0	69655	0	0,20
50-11	3391	29392	484890245	0,18	170862	4256706	59650	2	0,22
50-12	3329	23896	438785796	0,16	159235	1004151	59606	2	0,20
50-13	3214	32713	430854775	0,16	178258	24415845	79810	2	0,19
50-14	3303	31832	441980359	0,16	184067	21466779	80337	2	0,21
50-15	3295	31521	437402339	0,16	179595	6984579	69787	2	0,20
Média Total		9195	134930940	0,12	67591	1833590			0,16

Tabela 5.13: Segunda comparação entre Presti × RPA_Mod - para 10 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
10-1	613	162	1554041,12	0,04	540	41127	1957	1	0,06
10-2	526	172	1550762,17	0,04	806	151359	2790	2	0,05
10-3	519	184	1731698,68	0,04	719	91770	2355	2	0,06
10-4	632	183	1733873,88	0,04	760	203454	2842	2	0,07
10-5	445	195	933724,49	0,03	572	24546	2285	2	0,05
10-06	627	10824	37649681,93	0,16	35326	8692653	12546	6	0,18
10-07	675	9764	44367110,45	0,17	32406	8389851	11765	3	0,20
10-08	641	10517	36278765,69	0,16	34231	8328798	13455	4	0,17
10-09	659	10591	37455835,70	0,16	34729	7333470	14047	5	0,18
10-10	649	15365	32309669,04	0,16	37067	9230601	14724	4	0,18
10-11	627	10824	37649681,93	0,16	32542	9847590	12485	10	0,18
10-12	675	9764	44367110,45	0,17	32046	9195570	11743	3	0,19
10-13	641	10517	36278765,69	0,16	32048	9399990	13414	4	0,17
10-14	659	10591	37455835,70	0,16	31709	8763336	14059	5	0,18
10-15	649	15051	33266315,49	0,16	34701	10087113	14636	4	0,18

Tabela 5.14: Segunda comparação entre Presti × RPA_Mod - para 20 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
20-1	715	108	2033927,19	0,02	760	90615	3389	2	0,05
20-2	1525	256	5057020,33	0,06	1265	229023	4031	2	0,10
20-3	1602	374	4157492,21	0,05	1332	141186	4688	2	0,09
20-4	1224	285	3974382,00	0,05	1280	335514	4813	4	0,07
20-5	1366	220	4050957,53	0,04	990	90624	3931	1	0,08
20-06	1289	20060	87903911,83	0,17	81130	19893549	31649	8	0,19
20-07	1356	13375	96507516,31	0,18	69492	16736676	24064	3	0,21
20-08	1314	15855	90615449,31	0,17	73497	17293704	27649	6	0,19
20-09	1352	20274	88376222,86	0,17	76018	19629210	29567	5	0,19
20-10	1367	15662	83120345,62	0,16	66213	14971632	25706	5	0,20
20-11	1289	20060	88429582,71	0,17	70190	23989899	31278	15	0,19
20-12	1356	13375	96507516,31	0,18	65269	18298290	24074	3	0,21
20-13	1314	15855	90615449,31	0,17	68645	19385367	27710	8	0,20
20-14	1352	20274	88376222,86	0,17	68825	23077509	29572	5	0,19
20-15	1367	15662	83120345,62	0,16	65608	15301494	25709	5	0,20

lações se aproximaram um pouco mais. Novamente, o algoritmo distribuído RPA_Mod apresentou resultados satisfatórios, indicando uma melhora no balanceamento de carga com apenas uma modificação simples.

Os gráficos das figuras 5.1b, 5.2b, 5.3b e 5.4b ilustram melhor a diferença entre as cargas encontradas. É possível notar, novamente, que o RPA_Mod apresenta cargas maiores que o Presti. Entretanto, as cargas obtidas pelo RPA_Mod diminuíram em relação à primeira comparação, enquanto que as cargas obtidas pelo Presti praticamente continuam as mesmas.

