UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO GONÇALVES FILHO

VDTCON - UMA ARQUITETURA PARA REDES VEICULARES BASEADA EM DTN E NDN

FORTALEZA – CEARÁ

2016

JOÃO GONÇALVES FILHO

VDTCON - UMA ARQUITETURA PARA REDES VEICULARES BASEADA EM DTN E NDN

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Joaquim Celestino Júnior

FORTALEZA – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Sistema de Bibliotecas

Gonçalves, João Gonçalves Filho.
VDTCON - Uma Arquitetura para Redes Veiculares Baseada em DTN e NDN [recurso eletrônico] / João Gonçalves Filho Gonçalves. - 2016.
1 CD-ROM: il.; 4 ¾ pol.

CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico com 73 folhas, acondicionado em caixa de DVD Slim (19 x 14 cm x 7 mm).

Dissertação (mestrado acadêmico) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Mestrado Acadêmico em Ciência da Computação, Fortaleza, 2016.
Área de concentração: Ciência da Computação.
Orientação: Prof. Ph.D. Joaquim Celestino Júnior.

1. VANET. 2. DTN. 3. NDN. 4. VDTCON. I. Título.

ATA DA OCTOGÉSIMA QUARTA DEFESA PÚBLICA DE DISSERTAÇÃO DE MESTRADO

Ao vigésimo oitavo dia do mês de março de dois mil e dezesseis, no miniauditório do prédio de Pesquisa e Pós-Graduação em Computação, do Mestrado Acadêmico em Ciência da Computação – MACC, realizou-se a sessão pública de defesa da dissertação de **JOÃO GONÇALVES FILHO**, aluno regularmente matriculado no Mestrado Acadêmico em Ciência da Computação–MACC, intitulada: **" *VDTCON* - Uma Arquitetura para Redes Veiculares Baseada em DTN E NDN"**, A Banca Examinadora reuniu-se no horário de 15h às 17h00 horas, sendo constituída pelos Professores Doutores **Joaquim Celestino Júnior (Orientador/UECE); Marcial Porto Fernandez (UECE); Atslands Rego da Rocha (UFC).** Inicialmente o mestrando expôs seu trabalho e a seguir foi submetido à arguição pelos membros da Banca, dispondo cada membro de tempo para tal. Finalmente a Banca reuniu-se em separado e concluiu por considerar o mestrando APROVADO, por sua dissertação e sua defesa pública. Eu, **Professor Dr. Joaquim Celestino Júnior**, Orientador e Presidente da Banca, lavrei a presente Ata que será assinada por mim e demais membros da Banca. Fortaleza, 28 de março de 2016.

Prof. Dr. Joaquim Celestino Júnior
Orientador – UECE

Prof. Dr. Marcial Porto Fernandez (UECE);

Profª. Drª. Atslands Rego da Rocha (UFC).

AGRADECIMENTOS

Primeiramente a Deus que na sua infinita graça meu concedeu a oportunidade de estar concluindo mais esta etapa da minha vida.

Aos meus pais João e Eliene, pelo amor e todo cuidado que tiveram comigo.

A minha esposa Vanice que passou comigo em cada etapa desse mestrado.

Aos amigos do Larces. Em especial o Daniel Sucupira e o Henrique Brandão que caminharam comigo ao longo desses anos.

Ao professor Celestino, não somente pela orientação, mas também pela amizade e conselhos para vida.

Agradeço a todos os demais que de alguma forma contribuiram para minha formação.

“Para tudo há uma ocasião certa; há um tempo
certo para cada propósito debaixo do céu:”

(Eclesiastes 3:1)

RESUMO

Diversos tipos de redes vêm sendo propostas para solucionar problemas de comunicação. Dentre elas, as *Delay Tolerant Networks* (DTNs) que utilizam o paradigma armazena-carrega-e-encaminha para lidar com ambientes de desconexões constantes nos quais há dificuldades na construção de um caminho fim-a-fim entre os nós comunicantes, tal como pode acontecer nas *Vehicular Ad hoc Networks* (VANETs). Um outro tipo de rede é a *Named Data Networking* (NDN), cuja a proposta é mudar o tipo de endereçamento tradicional para um com base em nomes. O presente trabalho traz a proposta da arquitetura chamada *Vehicular Delay Tolerant Content-Oriented Networks* (VDTCON) que utiliza o paradigma de armazena-carrega-e-encaminha de DTN e, da mesma forma que na proposta NDN, faz o endereçamento com nomeação hierárquica. O próposito da VDTCON é fornecer uma rede orientada a conteúdo com o paradigma utilizado em DTN para aplicações de VANETs. A arquitetura VDTCON é composta por diversos módulos. Dentre eles, um importante a ser ressaltado é o módulo de roteamento, onde pode ser inserido qualquer protocolo de roteamento DTN, bastando adaptá-lo para a arquitetura. Neste trabalho, os protocolos de roteamento DTN Epidêmico e *Spray-And-Wait* foram adaptados e inseridos no módulo de roteamento. As adaptações foram nomeadas VDTCON-Epidemic e VDTCON-Spray. Além disso, foi proposto um protocolo de roteamento chamado VDTCON-QLR que também foi inserido no módulo de roteamento. Semelhantemente, os protocolos de roteamento DTN orientado a conteúdo STIR e SIR foram implementados e colocados também no módulo de roteamento. A arquitetura VDTCON com os protocolos de roteamento foram implementados no *Network Simulator 3* (NS3) para avaliação de desempenho.

Palavras-chave: VANET. DTN. NDN. VDTCON.

ABSTRACT

Different types of networks have been proposed to deal with communication problems. Among them, there are Delay Tolerant Networks (DTNs) that using the stores-carry-and-forward paradigm to deal which environments that show regular disconnections, causing difficulties in forming an end-to-end path in communicating nodes, such as what may happen in Vehicular Ad Hoc Networks (VANETs). Another type of network is the Named Data Networking (NDN), whose proposal is to change the traditional form of addressing for one based on names. This work presents the proposed architecture called Vehicular Delay Tolerant Content-Oriented Networks (VDTCON) that using the paradigm of stores carries-and-switch NTD and in the same way that the NDN proposal is addressing with hierarchical naming. The purpose of VDTCON is to provide content-oriented network with the paradigm used in DTN for VANETs applications. The VDTCON architecture is composed of several modules. Among them, an important to be highlighted is the routing module, which can be inserted into any DTN routing protocol, simply adapt it to architecture. In this work, the DTN routing protocols Epidemic and Spray-And-Wait have been adapted and inserted to the routing module. The Adaptations were named VDTCON-Epidemic and VDTCON-Spray. Further, a routing protocol called VDTCON- QLR was proposed which was also inserted into the routing module. Similarly, the DTN content-oriented routing protocols STIR and SIR were implemented and placed also in the routing module. The VDTCON architecture with the routing protocols were implemented in Network Simulator 3 (NS3) for performance evaluation.

Keywords: VANET. DTN. NDN. VDTCON.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ADU	Application Data Units
CCDTN	Content-Centric Delay-Tolerant Networks
CCH	Control Channel
CCN	Content-Centric Networking
CON	Content-Oriented Networks
CS	Content Store
D-Packet	Pacote de Dado
DSRC	Dedicated Short Range Communications
DT-ICAN	Disruption-Tolerant Information-Centric Ad-Hoc Network
DTN	Delay Tolerant Network
DTNRG	Delay Tolerant Network Research Group
DTRB	Delay Tolerant Reinforcement-Based Routing
FCC	Federal Communications Commission
FIB	Forwading Information Base
I-Packet	Pacote de Interesse
LLC	Logical Link Control
MANET	Mobile Ad hoc Network
MIB	Management Information Base
NDN	Named Data Networking
OBU	On Board Unit
PDU	Protocol Data Units
PIT	Pending Interest Table
RCP	Resource Command Processor
RM	Resource Manager
RMA	Resource Manager Application
RSU	Road Side Unit
RTSB	Request-To-Send-Block
SCH	Service Channel
SIR	Swarm-based Intelligent Routing
STIR	STIgmergy Routing
SUMO	Simulation of Urban MObility

ToD	Trend Of Delivery
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad hoc Network
VCCN	Vehicular Content Centric Network
VDTCON	Vehicular Delay Tolerant Content-Oriented Networks
WAVE	Wireless Acess in the Vehicular Environment

SUMÁRIO

1 INTRODUÇÃO

Diversos tipos de arquiteturas de redes vêm surgindo para enfrentar variados tipos de problemas em diferentes cenários. Para redes que sofrem com problemas de constantes desconexões, tal como pode acontecer em uma *Vehicular Ad hoc Network* (VANET), foi proposta a arquitetura *Delay Tolerant Network* (DTN) que propõe lidar com esses problemas. Ambientes DTN possuem as características das chamadas redes desafiadoras (FALL, 2003), dentre as quais podemos destacar: redes móveis terrestres, redes com meio de transmissão exótico (tais como comunicação com satélites próximos da terra, raios de longas distâncias e links óticos), redes ad hoc militares e redes sensores. Nesses cenários, os protocolos de roteamento convencionais não são adequados devido a dificuldade de se manter um caminho fim-a-fim entre dois nós. Assim, em DTN, é utilizado o paradigma chamado de armazena-carrega-e-encaminha que permite a um pacote ser carregado por um nó durante um longo tempo.

