MELHORANDO O ENCAMINHAMENTO DE MENSAGENS EM REDES D2D COM MÚLTIPLOS SALTOS

MICHAEL DOUGRAS DA SILVA

MELHORANDO O ENCAMINHAMENTO DE MENSAGENS EM REDES D2D COM MÚLTIPLOS SALTOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Antônio Alfredo Ferreira Loureiro

Belo Horizonte - Minas Gerais

Junho de 2018

MICHAEL DOUGRAS DA SILVA

IMPROVING D2D MULTI-HOP MESSAGE FORWARDING

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Antônio Alfredo Ferreira Loureiro

Belo Horizonte - Minas Gerais
June 2018

da Silva, Michael Dougras

Improving D2D Multi-Hop Message Forwarding / Michael Dougras da Silva. — Belo Horizonte - Minas Gerais, 2018
xxi, 11 f. ; 29cm

Dissertação (mestrado) — Federal University of Minas Gerais
Orientador: Antônio Alfredo Ferreira Loureiro

1. d2d. 2. multi-hop. 3. network. 4. forwarding. I. Título.

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* é uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

Dedication text goes here.

Acknowledgments

Here goes aknowledgement text. texto muito importante

"Not everything that counts can be counted, and not everything that can be counted counts."

(William Bruce Cameron - "Informal Sociology: A Casual Introduction to Sociological Thinking")

Abstract

Abstract text goes here.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Chapter 2

Message Forwarding in D2D Networks

In this chapter we present an overview of message forwarding algorithms for D2D networks. The first section presents the main contributions to the topic, discussing the state-of-art forwarding algorithms. Then we introduce an implementation proposal of Groups NET algorithm de Oliveira Nunes et al. 2016 in a distributed environment. The implementation detail is explained in two parts: the first covers the group detection algorithm and the second the forwarding implementation itself.

2.1 Introduction

Introduce the challenge of forwarding messages in this scenario of dynamic nodes movement.

The main solutions were initially proposed in the context of DTNs (Delay Tolerant Networks) or are derived from them, since the two types of networks share some intrinsic characteristics.

Add a brief description about message forwarding in D2D.

Add description for the following algorithms:

* Epidemic routing * Prophet * Bubble Rap * Original Groups NET

2.2 The Distributed Groups NET

In this section we propose a distributed implementation of Groups NET forwarding algorithm. The solution is comprised of two stages: The first stage is group detection, in which nodes use neighborhood discovery to infer social group meeting using a local algorithm. The second stage is message forwarding, in which nodes use the generated group data to decide when to forward a message or not. The following sections describe in details the two stages.

2.2.1 Mobile Group Detection

The mobile group detection algorithm proposed here is derived from the ideas presented in **?**. A group is defined as a set of people that get in contact on a given point of space and time. The authors show that groups have regular encounters along the time. This property can be used by forwarding algorithms.

In the original proposal the authors use a approach to detect groups based on global network knowledge. They basically build a contact graph considering a pre-defined time interval, and for each graph is used the Clique Percolation community detection algorithm. Each community detected by the algorithm is considered a group meeting in that time window. Then a metric called *Group Correlation Coefficient* is used to detect instances of the same group across multiple time windows.

With this approach the authors show that groups usually have regular encounters along the time (specially in daily and weekly basis). But they also discuss that the detection approach is unfeasible for a distributed environment, because it assumes a global network knowledge. However, authors also discuss that implementing a distributed solution for this should be a simple task, because in the local scope nodes can use neighborhood discover strategies and process regular neighbors encounters to decide their group encounters.

In this work we expanded this idea to build a distributed solution for group detection.It is composed of four processes that are executed concurrently by each node using basically neighborhood inspection and some basic additional parameters. In the following sections each process is explained.

2.2.1.1 Device's Local Group Detection

This process is responsible for processing nodes' local contacts and decide when a group meeting happens. The algorithm is very simple. Each node keeps two lists of devices. The first is the *friends* list (lets call it FL) that keeps track of current nodes considered as friends, i.e, members of a group that have a recent encounter. The second list is the *strangers* list (lets call it SL), that keeps track of recent nodes' contacts that are not yet considered as friends. Each entry in these lists have a contact counter that keeps track of consecutive contacts with that node, and also have a inactive counter that keeps track of consecutive periods of time without contact with that node.

The node inspects its neighborhood at each predefined *time interval* and based on current neighbors it updates the friends and strangers list. This update is based on two predefined parameters: the *friend threshold* and *inactive threshold*. Lets call these parameters as FT and IT respectively. Based on these parameters the following steps are executed:

- At each time interval a node collects its current neighbors (lets call it CN).
- The first step is to update the counters of friends list. For each current neighbor that is present in the friends list, we increase the contact counter and reset the inactive counter. For each node present in the friends list that is not a current neighbor we reset the contact counter and increase the inactive counter.
- The second step is to update the strangers list. For each current neighbor that is present in the strangers list we reset the inactive counter and increase the contact counter. If the contact counter reaches the friend threshold the node is promoted to friends list. For each node that is present in strangers list but is not a current neighbor we reset the contact counter and increase the inactive counter. If the inactive counter reaches the inactive threshold the node is removed from the strangers list. Nodes that are current neighbors and are not present in both friends and strangers list are added in the strangers list with contact counter equals 1.
- As a final step, we check the number of nodes in the friends list that are considered inactive. A node is considered inactive when the inactive counter reaches the inactive threshold. If more than 50 percent of the nodes in the friends list is inactive, the friend list is archived and considered as a group meeting that happened in the past. When this happens we clear the friends and strangers list and restart the algorithm.

In the end this algorithm creates a list of local detected group meetings. This list is used by the local group combination process to detect instances of the same group with multiple encounters along the time.

2.2.2 Local Group Combination

In this step each mobile device process groups discovered in the previous step to discover groups that have multiple encounters along the time. The algorithm is based on a metric called *Group Correlation Coefficient* defined in **?**. This metric basically computes the proportion of nodes shared between two sets.

Each node keeps a list of *combined groups*, that keeps track of groups with multiple encounters along the time. For each local group detected in the previous process, we check if there is a combined group with Group Correlation Coefficient greater or equals 0.5. If there is, the local group is combined into the combined group adding its members to it and registering a new encounter. If there isn't, a new combined group is initialized with the local group data.

The combined groups generated in this process are used by neighborhood inspection to compose the final groups considered in the forwarding algorithm.

2.2.3 Neighborhood Inspection

In this step each node inspects its

Chapter 3

Buffer Management Strategies

Chapter 4

Conclusion

Put some text here

Bibliography

de Oliveira Nunes, I., de Melo, P. O. V., and Loureiro, A. A. F. (2016). Groups-net: Roteamento ciente de encontros de grupos em redes móveis d2d. In *Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems*, Salvador, Bahia.