Posicionamento de Réplicas em Redes de Distribuição de Conteúdos

Ubiratam Carvalho de Paula Junior, Lúcia M. A. Drummond, Yuri Frota, Luidi Simonetti
Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 3º andar - São Domingos - 24210-240 - Niterói - RJ
{upaula,lucia,yuri,luidi}@ic.uff.br

Eduardo Uchoa
Departamento de Engenharia de Produção – Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 4º andar - São Domingos - 24210-240 - Niterói - RJ
uchoa@producao.uff.br

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 uma heurística distribuída, considerando características mais realísticas do PPR do que as encontradas na literatura relacionada. Os resultados mostraram que a heurística distribuída proposta apresentou resultados satisfatórios para a versão dinâmica do problema quando comparada com a solução ótima global e a heurística ótima por período.

PALAVRAS CHAVE. Redes de Distribuição de Conteúdos, Problema de Posicionamento de Réplicas, Heurísticas Distribuídas

Área Principal: PO em Telecomunicações e Sistemas de Informações

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 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 a distributed heuristic are proposed, considering more realistic features of PPR than those found in related literature. The results showed that the proposed distributed heuristic presented satisfactory results for the dynamic version of the problem when compared with the global optimal solution and with the optimal heuristic for each period.

KEY WORDS. Content Distribution Network, Replica Placement Problem, Distributed Heuristics

Main area: OR on Telecommunications and Information Systems

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 a mesma 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* [1]. A estratégia de utilização de múltiplos servidores caracteriza uma Rede de Distribuição de Conteúdos (RDC), que consiste de uma rede sobreposta 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 dois problemas principais envolvendo uma RDC. O primeiro é o Problema de Posicionamento de Réplicas (PPR), que consiste em posicionar as réplicas dos conteúdos em seus servidores. O segundo 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 primeiro problema.

O objetivo deste artigo é apresentar uma solução para o PPR utilizando técnicas de programação distribuída e de otimização matemática, tratando características tais como: limitações de banda e armazenamento nos servidores 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 uma formulação matemática, uma heurística centralizada e um algoritmo distribuído para a versão dinâmica do PPR.

A justificativa para o desenvolvimento de um algoritmo distribuído 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.

A principal contribuição deste trabalho em relação aos demais presentes na literatura é tratar o problema de forma geral, considerando diversos requisitos da RDC simultaneamente. Os algoritmos encontrados na literatura são específicos para determinadas topologias, tratam apenas um conteúdo, ou são estáticos.

O restante do artigo está organizado da seguinte forma. Na Seção 2, alguns trabalhos relacionados são comentados. Na Seção 3, são apresentadas a definição do problema, a formulação matemática e a heurística centralizada. Na Seção 4, é apresentado o pseudocódigo do algoritmo distribuído e sua complexidade. Na Seção 5, são expostos os resultados experimentais para a formulação matemática, a heurística centralizada e o algoritmo distribuído introduzido na seção anterior. Finalmente, na Seção 6, uma conclusão é apresentada, resumindo as contribuições deste trabalho e apontando os trabalhos futuros.

2 Trabalhos Relacionados

Embora existam trabalhos centralizados para o PPR em RDC, como, por exemplo, Bartolini *et al*. (2003) e Bektas *et al*. (2007), nesta seção são descritos apenas os trabalhos relacionados que utilizam soluções distribuídas para o PPR e problemas correlatos.

Em Frank e Römer (2007), algoritmos centralizados e distribuídos são propostos para solucionar o problema da auto-configuração (por exemplo, eleição de líder) em redes de sensores através do Problema de Localização de Facilidades. Entretanto, estes algoritmos não podem ser

[1] http://www.akamai.com

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 Huang e Abdelzaher (2004), 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 o algoritmo proposto neste artigo é executado apenas uma vez para todos os conteúdos.

Em Aioffi *et al*. (2005), é considerado um modelo de RDC dinâmica e móvel, que considera variações de demandas para o posicionamento das réplicas. Entretanto, não são consideradas restrições de armazenamento e banda nos servidores. É considerado apenas um limite máximo de atendimentos a requisições externas.

Em Coppens *et al*. (2006), 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 Wauters *et al*. (2006), 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. O algoritmo proposto neste artigo considera um topologia de rede genérica, onde a única restrição é que todos os servidores formem um grafo conexo. Além disso, o objetivo do algoritmo proposto é minimizar os custos de replicação e transporte, e não realizar o balanceamento de carga no servidor central, como proposto neste trabalho relacionado.