Em (JACOBSON et al., 2009) é proposta um tipo de rede orientada a conteúdo, a *Content-Centric Networking* (CCN) que hoje é conhecida como projeto *Named Data Networking* (NDN). Nessa rede, o autor propõe mudar o modelo de comunicação da internet trocando a preocupação com endereço (*onde*) para preocupação com conteúdo (*o que*). Nesse novo modelo, o autor sugere lidar com questões de disponibilidade, segurança e dependência local. O modelo atual sofre com essas questões por que o usuário está preocupado com o *o que* ele quer, mas o modelo ainda o mantém em termos de *onde*. As NDNs podem tirar o máximo de proveito das múltiplas conexões simultâneas (JACOBSON et al., 2009), assim, elas podem funcionar bem com redes ethernet e também redes sem fio. Dessa forma, alguns trabalhos vem propondo a utilização de NDN em *Mobile Ad hoc Network* (MANET) e VANET (OH et al., 2010) (AMADEO et al., 2012) (GRASSI et al., 2014).

Fornecer um bom desempenho para aplicações de VANETs é sempre desafiador devido a suas características de mobilidade que dificultam a comunicação dos nós da rede. Neste trabalho, é proposta uma arquitetura que utiliza o paradigma de armazena-carrega-e-encaminha de DTN juntamente com algumas características da proposta de NDN visando fornecer uma rede tolerante a atrasos orientada a conteúdo para aplicações de VANET. Essa arquitetura é nomeada de *Vehicular Delay Tolerant Content-Oriented Networks* (VDTCON). A arquitetura possui diversos módulos, dentre eles o módulo de roteamento, onde é inserido o protocolo de roteamento. Neste trabalho, o protocolo Epidêmico (VAHDAT et al., 2000) e o *Spray-And-Wait* (SPYROPOULOS et al., 2005) foram implementados e adaptados para a arquitetura proposta.

Além disso, foi proposto o protocolo de roteamento VDTCON-QLR. Os protocolos STIR (NGUYEN et al., 2010) e SIR (NGUYEN et al., 2011) que são protocolos de roteamento DTN orientados a conteúdo, foram implementados. Os cinco protocolos foram colocados no módulo de roteamento do VDTCON e comparados em um cenário VANET.

1.1 MOTIVAÇÃO

Em VANETs, em geral, as aplicações são divididas em dois tipos: aplicações de conforto que são apenas para entretenimento ou facilidades para o motorista e aplicações de segurança que fornecem mecanismos a fim de ajudar o motorista a evitar acidentes cooperando para preservação da vida no trânsito (YOUSEFI et al., 2006). Podemos ter aplicações onde veículos estão interessados em algum tipo de conteúdo de um determinado produtor fixo na estrada. Esse produtor pode ser, por exemplo, uma loja que fornece os preços de seus produtos aos veículos solicitantes. Outros exemplos de conteúdos podem ser: propagandas, informações atualizadas da estrada, dentre outros.

Tendo em vista essas aplicações, é necessária uma arquitetura que considere os problemas das VANETs. Utilizando DTN é possível aproveitar a característica de armazena-carrega-e-encaminha para lidar com as constantes quebras de links das VANETs e problemas de conectividade de cenários esparsos que dificultam a existência de um caminho fim-a-fim. Utilizando também a abordagem de nomeação hierárquica de NDN, retiramos a necessidade do endereçamento IP em VANETs, facilitando a comunicação das aplicações VANETs.

1.2 OBJETIVOS

Propor uma arquitetura para fornecer uma rede tolerante a atrasos orientada a conteúdo para aplicações de VANETs. Possibilitar a inserção de qualquer protocolo de roteamento DTN na arquitetura.

1.3 METODOLOGIA

As simulações foram feitas utilizando o NS3 (versão 3.16) (NSNAM, 2015). Os protocolos e as simulações foram implementados utilizando a linguagem de programação C++. Os modelos de propagação utilizados foram: Nakagami (SARKAR et al., 2003) e Friis (KAI, 2000), com alcance de transmissão de aproximadamente 300 metros. De acordo com (TALIWAL

et al., 2004) e (SCHMIDT-EISENLOHR et al., 2007), o melhor modelo de propagação para VANET é o Nakagami.

O módulo do NS3 utilizado em (VIEIRA et al., 2013) e (FILHO et al., 2014) foi adaptado para a arquitetura VDTCON para que pudesse ser simulada. Essa adaptação mantém várias estruturas utilizadas no módulo original. Dessa forma, a implementação da camada DTCON ficou dentro da *Bundle Layer* do módulo original.

Para cada experimento, foi criado um arquivo de simulação. Neles, a mobilidade dos veículos foi feita por meio do gerador de mobilidades *Simulation of Urban MObility* (SUMO) (KRAJZEWICZ et al., 2002).

Foram implementados cinco protocolos de roteamento para serem colocados no módulo de roteamento da arquitetura VDTCON, sendo eles: VDTCON-Epidemic (adaptação do Epidêmico), VDTCON-Spray (adaptação do *Spray-And-Wait*), VDTCON-QLR (desenvolvido neste trabalho), STIR (NGUYEN et al., 2010) e SIR (NGUYEN et al., 2011). Cada um deles foi avaliado utilizando as métricas: taxa de interesses satisfeitos, *overhead* e atraso médio. Todos os protocolos foram comparados em um cenário VANET.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 NAMED DATA NETWORKING - NDN

As NDNs (Conhecidas anteriormente como CCN) (JACOBSON et al., 2009) é um recente tipo de *Content-Oriented Networks* (CON). A proposta das CONs é trocar o modelo de comunicação que se preocupa com *onde* localizar o conteúdo por uma que se preocupe em *qual* conteúdo é requerido.

Recentemente, observou-se que a maior parte do tráfego da Internet é causado pelas aplicações populares, tais como: Netflix, Facebook, Youtube, dentre outras (PERINO; VARVELLO, 2011). Podemos, então, perceber que a demanda está preocupada em obter o conteúdo não importando onde ele esteja. Esse modelo é adequado aos CONs. Já foram propostos vários tipos de CONs diferentes da NDN, sendo alguns deles: CBN (Pioneira) (CARZANIGA et al., 2000), DONA (KOPONEN et al., 2007) e ROFL (CAESAR et al., 2006).

As características principais das NDNs são (PERINO; VARVELLO, 2011) (JACOBSON et al., 2009):

- Não precisar do roteamento IP.
- Cada conteúdo possui um identificador (URI) ao invés de um *host*.
- Para a realização das decisões de roteamento, são considerados os identificadores dos conteúdos.
- Os conteúdos podem ficar em *cache* nos nós intermediários.
- Existe um suporte nativo para *multicast*.
- Além disso, o modelo de nomeação utilizado é o hierárquico.

2.1.1 Nomeação de Conteúdo

No modelo atual da Internet, para se obter um conteúdo é necessário que um consumidor tenha conhecimento do endereço IP de quem possui esse conteúdo. No modelo da NDN, é trocada a abordagem tradicional da arquitetura centrada em estações pela abordagem de obtenção do conteúdo exclusivamente pelo nome. A forma como esse nome se estrutura deve ser organizada por um esquema de nomeação de conteúdo que, de maneira ideal, deve apresentar bem as seguintes características (BRITO et al., 2012):

- **Unicidade:** Garantir que cada conteúdo possua sempre um nome único na rede.
- **Persistência:** Possibilidade de validar o nome do conteúdo em sincronismo com a validade

do próprio conteúdo.