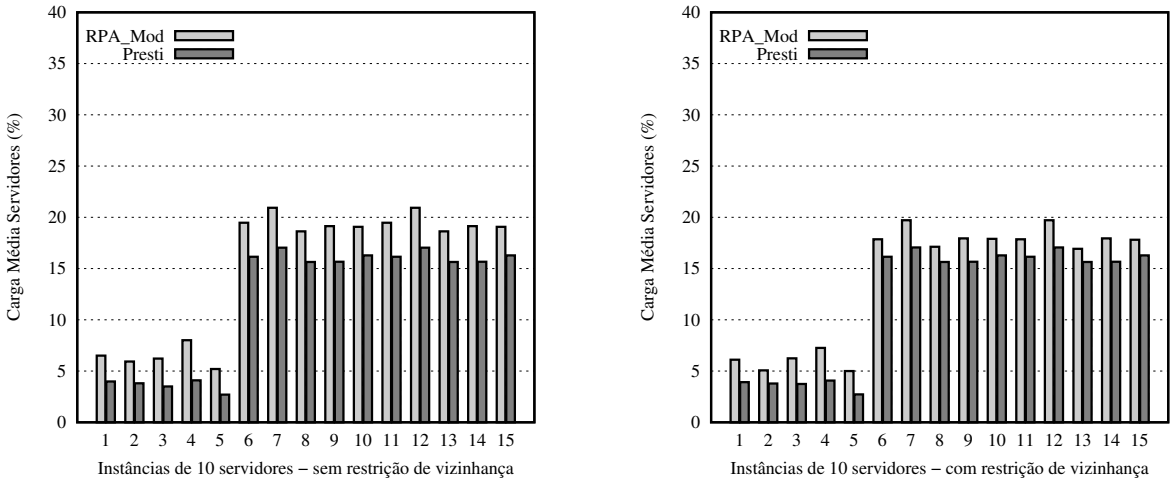
Nos gráficos das Figuras 5.1, 5.2, 5.3 e 5.4 também é possível visualizar a diferença entre as cargas médias entre as duas comparações realizadas. Os gráficos (a) apresentam as cargas médias da primeira comparação feita, onde não existe restrição de vizinhança e os gráficos (b) para a segunda comparação, onde existe restrição de vizinhança. Pode-se observar que, na segunda comparação, o RPA_Mod com o mecanismo simples de balanceamento de carga obteve cargas menores, ficando mais próximo das cargas obtidas por Presti.

Tabela 5.15: Segunda comparação entre Presti × RPA_Mod - para 30 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
30-1	2190	419	6838183,26	0,06	1852	552582	7220	3	0,08
30-2	1793	474	4438240,21	0,04	1738	438879	8424	3	0,06
30-3	1845	635	5661406,64	0,05	2290	520956	8134	4	0,08
30-4	1896	570	5562869,92	0,04	2244	562119	8446	4	0,07
30-5	2384	530	6201761,96	0,06	2233	669870	8574	3	0,08
30-06	2007	25018	131896504,72	0,17	104415	23441139	38718	5	0,20
30-07	1963	25307	125146975,94	0,16	101919	19456296	38357	4	0,19
30-08	2021	21787	138733004,72	0,17	97091	20290428	35481	5	0,20
30-09	1991	21075	132274756,54	0,16	98097	21061818	35326	7	0,20
30-10	1998	21802	126597973,27	0,16	95254	20895285	35360	5	0,18
30-11	2007	25018	131896504,72	0,17	99416	25641480	38742	5	0,20
30-12	1963	25307	125146975,94	0,16	98411	20835075	38383	4	0,19
30-13	2021	21787	138733004,72	0,17	94417	21152436	35485	5	0,20
30-14	1991	21075	132274756,54	0,16	95174	22076874	35259	7	0,19
30-15	1998	21802	126597973,27	0,16	94554	21296730	35363	5	0,18

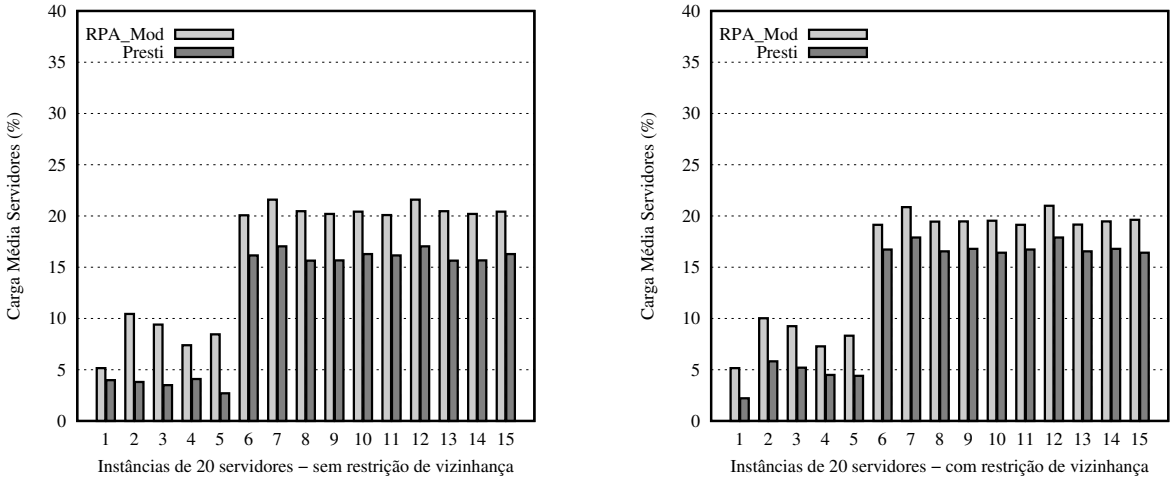
Tabela 5.16: Segunda comparação entre Presti × RPA_Mod - para 50 servidores