Em Tenzakhti *et al*. (2004), 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. Neste artigo, como já mencionado anteriormente, não existe restrição quanto a topologia de rede utilizada, sendo necessário apenas a comunicação entre servidores da RDC.

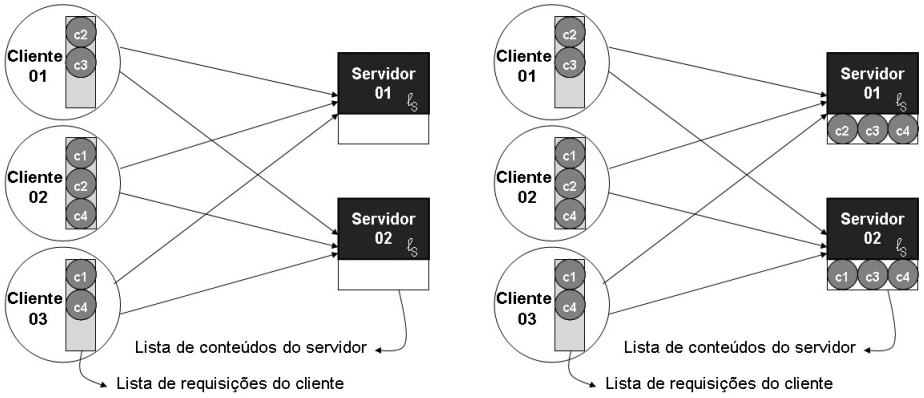

Figura 1: Exemplo do problema de posicionamento de réplicas

Em Leighton e Lewin (2000), 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 Neves *et al*. (2010), apresentando resultados insatisfatórios em relação ao problema. O algoritmo aqui proposto tem

como objetivo posicionar as réplicas em servidores de modo a minimizar os custos totais de replicação e comunicação.

Em Vicari *et al*. (2007), 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 artigo é, além de minimizar os custos de replicação, também minimizar os custos de comunicação dos atendimentos das requisições.

Neste artigo, uma formulação matemática, uma heurística centralizada e um algoritmo guloso distribuído são apresentados para a versão dinâmica do PPR em RDC. Na versão dinâmica, as demandas representam requisições em cada período. Um período pode ser um intervalo de tempo pequeno (segundos ou minutos) ou grande (horas ou dias), dependendo da aplicação do algoritmo.

3 Definição do Problema

O PPR em RDC consiste 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 disponíveis.

A Figura 1 ilustra um exemplo deste problema. Na Figura 1(a), tem-se os clientes com suas respectivas requisições de conteúdos e os servidores com os seus conjuntos de réplicas inicialmente vazios. Na Figura 1(b), tem-se os posicionamentos das réplicas dos conteúdos em cada servidor, de acordo com as requisições dos clientes e limitações de recursos.

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 as 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.

Neste trabalho é considerado que as requisições dos clientes já são conhecidas pelos seus servidores mais próximos.

3.1 Formulação Matemática

Para a 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.

A seguir é apresentada a formulação proposta.

Conjuntos

S	Conjunto de servidores, indexado por s.
R	Conjunto de requisições, indexado por r.
K	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).
$Cap(s)$	Capacidade de armazenamento do servidor s (Mb).
$Band(s)$	Capacidade de banda do servidor s (Mb/t).
$Tam(k)$	Tamanho do conteúdo k (Mb).

B_r	Banda necessária para a requisição r (Mb/t).
$Cont(r)$	Conteúdo da requisição r.
$Ori(r)$	Servidor de origem da requisição r.
$Temp(r)$	Período da requisição r.
τ_r	$\lceil Tam_{Cont(r)}/B_{Cont(r)} \rceil$.
$Tmax(r)$	$T_f - \tau_r$
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.