- **Escalabilidade:** Possibilidade de aplicação em diversos cenários.

Nas CONs, são utilizados três esquemas de nomeação: nomeação plana, nomeação por atributos e nomeação hierárquica (BRITO et al., 2012). Neste trabalho, foi feita uma descrição apenas da nomeação hierárquica, pois ela é utilizada nas NDNs.

Na nomeação hierárquica, os nomes únicos dos conteúdos são montados através da concatenação dos componentes hierárquicos do conteúdo. Dessa forma, o nome do conteúdo possui uma forte característica semântica, uma vez que seu nome é composto por características descritivas desse conteúdo: nome do publicador, versão, formato etc.

Um consumidor que queira um conteúdo precisará, de forma determinística, construir o nome do dado, mesmo sem ter nenhum conhecimento proveniente da rede acerca dele. Todavia com o modelo hierárquico, é possível ao usuário montar o nome do conteúdo desejado. Exemplo: se um usuário precisar solicitar o vídeo *aula.avi* que é produzido pela *UECE*, ele sabe que o nome do conteúdo deve estar na forma ***UECE/videos/aula.avi***, onde '/' é o delimitador de cada componente na representação do texto, similar às URLs.

O usuário pode ser mais específico no pedido dizendo qual versão do vídeo ele quer. Além disso, a recuperação do conteúdo se dá por *pedaços (chunks)*. Dessa forma, no caso dele conhecer a estrutura completa do nome, o usuário solicitará **UECE/videos/aula.avi/versao1/pedaço1** e, logo em sequência, os pedaços seguintes. O usuário não necessariamente precisa saber o nome completo do conteúdo. Se o usuário, por exemplo, não especificar a versão ou o pedaço, o produtor pode inferir que ele está solicitando o primeiro pedaço da última versão do vídeo **UECE/videos/aula.avi/versao_ultima/pedaço1**. Com as informações do primeiro pedaço, o usuário pode inferir os pedaços subsequentes.

Os autores da NDN quiseram uma arquitetura que preservasse algumas decisões de projeto do TCP/IP, ao buscar mantê-la simples, robusta e escalável (JACOBSON et al., 2009). A questão da nomeação hierárquica também acontece por esse motivo, pois nela é utilizada a agregação dos nomes com uso de mapeamento de prefixos mais longos, semelhante ao que é utilizado no roteamento IP.

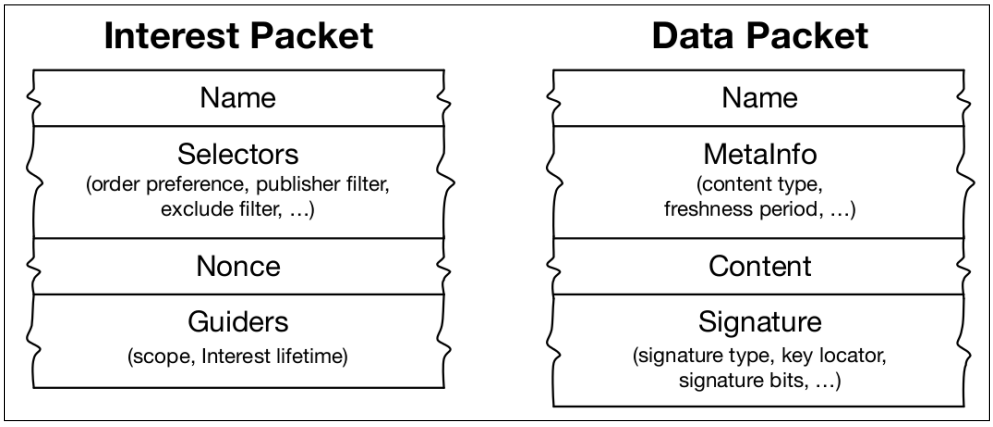
Um problema que esse tipo de nomeação enfrenta é a questão da persistência. Uma vez que o nome é associado a semântica do conteúdo, qualquer mudança hierárquica do conteúdo como transferência de propriedade deve ser refletida no seu nome (BRITO et al., 2012).

2.1.2 Arquitetura NDN

A aplicação típica das NDNs possui a figura do produtor e do consumidor de conteúdo. A comunicação ocorre por meio da troca de dois tipos de pacotes: Pacote de Interesse (*I-Packet*) e Pacote de Dado (*D-Packet*). Quando um consumidor quer solicitar um dado, coloca o nome do conteúdo no *I-Packet*, então o envia pela rede. Os roteadores usam esse nome para encaminhar o interesse até o produtor do dado. Uma vez que o *I-Packet* alcançou o produtor do dado, então ele gera um *D-Packet* e o manda para o consumidor. Para chegar até o consumidor, o *D-Packet* segue um caminho reverso do que foi montado na viagem do *I-Packet*.

As estruturas dos pacotes são mostradas na Figura 1. No site do projeto da NDN, (PROJETO-NDN, 2015) é dada um descrição completa dos campos dos pacotes.

Figura 1 – Pacotes da arquitetura NDN.



Fonte: (ZHANG et al., 2014)

Para realizar o encaminhamento dos *I-Packets* e *D-Packets*, cada roteador NDN mantém três estruturas de dados básicas: *Pending Interest Table* (PIT), *Forwading Information Base* (FIB) e o *Content Store* (CS).

A PIT registra quais *I-Packets* ainda não foram satisfeitos no roteador. Cada entrada da PIT associa qual interface fez a solicitação de cada *I-Packet*. Dessa forma, é possível montar o que é chamado caminho da "migalha de pão" para cada *D-Packet* correspondente fazer o caminho de volta. A cada roteador que o *D-Packet* passa, é consumido (apaga a entrada I-Packet na PIT) o *I-Packet* correspondente no roteador e então o *D-Packet* é reencaminhado pela interface associada, assim podendo fazer o caminho de volta para o consumidor. Como podemos ver, cada *I-Packet* recupera apenas um *D-Packet*. Essa regra básica garante que a balança de fluxo seja mantida na rede e permite uma eficiente comunicação entre variados tipos de máquinas sobre

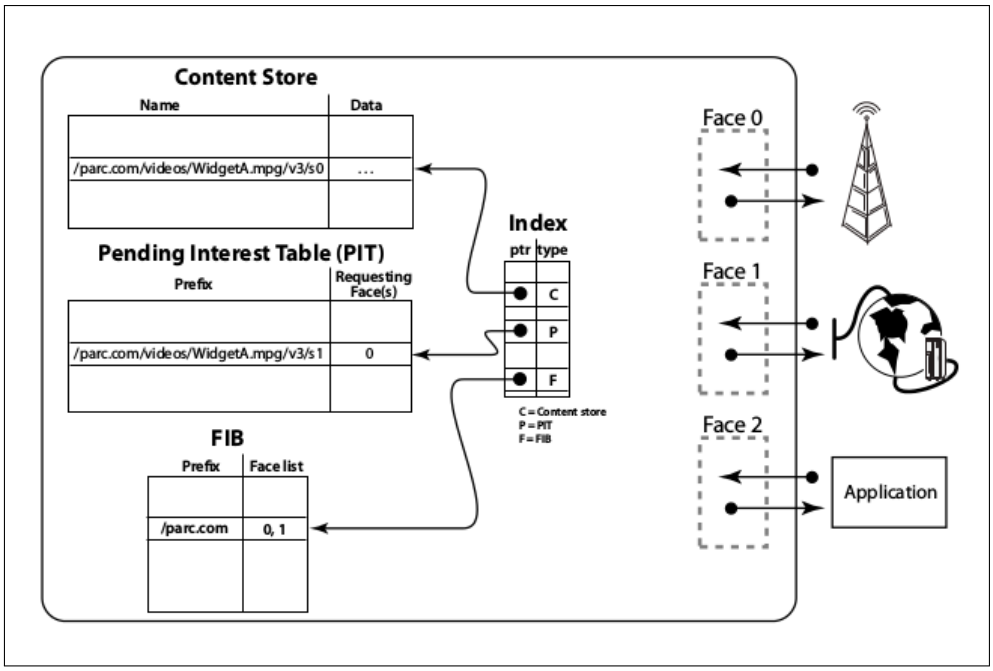
redes com diferentes velocidades (JACOBSON et al., 2009).