Instâncias	Num Req	Solução Presti			Solução RPA_Mod				
		Custo Repli	Custo Trans	Carga	Custo Repli	Custo Trans	Mensagens	Tempo Global	Carga
50-1	3660	675	8909243,80	0,05	2751	426906	9831	2	0,07
50-2	3231	1015	7157471,45	0,04	3366	522762	13815	2	0,06
50-3	3534	905	8822379,81	0,05	3408	902961	13530	2	0,09
50-4	3646	724	7246838,50	0,05	2440	223833	9807	2	0,08
50-5	3762	899	10327717,34	0,06	3200	343044	10539	2	0,08
50-06	3391	47622	186933053,03	0,18	166197	33295689	58597	5	0,20
50-07	3329	44672	172283019,37	0,16	151486	29149530	58825	5	0,19
50-08	3214	65140	172939253,67	0,16	198387	37995426	77192	9	0,18
50-09	3303	57722	183915971,74	0,16	195834	47347692	78468	9	0,19
50-10	3295	53291	167537636,05	0,16	174848	33340155	68113	5	0,19
50-11	3391	47622	186933053,03	0,18	159041	34350852	58674	5	0,20
50-12	3329	44672	172283019,37	0,16	151391	29515779	58800	5	0,19
50-13	3214	65140	172939253,67	0,16	168277	48426126	77007	17	0,18
50-14	3303	57722	183915971,74	0,16	173367	55420869	78058	14	0,19
50-15	3295	52993	167759579,61	0,16	165571	36094974	67997	5	0,19
Média Total		17664	74088893	0,12	62656	14758235			0,15



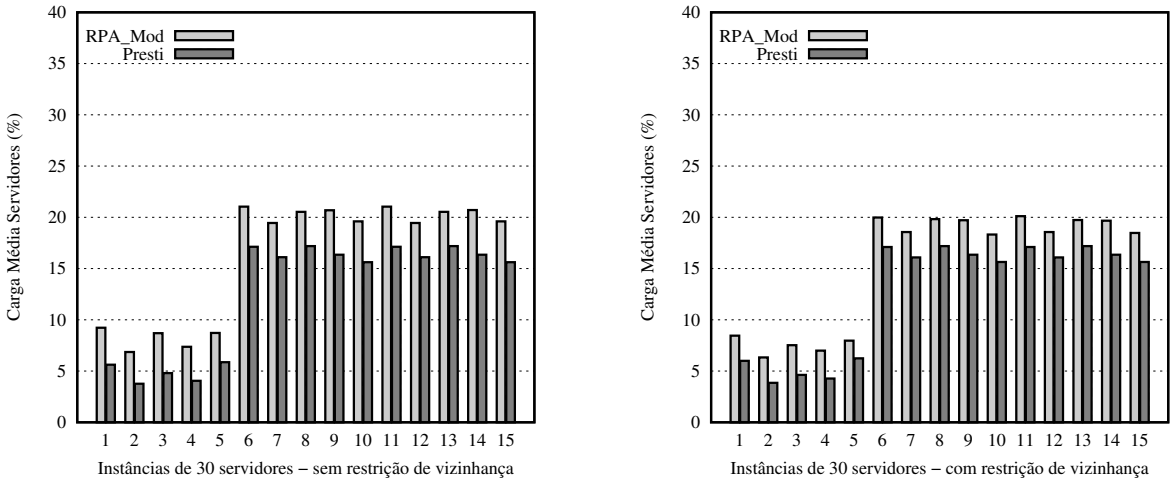
(a) (b)

Figura 5.1: Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 10 servidores.