$$min \sum_{r \in R} \sum_{\substack{t \in T| \\ Temp(r) \leq t \\ t \leq Tmax(r)}} \sum_{s \in S} Tam(Cont(r)) C_{Ori(r)s} y_{rst} + \sum_{s \in S} \sum_{k \in K} \sum_{t \in T} Tam(k) f_{sk} x_{skt} \quad (1)$$

satisfazendo:

$$\sum_{t=Temp(r)}^{Tmax(r)} \sum_{s \in S} y_{rst} = 1, \forall r \in R \quad (2)$$

$$w_{skt} = \sum_{t'=0}^{t} (x_{skt'} - x'_{skt'}), \forall s \in S, \forall k \in F, \forall t \in T \quad (3)$$

$$y_{rst} \leq w_{skt} - \sum_{t'=t}^{t+\tau_r} x'_{skt}, \forall r \in R, \forall s \in S,$$
$$\forall t \in T | Temp(r) \leq t \leq Tmax(r), k = cont(r) \quad (4)$$

$$\sum_{k \in K} w_{skt}.Tam(k) \leq Cap(s), \forall s \in S, \forall t \in T \quad (5)$$

$$\sum_{r \in R} \sum_{t'=max(0, t-\tau_r+1)}^{t} y_{rst'}.B_{Cont(r)} \leq Band(s), \forall s \in S, \forall t \in T, \quad (6)$$

$$x_{skt} \in \{0,1\}, \forall s \in S, \forall k \in K, \forall t \in T \quad (7)$$

$$y_{rst} \in \{0,1\}, \forall r \in R, \forall s \in S, \forall t \in T | Temp(r) \leq t \leq Tmax(r) \quad (8)$$

A função objetivo (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 (2) garante que as requisições sejam atendidas em um intervalo de tempo válido. O conjunto de restrições (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 (4) garantem que as requisições sejam atendidas por servidores que possuem o conteúdo solicitado durante todo os períodos de atendimento. O conjunto de restrições (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 (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.

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 este a estar replicado, 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. 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.

4 Algoritmo Distribuído

Nesta seção é apresentado o algoritmo distribuído guloso proposto para o PPR dinâmico em RDC e sua análise de complexidade.

A abordagem distribuída é utilizada uma vez que a natureza do problema tratado é distribuída. 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.

4.1 Pseudocódigo

O algoritmo se inicia com cada servidor tentando atender localmente suas requisições do primeiro período (Algoritmos 1 e 7). 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 pseudocódigo do algoritmo distribuído proposto é descrito a seguir.

As estruturas de dados utilizadas são as seguintes:

$numServidores$	número de servidores.
$numConteudos$	número de conteúdos.
$numPeriodos$	número de períodos.
$conteudos$	conteúdos replicados localmente.
$numReq_t$	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.
$numReqExt_t$	número de requisições externas atendidas no 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.
$custoTransporte$	custo de transporte local total.

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

1 $periodo := 0$;
2 $atenderLocal(periodo)$; */*ver Algoritmo 7*/*
3 Enviar $STATIC$(*requisições não atendidas*, $bandaTotal$, $capTotal$) para o servidor central;

O servidor central, ao receber as mensagens $STATIC$ de cada servidor, armazena localmente as informações recebidas e após receber de todos, direciona estas requisições para serem atendidas externamente por outros servidores, podendo também determinar replicações de conteúdos nos servidores (Algoritmo 8). Por fim, o servidor central envia mensagens $STATIC$ 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 2).

Algoritmo 2: 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 $verificaAtendimentos()$; */*ver Algoritmo 8*/*
4 **para** *cada servidor s* **faça**
5 Enviar $STATIC$(*requisições locais atendidas,requisições externas atendidas*, $contedos$, $bandaTotal$, $capTotal$) para o servidor s;

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

1 $atualizaLocal()$;
2 $periodo := periodo + 1$;
3 $atenderLocal(periodo)$; */*ver Algoritmo 7*/*
4 $contMsg := 0$;
5 Enviar $CHECK$(*requisições não atendidas do conteúdo contMsg*) para o servidor com menor custo de transporte e que possua o conteúdo $contMsg$;

Os servidores, ao receberem a mensagem $STATIC$ do servidor central, atualizam as informações recebidas e passam para o próximo período de tempo. Assim, estes tentam atender localmente requisições que não foram atendidas no período passado mais as requisições que surgiram neste período. 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 (Algoritmo 3). Para isso, pressupõe-se que a RDC fornece um serviço de localização de conteúdos, similar ao *DNS*.

Um servidor ao receber mensagem $CHECK$ 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 das mensagens $CHECKED$ (Algoritmo 4).

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. Se a requisição não conseguir ser atendida em todos os próximos períodos, esta será atendida pelo servidor central com um custo mais alto.