A FIB funciona de maneira semelhante a tabela de rotas do roteamento IP. Cada entrada da FIB indica por qual interface um determinado *I-Packet* deve ser encaminhado.

O CS armazena em *cache* os *D-Packets* que o roteador encaminhou. Dessa forma, se o mesmo *I-Packet* chegar novamente nesse roteador, não será necessário reencaminhá-lo, o roteador irá simplesmente devolver o *D-Packet* que está em *cache*.

A Figura 2 mostra o modelo das três estruturas mencionadas. O termo *face* é utilizado no lugar de interface porque não se trata necessariamente de hardware. A *face* pode ser também uma aplicação.

Figura 2 – Estruturas do Roteador NDN.

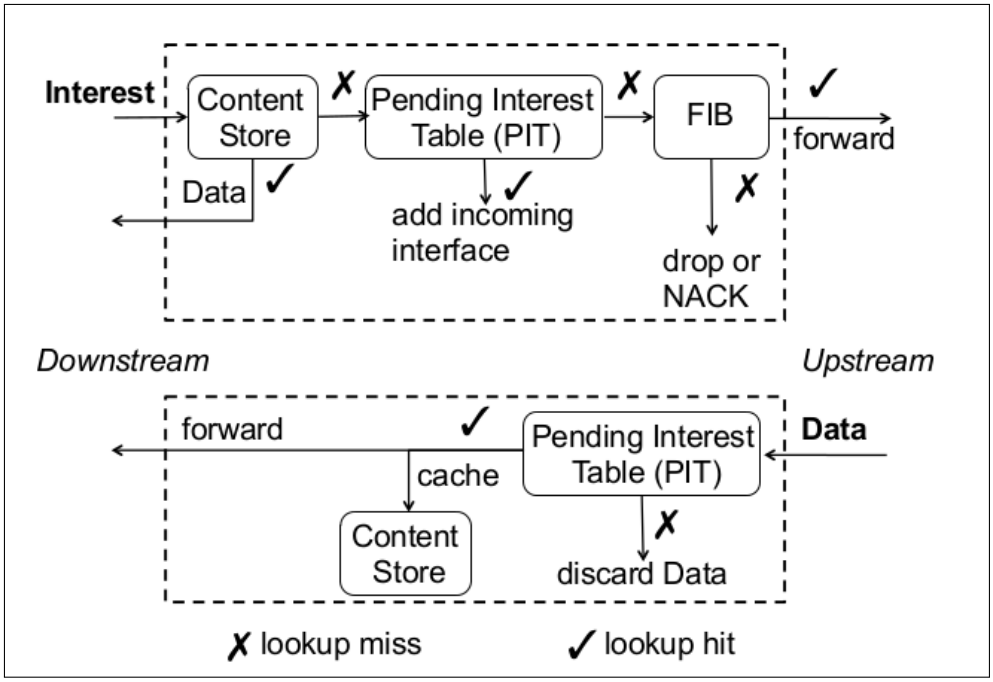


Fonte: (JACOBSON et al., 2009)

O encaminhamento é feito conforme mostrado na Figura 3.

Quando um *I-Packet* chega em um roteador, primeiramente checa se existe o *D-Packet* correspondente no CS. Caso contenha, o *I-Packet* é descartado e o *D-Packet* é devolvido pela interface por onde o *I-Packet* chegou. Caso não exista o *D-Packet* correspondente no CS, então o roteador checa se esse *I-Packet* já foi encaminhado por ele, fazendo uma busca da entrada na PIT. Caso contenha (já possui uma entrada para aquele I-Packet), o *I-Packet* é descartado e a interface de chegada é associada a ele. Dessa forma, quando o *D-Packet* correspondente voltar, uma única entrada de *I-Packet* é apagada na PIT e o *D-Packet* é encaminhado por todas as

Figura 3 – Processo de Encaminhamento no Roteador NDN.



Fonte: (ZHANG et al., 2014)

interfaces que solicitaram esse mesmo *I-Packet* no roteador. Caso não haja entrada na PIT, então é checada a FIB. Se existir uma correspondente nela, o *I-Packet* é encaminhado pela interface indicada e a entrada do *I-Packet* é adicionada na PIT. Caso não exista entrada na FIB, não existe caminho conhecido para esse *I-Packet*. Então, ele é descartado. Pode-se utilizar uma outra abordagem no caso de não haver entrada na FIB. A ideia é fazer *broadcast* em todas as interfaces disponíveis e isso se repetir até alcançar um roteador que tenha o *D-Packet* ou então, o próprio produtor do dado.

No caso da chegada de um *D-Packet*, primeiro é checado se existe uma requisição pendente deste *D-Packet* na PIT. Caso não haja, o *D-Packet* é descartado, pois não houve nenhum pedido do mesmo. Se existir uma entrada do *D-Packet* correspondente na PIT, então o mesmo é armazenado no CS e encaminhado pela interface que estava associada ao *I-Packet* na entrada da PIT. Esse processo vai se repetindo até o *D-Packet* chegar ao consumidor.

2.2 Delay Tolerant Network - DTN

A arquitetura TCP/IP conseguiu se consolidar para lidar com problemas enfrentados nas redes tradicionais. Todavia novos cenários de redes foram surgindo com dificuldades que essa arquitetura não teria como lidar adequadamente. Em (FALL, 2003) são ditas três características

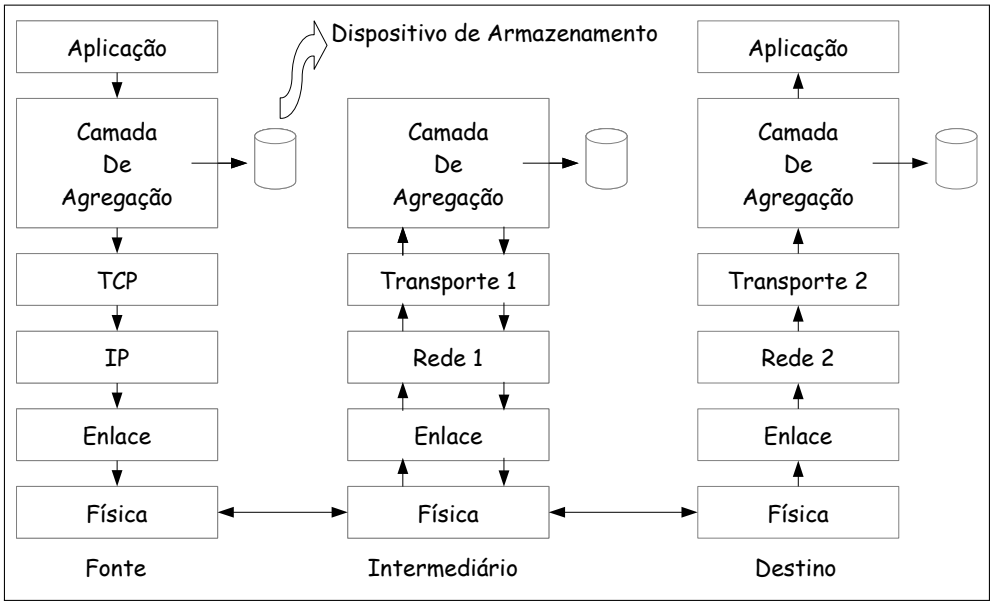
importantes para o bom funcionamento do TCP/IP: a existência de um caminho fim-a-fim entre nó fonte e destino, o valor máximo de *round trip time* não excessivo e uma baixa probabilidade de perda de pacotes. As chamadas redes desafiadoras violam uma ou mais dessas características. Algumas dessas redes incluem: redes móveis terrestres, redes com meio de transmissão exótico, redes ad hoc militares e redes sensores.

Nesse contexto, o problema com as redes TCP/IP é o tempo curto em que os pacotes ficam armazenados nos nós intermediários. Assim, em um ambiente em que haja muita desconexão, os pacotes serão frenquentemente perdidos, causando uma queda no desempenho (FILHO, 2014). Para lidar com as características das redes desafiadoras, o grupo *Delay Tolerant Network Research Group* (DTNRG) elaborou a arquitetura DTN (OLIVEIRA et al., 2007). A proposta da DTN é utilizar a estratégia chamada armazena-carrega-e-encaminha, onde um pacote é armazenado no *buffer* de um nó intermediário. Então, esse nó carrega o pacote até que encontre um outro nó para o qual ele possa reencaminhá-lo ou copiá-lo. Esse processo se repete até que o pacote chegue ao destino.