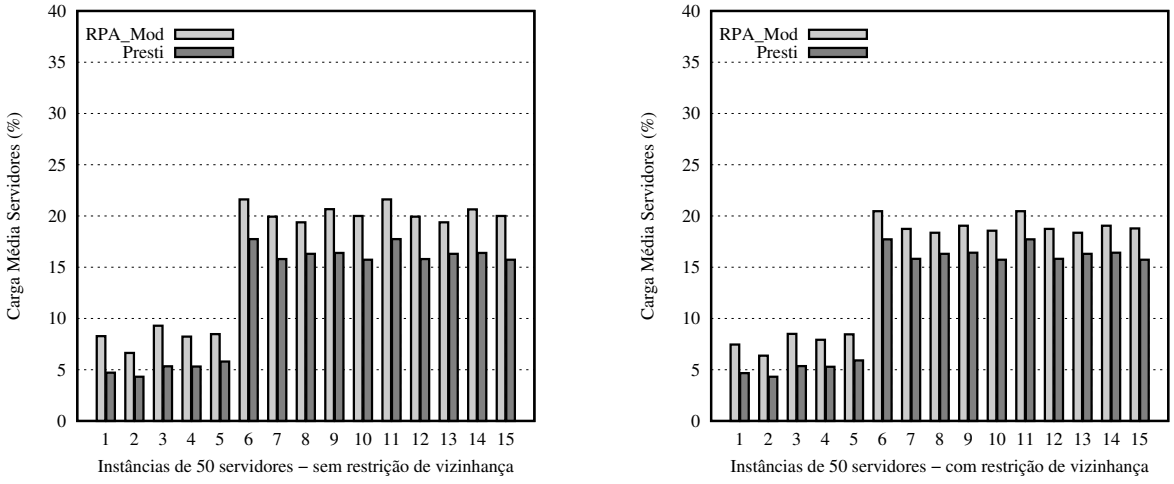
(a) (b)

Figura 5.2: Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 20 servidores.



(a) (b)

Figura 5.3: Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 30 servidores.



(a) (b)

Figura 5.4: Carga Média (a) sem restrição de vizinhança e (b) com restrição de vizinhança das instâncias com 50 servidores.

Capítulo 6

Conclusão e Trabalhos Futuros

Este trabalho aborda a versão dinâmica do Problema de Posicionamento de Réplicas em uma RDC, propondo uma formulação matemática para a solução ótima, uma heurística centralizada ótima por período e duas heurísticas distribuídas, RPA e RPA_Mod.

Foram consideradas duas abordagens para o PPR On-Line. Na primeira, algumas características das RDCs reais não foram consideradas, como por exemplo, penalização por atrasos de início de atendimento de requisições, mudanças no estado da rede e tempo de replicação. Já na segunda, essas características foram incorporadas ao PPR, tornando-o mais próximo da realidade. Com isso, o algoritmo distribuído preliminar RPA foi aprimorado, originando o algoritmo distribuído RPA_Mod. Para avaliar os algoritmos projetados, três comparações foram realizadas.

Na primeira, o RPA foi comparado com a Formulação Matemática I e com a heurística centralizada ótima por período. As simulações realizadas mostraram que o algoritmo distribuído RPA obteve resultados satisfatórios, com soluções em média 19.5% do ótimo, e no pior caso a 44.4% do ótimo. Entretanto, devido ao tamanho das mensagens enviadas e ao ponto de centralização (servidor central), um segundo algoritmo distribuído foi proposto, o RPA_Mod.

Na segunda comparação, testes foram realizados entre a Formulação Matemática II e o algoritmo distribuído RPA_Mod. Os resultados das simulações mostraram que o RPA_Mod apresentou resultados satisfatórios, ficando em média a 1,33% do ótimo.

Por fim, foram realizados testes entre o algoritmo distribuído RPA_Mod e a heurística distribuída proposta em [22, 19], aqui chamada de Presti. Este trabalho relacionado foi escolhido para comparação por ser o mais recente da bibliografia pesquisada e por assumir considerações para o PPR próximas as feitas nesta dissertação. Para realizar uma com-

paração justa, o trabalho relacionado foi simulado em um cenário com as considerações assumidas neste trabalho para o algoritmo distribuído RPA_Mod, assim como também o RPA_Mod foi simulado em um cenário com as considerações assumidas em [22, 19]. Em ambos as simulações, o Presti realiza um balanceamento de carga melhor. Entretanto, o RPA_Mod possui um menor custo de transporte. Para a segunda simulação, foi adicionado ao RPA_Mod um mecanismo simples para tentar balancear a carga. Este mecanismo resultou em uma melhora de 1% na carga média dos servidores. Então, pode-se concluir que o algoritmo distribuído RPA_Mod apresentou resultados satisfatórios em ambas as simulações.