Algoritmo 4: Ao receber uma mensagem $CHECK$(*requisições não atendidas do conteúdo contMsg*) do servidor s /* *Executado por um servidor* s_j */

```
1 atendeExternas();
2 Enviar CHECKED(requisições atendidas e não atendidas do conteúdo contMsg) para o servidor s;
```

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 as requisições não atendidas. Estas ações são descritas no Algoritmo 5 e são disparadas com o recebimento de uma mensagem $CHECKED$.

Algoritmo 5: Ao receber uma mensagem $CHECKED$(*requisições atendidas e não atendidas do conteúdo contMsg*) do servidor s /* *Executado por um servidor* s_j */

```
1  atualizaLocal();
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 6: 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 $ATTEND\_CENTRAL$, terá que atender as requisição do servidor remetente (Algoritmo 6).

Os algoritmos dos procedimentos $atualizaLocal()$ e $atendeExternas()$ não são apresentados devido a limitações de espaço. Todavia, uma breve descrição deles é feita a seguir.

O procedimento $atualizaLocal()$ realiza as seguintes ações: atualiza as requisições atendidas, armazenando os período de início e término de atendimento. Além disso, se o parâmetro *requisições externas atendidas* for diferente de NIL, a banda, capacidade, conteúdos replicados localmente são atualizados e são armazenadas as informações das requisições externas, como período de início e término de atendimento. Por fim, todas as requisições atendidas localmente ou externamente são percorridas, atualizando os recursos locais para as que já foram concluídas.

O procedimento $atendeExternas()$ realiza as seguintes ações: verifica se é possível atender as requisições externas solicitadas, atualizando os recursos locais, bem como as informações de início e término dos atendimentos realizados.

Algoritmo 7: atenderLocal(periodo) */* Executado por um servidor s */*

```
1  para p:=1 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                      bandaTotal := bandaTotal − bandReq_rt;
10             senão
11                 se capTotal ⩾ tamReq_rt e bandaTotal ⩾ bandReq_rt então
12                     Replicar localmente o conteúdo contReq_rt;
13                     iniReq_rt := p;
14                     fimReq_rt := p + ⌈tamReq_rt/bandReq_rt⌉;
15                     bandaTotal := bandaTotal − bandReq_rt;
16                     capTotal := capTotal − tamReq_rt;
17                     custoReplicacaoTotal := custoReplicacaoTotal + custoReplicacao_contReq_rt;
```

Algoritmo 8: 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;
```

4.2 Complexidades

Nesta seção, as complexidades do algoritmo distribuído proposto 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. Então, o número total de mensagens enviadas é: $(2 + 2 \times (numConteudos \times (numServidores - 1) \times (numPeriodos - 1)) \times numServidores)$. Desta forma, a complexidade de mensagens é $O((numConteudos \times (numServidores - 1) \times (numPeriodos - 1)) \times numServidores)$.

A complexidade de tempo global é definida como a maior cadeia de causalidade do tipo "recebeu uma mensagem e enviou outra em consequência"que ocorre na execução do algoritmo. No pior caso, para o atendimento de uma requisição é necessário percorrer todos os demais servidores. Dessa forma, a complexidade de tempo global para o atender uma requisição é $O(numServidores)$.

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 ao número de requisições locais atendidas mais o número de requisições externas atendidas no primeiro período de tempo. O maior tamanho da mensagens $CHECK$ é igual ao maior número de requisições de um mesmo conteúdo em todos os períodos. O tamanho da mensagem $ATTEND\_CENTRAL$ é definida pelo maior número 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 $ATTEND\_CENTRAL$.

5 Resultados Experimentais

Para avaliar a formulação matemática, a heurística centralizada e o algoritmo distribuído 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, foi utilizado o resolvedor $CPLEX$ 12.1 [2]. O algoritmo distribuído 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 Neves *et al*. (2010) e estão disponíveis em *LabIC* [3]. De acordo com Neves *et al*. (2010), as instâncias são divididas em três tipos: fáceis de resolver, intermediárias e difíceis. 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 estão na Tabela 1. A primeira coluna contém o nome da instância. A segunda e a terceira, o valor e o tempo da solução ótima. A quarta, quinta e sexta, contêm o valor, o número de mensagens e o tempo da simulação do algoritmo distribuído. 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 apresenta o *gap* entre a solução da heurística ótima por período e o ótimo. Em todas as simulações realizadas, o tamanho máximo das mensagens foi de 136 *bytes* (17 variáveis *double*). Além disso, para a formulação matemática e a heurística centralizada o início de atendimento das requisições foi fixado para o período que estas surgem na RDC.