2.2.1 Arquitetura

Na arquitetura DTN, uma nova sobrecamada é colocada na pilha: a camada de agregação (*Bundle Layer*) conforme podemos ver na Figura 4. Nessa camada, é feita a persistência de

Figura 4 – A camada de agregação.



Fonte: (FILHO, 2014)

dados. Observe que a camada de agregação fica acima da camada de transporte. O motivo disso é poder utilizar qualquer tipo de pilha abaixo, mesmos redes que não sejam TCP/IP. Essa adaptação de diversos tipos de redes é feita pelas camadas de convergência (*Convergence Layers*), que funciona semelhante a *drivers* dentro de um sistema operacional (JAIN; PATRA, 2003). Como nem todos os protocolos que operam nas camadas abaixo da de agregação oferecem as mesmas funcionalidades, existe a necessidade das camadas de convergência que fica entra a camada de agregação e os protocolos das camadas de baixo para que assim a camada de agregação possa operar com diversos tipos de redes.

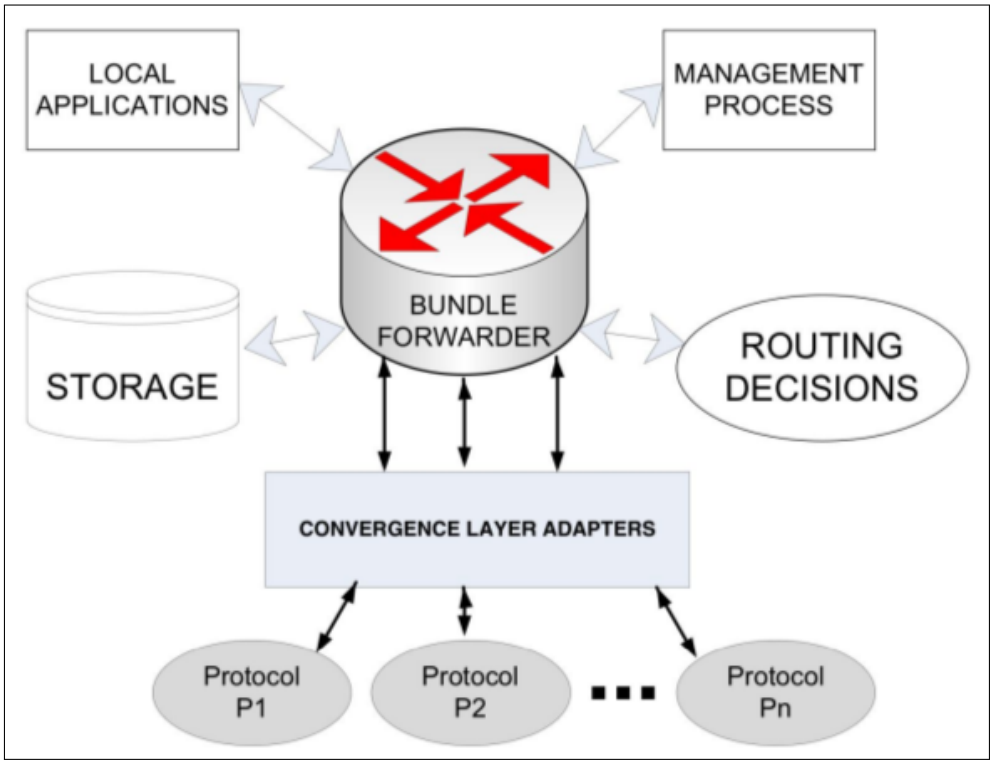
As Aplicações DTN geram mensagens de tamanhos variáveis chamadas *Application Data Units* (ADUs). Quando essas mensagens chegam a camada de agregação, são transformadas em *Protocol Data Units* (PDU) sendo chamadas de agregados. Eles é que são armazenados e encaminhados nos nós DTN. Na Figura 5 podemos ver uma implementação da arquitetura da camada de agregação. O *Bundle Forwarder* é a entidade que executa decisões de roteamento por meio da interação com as outras entidades (JAIN; PATRA, 2003), conforme mostrado na Figura 5.

2.2.1.1 Tipos de Contatos

Uma questão importante para a arquitetura são os tipos de contatos que podem ocorrer. Diferente do que acontece na Internet, as DTNs não assumem que todos os nós são alcançáveis. Dessa forma, é importante aproveitar ao máximo os contatos para realizar as trocas de dados, pois uma oportunidade futura pode não ocorrer. Os cinco tipos de contatos são (OLIVEIRA et al., 2007):

- **Persistentes:** São contatos que estão sempre disponíveis. Um exemplo seria uma rede *Digital Subscriber Line* (DSL).
- **Sob demanda:** São contatos que requerem alguma ação para serem ativados. Após eles serem ativados, funcionam da mesma forma que contato persistente. Um exemplo disso é o que pode ocorrer em uma rede de sensores na qual alguns sensores precisam de uma mensagem específica para acordar.
- **Programados:** São contatos agendados para ocorrer. Nesse caso, os nós sabem quando esses contatos irão ocorrer. A exigência para esse tipo de contato é que haja uma sincronização do tempo na rede. Um exemplo desse contato acontece em rede de sensores, onde um nó da rede pode ter um determinado horário para acordar e assim estabelecer contato

Figura 5 – Um exemplo de implementação de arquitetura que mostra como o *Bundle Forwarder* interage com o armazenamento, decisões de roteamento e as camadas de convergência para utilizar vários protocolos para entrega.



Fonte: (FALL; FARRELL, 2008)

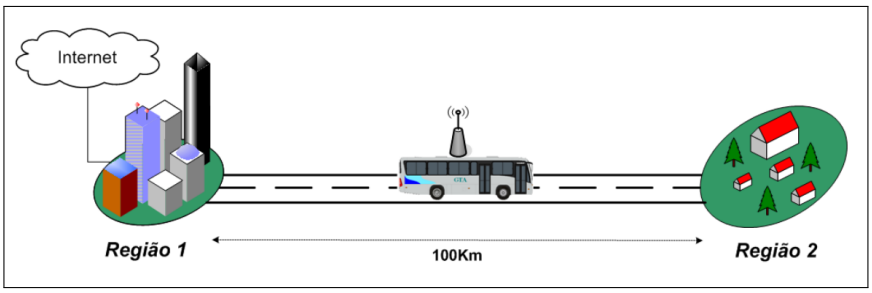
com um outro nó.

- **Previsíveis:** São contatos em que os nós podem ter uma certa previsão de horário de um possível contato. Um exemplo disso é uma rede rural, na qual um ônibus pode ser utilizado para carregar dados da Internet da cidade para um região do interior que não possui uma infraestrutura para fornecer Internet. Isso pode ser visto na Figura 6.
- **Oportunistas:** São contatos que ocorrem sem qualquer tipo de previsão. Nesse tipo de contato não é possível saber se o contato irá ocorrer ou em qual tempo. Dessa forma, os nós devem aproveitar ao máximo suas oportunidades. Como exemplo, imagine uma MANET em modelo *Random Way Point*: os nós se movem de maneira totalmente aleatória. Assim, não há nenhuma previsibilidade de contato.

2.2.1.2 Transferência de Custódia

Quando um nó DTN possui um agregado e estabelece contato com um outro nó, ele precisa decidir se irá manter, copiar ou encaminhar o agregado. Em primeira instância, podemos

Figura 6 – Exemplo de contato previsível.