Como trabalhos futuros, tem-se três objetivos principais. O primeiro é avaliar os algoritmos distribuídos propostos com instâncias maiores (em termos de número de servidores, requisições e conteúdos) e assim, mais próximas da realidade. O segundo objetivo, conforme descrito na Seção 4.1, é simular os algoritmos propostos de maneira assíncrona, utilizando um servidor (processo) adicional para gerar e distribuir as requisições dos clientes, permitindo que as comparações realizadas no Capítulo 5 continuem sendo feitas de maneira justa. Por fim, o terceiro objetivo é estudar e pesquisar a possibilidade de utilizar os resultados de redes complexas para modelar e propor novos algoritmos distribuídos para o problema em uma RDC de larga escala (com milhares de nós).

Referências

[1] Aioffi, W., Mateus, G., Almeida, J., Loureiro, A. Dynamic content distribution for mobile enterprise networks. *IEEE Journal on Selected Areas in Communications 23*, 10 (2005), 2022–2031.

[2] Akamai. http://www.akamai.com, 2011. Acessado em 17 de abril.

[3] Bakiras, S., Loukopoulos, T. Combining replica placement and caching techniques in content distribution networks. *Computer Communications 28* (2005), 1062–1073.

[4] Barbosa, V. C. *An Introduction to Distributed Algorithms.* The MIT Press (September 1, 1996), Cambridge, Massachusetts, 1996.

[5] Bartolini, N., Presti, F. L., Petrioli, C. Optimal dynamic replica placement in content delivery networks. In *In Proc. IEEE Int. Conf. on Networking* (2003), p. 125–130.

[6] Bektas, T., Oguz, O., Ouveysi, I. Designing cost-effective content distribution networks. *Computers & Operations Research 34*, 8 (2007), 2436 – 2449.

[7] Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik 4* (1962), 238–252.

[8] Coppens, J., Wauters, T., Turck, F. D., Dhoedt, B., Demeester, P. Design and performance of a self-organizing adaptive content distribution network. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP* (2006), p. 534–545.

[9] CPLEX. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, 2011. Acessado em 17 de abril.

[10] de Paula Junior, U. C., Drummond, L. M. A., Frota, Y., Simonetti, L., Uchoa, E. Posicionamento de réplicas em redes de distribuição de conteúdos. In *XLIII Brazilian Symposium of Operational Research - A ser publicado* (2011).

[11] Frank, C., Römer, K. Distributed facility location algorithms for flexible configuration of wireless sensor networks. In *DCOSS'07: Proceedings of the 3rd IEEE international conference on Distributed computing in sensor systems* (Berlin, Heidelberg, 2007), Springer-Verlag, p. 124–141.

[12] Huang, C., Abdelzaher, T. Towards content distribution networks with latency guarantees. In *Twelfth IEEE International Workshop on Quality of Service. IWQOS* (2004), p. 181–192.

[13] Kurose, J. F., Ross, K. W. *Redes de Computadores e a Internet.* Addison-Wesley, 2006.

[14] LabIC. `http://labic.ic.uff.br/`, 2011. Acessado em 17 de abril.

[15] Leighton, F. T., Lewin, D. M. Global hosting system, 2000. US Patent: US006108703.

[16] MATLAB. `http://www.mathworks.com/products/matlab/`, 2011. Acessado em 17 de abril.

[17] Neves, T. A., de A. Drummond, L. M., Ochi, L. S., Albuquerque, C., Uchoa, E. Solving replica placement and request distribution in content distribution networks. *Electronic Notes in Discrete Mathematics 36* (2010), 89–96.

[18] Nygren, E., Sitaraman, R. K., Sun, J. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev. 44* (August 2010), 2–19.

[19] Presti, F. L., Petrioli, C., Vicari, C. Distributed dynamic replica placement and request redirection in content delivery networks. In *Proceedings of the 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (Washington, DC, USA, 2007), IEEE Computer Society, p. 366–373.

[20] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[21] Tenzakhti, F., Day, K., Ould-Khaoua, M. Replication algorithms for the world-wide web. *Journal of Systems Architecture 50*, 10 (2004), 591 – 605.

[22] Vicari, C., Petrioli, C., Presti, F. L. Dynamic replica placement and traffic redirection in content delivery networks. *SIGMETRICS Perform. Eval. Rev. 35* (December 2007), 66–68.

[23] Wauters, T., Coppens, J., De Turck, F., Dhoedt, B., Demeester, P. Replica placement in ring based content delivery networks. *Comput. Commun. 29*, 16 (2006), 3313–3326.

[24] Wauters, T., Coppens, J., Dhoedt, B., Demeester, P. Load balancing through efficient distributed content placement. In *Next Generation Internet Networks, 2005* (18-20 2005), p. 99 – 105.

[25] Zhou, X., Xu, C.-Z. Efficient algorithms of video replication and placement on a cluster of streaming servers. *Journal of Network and Computer Applications 30*, 2 (2007), 515 – 540.