O algoritmo distribuído 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 algoritmo distribuído 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 inst_20_06, inst_50_13 e inst_50_14, pode-se observar 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.

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

[2] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
[3] http://labic.ic.uff.br/

Tabela 1: Resultados Computacionais

Instâncias	Solução Ótima	Tempo (s)	Solução Distribuída	Mensagens	Tempo (s)	Gap (%)	Solução Ótima por período	Tempo (s)	Gap (%)
inst_10_1	540,00	0,30	540,00	1765,00	0,00	0,00	540,00	0,00	0,00
inst_10_2	860,00	0,25	860,00	2616,00	0,00	0,00	860,00	1,00	0,00
inst_10_3	751,00	0,24	751,00	2192,00	0,10	0,00	751,00	0,00	0,00
inst_10_4	760,00	0,31	760,00	2614,00	0,00	0,00	760,00	0,00	0,00
inst_10_5	588,00	0,19	588,00	2186,00	0,10	0,00	588,00	0,00	0,00
inst_10_6	30162,00	0,92	38302,00	12395,00	0,40	26,99	36871,00	1,00	22,24
inst_10_7	29303,00	0,95	33702,00	11363,00	0,20	15,01	35330,00	0,00	20,57
inst_10_8	29548,00	0,98	36776,00	13415,00	0,30	24,46	34811,00	0,00	17,81
inst_10_9	29107,00	1,73	38716,00	14441,00	0,10	33,01	35186,00	1,00	20,89
inst_10_10	32662,00	1,11	40875,00	15472,00	0,40	25,15	40887,00	0,00	25,18
inst_10_11	30162,00	0,95	37810,70	12387,60	0,10	25,36	36409,00	0,00	20,71
inst_10_12	29303,00	0,96	33844,20	11360,00	0,30	15,50	35378,00	1,00	20,73
inst_10_13	29548,00	0,99	36299,30	13404,40	0,30	22,85	34175,00	0,00	15,66
inst_10_14	29107,00	1,76	36005,70	14418,00	0,40	23,70	35407,00	1,00	21,64
inst_10_15	32681,00	1,26	39982,60	15456,00	0,30	22,34	40508,00	0,00	23,95
inst_20_1	760,00	1,19	760,00	3510,00	0,00	0,00	760,00	1,00	0,00
inst_20_2	1280,00	3,04	1280,00	3506,00	0,00	0,00	1280,00	1,00	0,00
inst_20_3	1372,00	3,06	1372,00	4381,00	0,10	0,00	1372,00	1,00	0,00
inst_20_4	1488,00	2,09	1488,00	4376,00	0,10	0,00	1488,00	2,00	0,00
inst_20_5	1010,00	2,40	1010,00	3522,00	0,00	0,00	1010,00	1,00	0,00
inst_20_6	66046,00	9,37	95377,00	30961,00	0,70	44,41	81395,00	1,00	23,24
inst_20_7	56339,00	4,40	70410,00	22723,00	0,60	24,98	69970,00	2,00	24,19
inst_20_8	60378,00	4,11	82295,00	26838,00	0,50	36,30	73630,00	1,00	21,95
inst_20_9	63559,00	19,59	85903,00	28903,00	0,60	35,15	80787,00	2,00	27,11
inst_20_10	56090,00	10,71	71141,00	24773,00	0,60	26,83	68660,00	2,00	22,41
inst_20_11	66060,00	9,52	88969,50	30895,00	0,50	34,68	80020,00	1,00	21,13
inst_20_12	56339,00	4,53	71311,40	22707,00	0,50	26,58	69958,00	2,00	24,17
inst_20_13	60475,00	7,03	80110,60	26794,00	0,60	32,47	72706,00	1,00	20,22
inst_20_14	63560,00	14,88	80903,10	28861,80	0,50	27,29	80159,00	2,00	26,12
inst_20_15	56090,00	9,57	70843,40	24759,10	0,60	26,30	68550,00	2,00	22,21
inst_30_1	2010,00	9,30	2010,00	6530,00	0,30	0,00	2010,00	3,00	0,00
inst_30_2	1996,00	4,75	1996,00	7831,00	0,20	0,00	1996,00	2,00	0,00
inst_30_3	2484,00	6,02	2484,00	7797,00	0,20	0,00	2484,00	3,00	0,00
inst_30_4	2526,00	5,17	2526,00	7840,00	0,20	0,00	2526,00	3,00	0,00
inst_30_5	2430,00	8,56	2430,00	7858,00	0,20	0,00	2430,00	3,00	0,00
inst_30_6	87312,00	46,97	112257,00	37176,00	1,00	28,57	104999,00	3,00	20,26
inst_30_7	82600,00	18,20	108046,00	37176,00	0,80	30,81	101629,00	4,00	23,04
inst_30_8	82370,00	26,57	103030,00	34096,00	0,90	25,08	99820,00	4,00	21,18
inst_30_9	80877,00	37,20	104749,00	34081,00	0,80	29,52	98719,00	4,00	22,06
inst_30_10	80030,00	16,74	101120,00	34089,00	0,50	26,35	99488,00	4,00	24,31