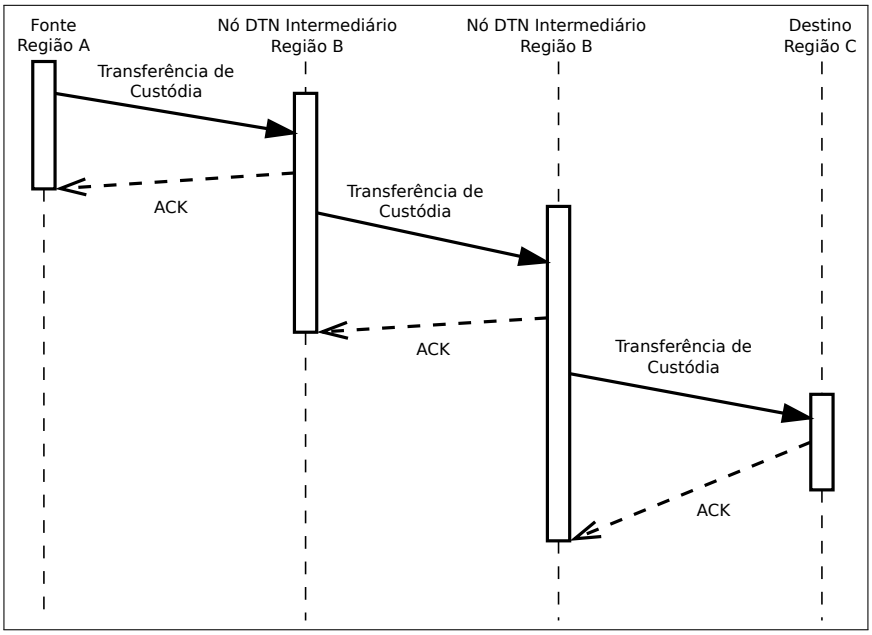


Fonte: (OLIVEIRA et al., 2007)

pensar que copiar seria a melhor solução, uma vez que mais cópias espalhadas na rede aumentam a possibilidade do agregado chegar ao seu destino. No entanto, para evitar uma sobrecarga nos *buffers*, podemos utilizar a transferência da custódia do agregado. Basicamente, um nó encaminha seu agregado para outro e então o deleta para liberar espaço no *buffer*.

Na Figura 7, podemos ver o mecanismo da transferência em ação. Um nó fonte *A*

Figura 7 – Mecanismo de transferência de custódia.



Fonte: (VIEIRA, 2012)

copia seu agregado para um nó intermediário *B*, mas *A* espera um *ACK* confirmando que o nó *B* recebeu e aceitou o agregado para que o nó *A* possa deletar o agregado do seu *buffer*. Esse aspecto estabelece uma transferência confiável ponto-a-ponto, ou seja, o nó *A* só apaga o seu agregado quando tiver certeza de que a custódia do agregado está com o nó *B*. Esse processo se repete até o agregado chegar no destino.

2.2.2 Desafios para Pesquisas

Nas redes desafiadoras em que as DTNs se encaixam, podem existir redes esparsas nas quais será muito difícil estabelecer um caminho fim-a-fim entre dois nós que querem trocar alguma informação. Contatos podem ser totalmente imprevisíveis. Além disso, o tempo de contato pode ser muito curto, tal como acontece numa VANET. Essas características geram desafios que os pesquisadores devem enfrentar ao implementar as DTNs. Em seguida, podemos ver alguns desses desafios (KHABBAZ et al., 2012) (FILHO, 2014):

- **Imprevisibilidade dos contatos:** Existem formas do nó DTN conseguir previsibilidade da rede. Um exemplo são os oráculos que é uma abstração que corresponde a dizer “a informação sobre o assunto está disponível para todos os nós"(OLIVEIRA et al., 2007). Eles funcionam em um ambiente DTN determinístico, todavia existem cenários estocásticos onde se tem pouco ou nenhum conhecimento sobre a rede. Assim, os protocolos de roteamento devem procurar meios para obter informações da rede e assim tomar melhores decisões de roteamento.
- **Gerenciamento de *buffer*:** A utilização do armazena-carrega-e-encaminha para lidar com o ambiente esparso requer o uso de armazenamento persistente nos nós intermediários. Dessa forma, irá surgir o problema de sobrecarga no *buffer*. Por isso é importante estabelecer políticas de descarte (caso de lotação do *buffer*), além disso, estabelecer quando é melhor copiar, encaminhar ou não realizar a replicação em uma oportunidade de contato.
- **Notificação de um agregado recebido no destino:** Quando um agregado chega ao destino, existem muitas cópias que não são mais úteis espalhadas pela rede. Estabelecer um modo de notificar os nós que estão com essas cópias não é fácil devido aos problemas de conexões das DTNs.
- **Tempo de vida para um agregado:** Todo agregado tem um tempo de vida que determina seu tempo de armazenamento no *buffer*. Uma escolha de tempo baixo pode fazer com que o agregado seja apagado antes que o destino seja encontrado. Todavia um tempo alto pode fazer com que o agregado fique muito tempo armazenado sem necessidade, pois o agregado pode já ter sido entregue ao destino.

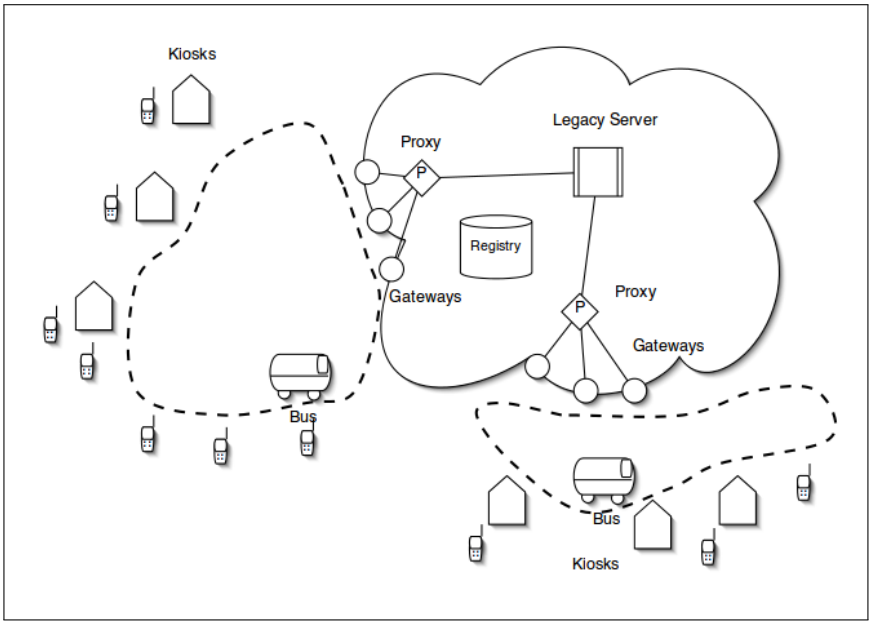
2.2.3 Aplicações

A arquitetura DTN pode ser utilizada para prover acesso não usual à Internet, monitoramento, comunicação submarina, comunicação em ambientes de batalhas e comunicação espacial (OLIVEIRA et al., 2007).

Diversas localidades do mundo não possuem uma infraestrutura para prover acesso à serviços da Internet, tais como correio eletrônico e web. Dessa forma, vários projetos com DTN foram criados para auxiliar essas regiões a serem inclusas no mundo digital. Um exemplo desse tipo de projeto é o KioskNet (GUO et al., 2011). Uma KioskNet consiste de um conjunto de *kiosks* que fornecem acesso a Internet por meios convencionais ou utilizando o armazena-carrega-e-encaminha de DTN.

Na Figura 8 podemos ter uma visão geral do KioskNet. Cada *kiosk* tem um controla-

Figura 8 – Visão geral do KioskNet.



Fonte: (GUO et al., 2011)

dor *kiosk* que é um servidor com um computador reciclado que é um computador que apenas precisa ter uma BIOS e uma interface ethernet que suporta boot PXE. Ele pode conseguir acesso à Internet via conexão dial-up e também utilizando, por exemplo, um ônibus para carregar os dados. O controlador *kiosk* possui uma antena WIFI que pode se comunicar com a antena WIFI do ônibus que carrega os dados. Outros usuários que possuem recursos como smartphones podem fazer acesso aos *kiosk*s ou diretamente aos ônibus sem precisar usar o computador reciclado.

