

Avaliação de Desempenho

Trabalho Prático 3 — Relatório

Resumo da atividade

A atividade deste trabalho consistiu em responder à Questão 5, proposta aleatoriamente via ColabWeb, que trata da implementação do algoritmo *dLife* no ambiente de simulação The ONE. A implementação foi feita utilizando os arquivos e explicações providas pelos autores do artigo *Opportunistic Routing Based on Daily Routines* [Moreira *et al.* 2014], no qual apresentam o algoritmo estudado neste trabalho e sua comparação a outros algoritmos de roteamento em redes oportunistas.

O algoritmo dLife

O algoritmo estudado e implementado nesta atividade é baseado nas interações sociais entre os usuários de uma rede oportunista ao longo da rotina de um dia comum. Diferente dos demais algoritmos de roteamento, ele leva em consideração que cada usuário/nó, dependendo das suas relações com os demais, passa mais ou menos tempo em contato com certos nós da rede ao longo de um dia. Essa informação permite um roteamento mais eficiente, uma vez que é possível utilizar essa rotina conhecida e as interações entre usuários para aumentar a probabilidade de entrega e diminuir a latência.

A implementação em si é feita a partir de um grafo no qual cada contato (aresta) tem um peso determinado pelo tempo em que dois nós da rede ficam próximos durante um determinado período do dia. O roteamento é feito com duas funções auxiliares: a função *TECD*, na qual mensagens são passadas com prioridade a quem tiver maior relação com o nó destinatário. Essa função calcula a média de tempo de contato de cada nó com os demais durante um período determinado de um dia, ao longo de vários dias; e a função *TECDi*, na qual nós que não possuem relações definidas com o destinatário repassam a mensagem prioritariamente para aqueles que têm maior importância na rede (maior probabilidade de entrega). Essa importância de um determinado nó é calculada com base nos pesos que este nó específico tem com os seus vizinhos em um determinado período do dia (*TECD*) e também a importância de cada um desses nós vizinhos na rede.

Tutorial de uso

Para adicionar o algoritmo *dLife* ao simulador The ONE, alguns passos importantes devem ser seguidos:

1. Baixar o The ONE versão 1.4.1, que pode ser encontrado em <https://www.netlab.tkk.fi/tutkimus/dtn/theone/>. O algoritmo **NÃO** funciona com a versão mais recente devido a mudanças drásticas no código do simulador (ver considerações abaixo).
2. Seguir as instruções providas junto com o algoritmo pelos seus autores, que consistem em basicamente alterar o código base do The ONE antes de compila-lo para suportar as funções auxiliares TECD e TECDi, bem como os próprios algoritmos do artigo e seus cálculos:

- a. No arquivo **core.DTNSim** alterar o método main para determinar os períodos de tempo da função TECD:

```
public static void main(String[] args) {  
    // TECD *****  
    ArrayList<Long> arl= new ArrayList<Long>();  
    /** 24 slots of 1 hour */  
    arl.add((long)3600); // 1  
    arl.add((long)7200); // 2  
    arl.add((long)10800); // 3  
    arl.add((long)14400); // 4  
    arl.add((long)18000); // 5  
    arl.add((long)21600); // 6  
    arl.add((long)25200); // 7  
    arl.add((long)28800); // 8  
    arl.add((long)32400); // 9  
    arl.add((long)36000); // 10  
    arl.add((long)39600); // 11  
    arl.add((long)43200); // 12  
    arl.add((long)46800); // 13  
    arl.add((long)50400); // 14  
    arl.add((long)54000); // 15  
    arl.add((long)57600); // 16  
    arl.add((long)61200); // 17  
    arl.add((long)64800); // 18  
    arl.add((long)68400); // 19  
    arl.add((long)72000); // 20  
    arl.add((long)75600); // 21  
    arl.add((long)79200); // 22  
    arl.add((long)82800); // 23  
    arl.add((long)86400); // 24  
    SlotTimeCheck g = new SlotTimeCheck(arl);  
    // TECD *****  
}
```

- ```
}
```
- b. No arquivo **core.SimClock** alterar os seguintes métodos para que suportem os slots de tempo necessários no TECD:

```
public void setTime(double time) {
 clockTime = time;
 SlotTimeCheck.update(time); }
```

```
public static void reset() {
 clockTime = 0;
 SlotTimeCheck.currentday=1;}
```

- c. Adicionar a classe **SlotTimeCheck** à pasta/pacote **core** do simulador;
- d. Adicionar **DecisionEngineRouter**, **RoutingDecisionEngine** e **MessageRouter** à pasta/pacote **routing**;
- e. Adicionar **Dlife**, **DlifeComm**, **Duration**, **CommunityDetectionEngine**, **CommunityDetection**, **KCliqueCommunityDetection**, **SimpleCommunityDetection**, **Centrality**, **CWindowCentrality**, **SWindowCentrality** à pasta/pacote **routing.community**. Nem todos esses arquivos serão necessários para utilizar o algoritmo, mas servem para que o simulador compile sem problemas;
- f. Adicionar **CommunityDetectionReport** à pasta/pacote **report**;
3. Compilar todos os arquivos do simulador;

Seguidos estes passos, basta criar um arquivo de configuração que utilize o algoritmo dLife para roteamento (provido nos arquivos do trabalho como **dlife.txt**) e então executar a simulação com **one.bat** ou **one.sh** com esse arquivo de configuração como parâmetro na linha de comando.

## Considerações extras

A implementação no The ONE mais recente (versão 1.6) foi impossibilitada por mudanças no código-base do simulador, uma vez que a implementação original do algoritmo dLife provida pelos autores do artigo foi feita levando em consideração o código-base da versão 1.4.1. Algumas das mudanças que impossibilitaram a implementação na versão mais recente foram mudanças na estrutura de pacotes, mudanças na declaração e funcionamento interno de métodos de classes auxiliares e, principalmente, mudanças no funcionamento da classe abstrata **MessageRouter**, que é utilizada como base para todos os algoritmos de roteamento do simulador.

O algoritmo dLife não só altera esta classe diretamente para funcionar, como também utiliza convenções e variáveis estáticas (que controlam a simulação como um todo) que não existem mais na versão mais recente; desta forma, o simulador na sua versão mais recente teria que ser alterado por completo para que funcionasse com o algoritmo estudado e isso estaria completamente fora do escopo do trabalho proposto.