inst_30_11	87312,00	46,59	113369,10	37139,00	0,90	29,84	104786,00	4,00	20,01
inst_30_12	82600,00	17,00	108534,30	37159,00	0,70	31,40	101828,00	3,00	23,28
inst_30_13	82370,00	27,26	104274,70	34082,30	0,70	26,59	100212,00	3,00	21,66
inst_30_14	80877,00	34,68	102921,60	34065,00	0,90	27,26	98556,00	4,00	21,86
inst_30_15	80030,00	22.33	102093,20	34074,00	0,80	27,57	98398,00	4,00	22,95
inst_50_1	2839,00	45,23	2839,00	8807,00	0,40	0,00	2839,00	12,00	0,00
inst_50_2	3674,00	42,73	3674,00	13091,00	0,30	0,00	3674,00	9,00	0,00
inst_50_3	3580,00	38,18	3580,00	13021,00	0,50	0,00	3580,00	12,00	0,00
inst_50_4	2464,00	44,61	2464,00	8807,00	0,30	0,00	2464,00	12,00	0,00
inst_50_5	3200,00	51,61	3200,00	10917,00	0,40	0,00	3200,00	12,00	0,00
inst_50_6	136961,00	84,30	177074,00	56808,00	1,60	29,29	170238,00	12,00	24,30
inst_50_7	130875,00	48,42	160020,00	56808,00	1,90	22,27	158361,00	11,00	21,00
inst_50_8	155719,00	253,32	223552,00	77377,00	2,40	43,56	199304,00	11,00	27,99
inst_50_9	160044,00	255,28	225977,00	77365,00	2,20	41,20	200063,00	12,00	25,00
inst_50_10	143799,00	128,35	188945,00	67080,00	2,40	31,40	181674,00	11,00	26,34
inst_50_11	136961,00	80,47	177459,40	56775,00	1,70	29,57	168987,00	12,00	23,38
inst_50_12	130875,00	47,09	160332,00	56795,10	1,60	22,51	157591,00	12,00	20,41
inst_50_13	155748,00	189,89	221007,10	77235,40	2,30	41,90	192748,00	11,00	23,76
inst_50_14	160044,00	190,00	229452,50	77198,80	2,60	43,37	195287,00	12,00	22,02
inst_50_15	143799,00	86,80	187936,30	67019,00	2,00	30,69	180073,00	13,00	25,23
Média		139,28			0,68	19,47		4,20	15,04

construtiva gulosa. Além disso, a formulação matemática conhece todas as requisições da instâncias, o que não é suposição realista na prática. Já o algoritmo distribuído, considera apenas as requisições originadas no período atual e possíveis não atendidas de períodos anteriores.

6 Conclusão

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 uma heurística distribuída. As simulações realizadas mostraram que a heurística distribuída obteve resultados satisfatórios, com soluções em média 19.5% do ótimo, e no pior caso a 44.4% do ótimo e que pode ser uma boa opção para ser adotada em RDC's reais. Como trabalhos futuros, tem-se o objetivo de solucionar uma versão do PPR mais realista, onde os enlaces da rede mudam com o tempo e os conteúdos possuam tempo de vida. Para isto, a heurística distribuída proposta será alterada, e novas serão propostas para obtenção de soluções mais próximas do ótimo.

Referências

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

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

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

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

Frank, C. e Römer, K. Distributed facility location algorithms for flexible configuration of wireless sensor networks. *DCOSS'07: Proceedings of the 3rd IEEE international conference on Distributed computing in sensor systems*, p. 124–141, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-73089-7, 2007.

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

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

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

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

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

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