O ônibus é um exemplo de entidade carregadora de dados. Ela poderia ser também um carro, moto ou trem. No projeto a carregadora de dados recebe nome de *ferry*. Um *ferry* consiste de um computador de bordo que é alimentado pela bateria do veículo. Ele possui uma capacidade de memória de 20-40GB e uma antena WIFI. O papel do *ferry* é pegar os dados dos *kiosks* e os levá-los até um dos *gateways* da região. Um *gateway* é um computador com uma interface WIFI, dispositivo de armazenamento e com acesso para Internet através DSL ou conexão a cabo. O papel do *gateway* é receber os dados vindo dos *ferries* e os mandar para Internet utilizando o *proxy*. O *proxy* é uma entidade residente na Internet que funciona nos dois lados. Um lado tem uma conexão tolerante a atrasos com aplicações sendo executados nos computadores reciclados ou nos dispositivos móveis. O outro lado estabelece comunicação com o *legacy server* que tipicamente é acessado utilizando TCP/IP e um protocolo da camada de aplicação, tais como IMAP, SMTP, or HTTP através do *proxy*. O papel do *proxy* é intermediar os dois lados. O primeiro local que recebeu uma implementação desse projeto foi uma vila da Índia em 2006.

Como exemplo de aplicação de monitoramento temos ZebraNet (JUANG et al., 2002), de comunicação submarina Seaweb (*Sensor Networking with Delay Tolerance*) (RICE, 2005) , de comunicação espacial uma arquitetura para a Internet Interplanetária (InterPlaNet - IPN) (PROJETO-IPN, 2016).

2.3 Vehicular Ad hoc Network - VANET

2.3.1 Definição

As VANETs são um subtipo das MANETs que surgiram para prover um sistema de comunicação para diferentes tipos de veículos. O objetivo principal desses sistemas é possibilitar as condições necessárias para que esses usuários móveis consigam comunicação, fazendo com que diferentes tipos de aplicações possam ser atendidas (ALVES et al., 2009).

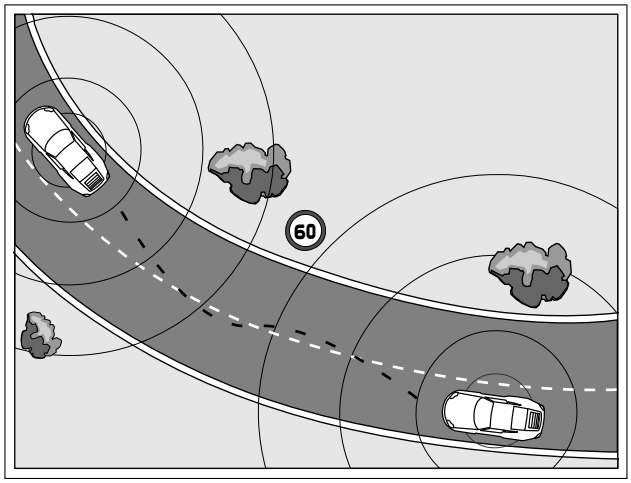
Apesar das semelhanças com as MANETs, as VANETs possuem características únicas (SINGH; AGRAWAL, 2014). Dessa forma, trabalhos propostos para MANETs devem ser repensados para considerar essas diferentes características.

2.3.2 Arquitetura

Cada veículo possui *On Board Unit* (OBU) que um é dispositivo utilizado para a comunicação com a *Road Side Unit* (RSU) que, por sua vez, é um dispositivo estacionário (FILHO, 2014) localizado ao longo das estradas. Assim, a arquitetura de comunicação entre os veículos pode ocorrer de três formas (ALVES et al., 2009): infraestruturada, *ad hoc* pura e híbrida, conforme são mostradas nas Figuras 9, 10 e 11.

Na comunicação *ad hoc* pura, os veículos se comunicam entre si sem a necessidade de qualquer infraestrutura centralizada, utilizando a abordagem *Vehicle-to-Vehicle* (V2V). Dessa forma, os veículos também passam a funcionar como roteadores intermediários que podem encaminhar mensagens de um nó fonte para um destino através de múltiplos saltos. A desvantagem

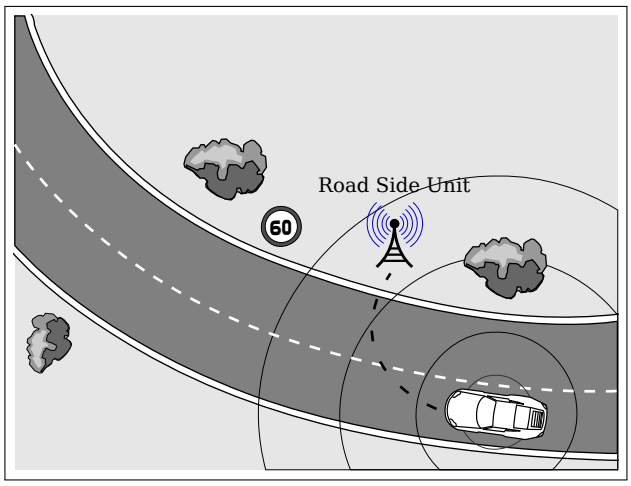
Figura 9 – Comunicação *ad hoc* pura.



Fonte: (VIEIRA, 2012)

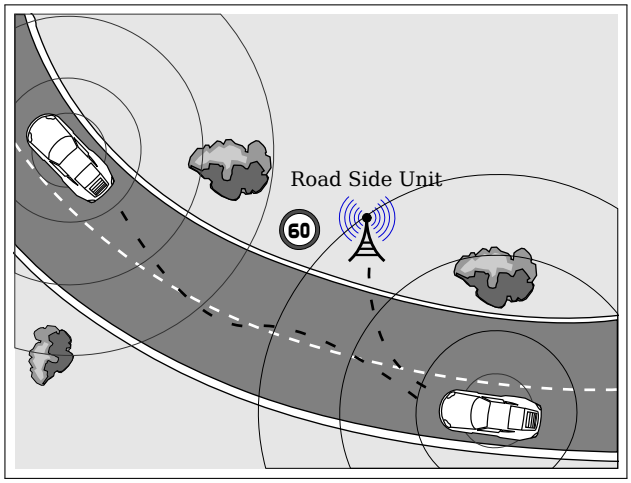
dessa abordagem é a dificuldade de manter a rede devido a alta mobilidade dos veículos. Para auxiliar na conectividade e também fornecer serviços como de ponto acesso de redes 802.11, as RSUs são utilizadas na comunicação infraestruturada: abordagem *Vehicle-to-Infrastructure* (V2I). As RSUs podem centralizar todo o tráfego da rede, servindo também como nós intermediários (ALVES et al., 2009). O problema de espalhar várias de RSUs pelas estradas é o custo alto. A arquitetura híbrida pode ser utilizada como uma solução intermediária, mantendo tanto as RSUs fixas provendo serviços por meio comunicação V2I como a utilização de veículos para uso múltiplos saltos com a comunicação V2V.

Figura 10 – Comunicação infraestruturada.



Fonte: (VIEIRA, 2012)

Figura 11 – Comunicação híbrida.



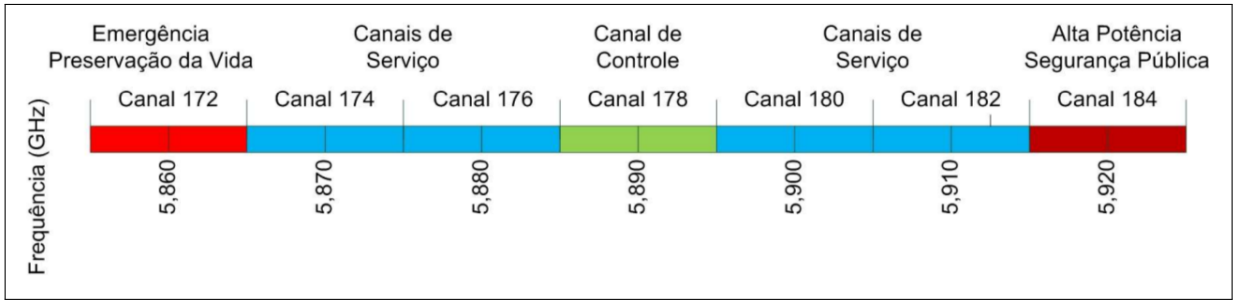
Fonte: (VIEIRA, 2012)

2.3.3 Padrões de Redes Veiculares

Os primeiros padrões para redes veiculares foram desenvolvidos nos Estados Unidos em 1999. A *Federal Communications Commission* (FCC) alocou 75 MHz de espectro de frequências, na faixa de 5,9 GHz, para aplicações *Dedicated Short Range Communications* (DSRC), conforme a Figura 12, na qual podemos observar dois tipos de canais, o *Control Channel* (CCH) e *Service Channel* (SCH). Os canais 172 e 184 são reservados para aplicações de altos riscos.

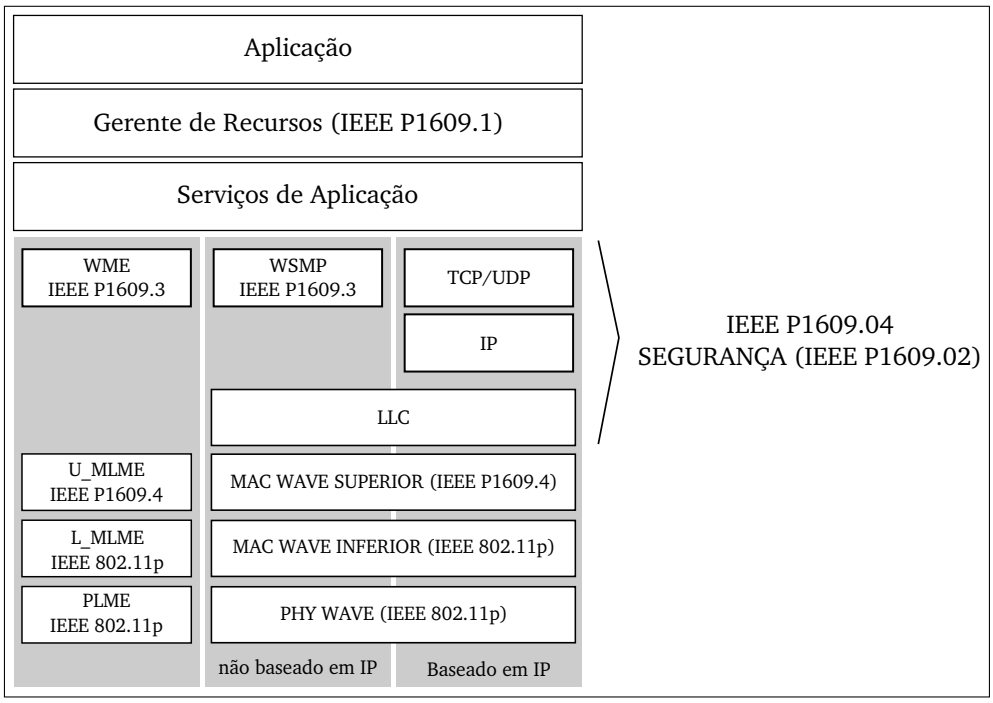
Em 2004, O IEEE iniciou a padronização das comunicações em VANETs no grupo de trabalho do IEEE 802.11. O padrão ficou então conhecido como 802.11p *Wireless Acess in the Vehicular Environment* (WAVE), cuja a pilha pode ser vista na Figura 13. Essa arquitetura é

Figura 12 – Alocação de espectro para aplicações DSRC.



Fonte: (ALVES et al., 2009)

Figura 13 – Arquitetura WAVE.



Fonte: (VIEIRA, 2010)

definida nos seguintes documentos (ALVES et al., 2009) (VIEIRA, 2012):

- **IEEE P1609.1 - Gerente de Recursos:** Especifica como deve ser realizada a comunicação entre *Resource Manager Application* (RMAs) e OBUs através das RSUs. RMAs são aplicações remotas fora da RSU que precisam acessar recursos nas OBUs. Esse acesso é solicitado ao *Resource Manager* (RM) que se comunica com o *Resource Command Processor* (RCP) o qual é localizado nas OBUs e faz controle de outros dispositivos do veículo, tais como memória, interfaces de usuários e interfaces com outros dispositivos no veículo. Dessa forma, o RM tem o papel de multiplexar diversos pedidos dos RMAs aos OBUs. O objetivo desse padrão é favorecer a interoperabilidade das aplicações WAVE, buscando reduzir custos e aumentar o desempenho.

- **IEEE P1609.2 - Segurança:** Define os formatos e processamentos para que as mensagens possam trafegar de maneira segura. Define também quando devem ser usadas mensagens com segurança e a maneira que ela deve ser processada, de acordo com a situação. O padrão também estabelece o uso de ferramentas tradicionais para segurança, tais como chaves públicas e certificação.
- **IEEE P1609.3 - Serviços de Rede:** Especifica serviços para a camada de controle lógico (*Logical Link Control* - LLC), para camada de transporte e para camada de rede. Para transporte e rede, podem ser usadas a pilha com IPV6/UDP/TCP ou mensagens curtas WAVE (*WAVE Short Messages*). Como a mobilidade dos veículos pode ser alta, fazendo com que os contatos sejam bem curtos, a mensagem curta pode ser utilizada por ser mais eficiente. Definições da *Management Information Base* (MIB) também são encontradas nesse documento.
- **IEEE P1609.4 - Operação de Múltiplos Canais:** Especifica como são utilizados o CCH e os múltiplos SCHs. Um dispositivo WAVE fica na escuta no CCH esperando algum anúncio de serviço. Nesse anúncio, vem especificado em qual SCH está o serviço. Um intervalo de tempo é sempre reservado para o CCH. Dessa forma, ele pode esperar anúncio de outros serviços. Esse intervalo de revezamento entre os canais é necessário por que o dispositivo opera apenas com uma interface de rádio.
- **IEEE 802.11p:** Estende a família de protocolos do padrão IEEE 802.11. Baseia-se, especificamente, no IEEE 802.11a, porém operando na faixa DSRC de 5,9 GHz.

2.3.4 Características

As VANETs possuem características que as diferem de outras MANETs. Podemos considerar algumas delas (AL-SULTAN et al., 2013) (FILHO, 2014):

- **Mobilidade Previsível:** As MANETs, em geral, movem-se no modelo *Random Way Point*. No caso das VANETs, é possível saber quais caminhos um nó pode tomar, bastando para isso conhecer a topologia das rodovias. Essa característica pode facilitar as decisões dos protocolos de roteamento na hora de decidir para quem encaminhar um pacote.
- **Não há restrições de energia:** Em geral, as MANETs possuem restrições de energia. Um exemplo são as redes de sensores sem fio (AKYILDIZ et al., 2002), nas quais existe uma preocupação alta com o consumo de energia. No caso das VANETs, a OBU é alimentada pela bateria do veículo. Dessa forma, enquanto o veículo estiver funcionando, sempre

haverá energia para manter a OBU.

- **Densidade da rede variável:** A densidade da rede vai depender do tráfego atual do trânsito. Enquanto existem áreas onde a densidade é muito alta, existem outras áreas onde a densidade será muito baixa.
- **Rápidas mudanças de topologia:** Como os veículos podem atingir altas velocidades, os contatos podem ser bem curtos, fazendo com que a topologia da rede fique sempre se modificando. Essa característica afeta diretamente o TCP, pois a constante mudança de topologia faz com que o TCP pense que está ocorrendo congestionamento por causa dos vários pacotes perdidos. Dessa forma, a janela de congestionamento sofre uma diminuição desnecessária causando perda de desempenho. Em (FU et al., 2002) é mostrado como o TCP é degradado devido a mobilidade da rede.
- **Possibilidade de alto poder computacional:** Como os nós das VANETs são veículos, eles podem ser equipados com diversos recursos computacionais. Devido à isso, protocolos como o VDTN-ToD (VIEIRA et al., 2013) e o GPSR (KARP; KUNG, 2000) podem utilizar *Global Position System* (GPS) para obter a localização do nó destinatário e utilizá-lo para auxiliar nas decisões de roteamento.

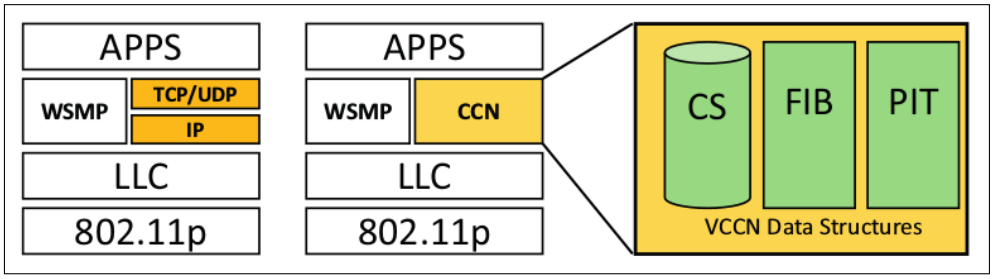
2.3.5 Aplicações

No trabalho de (YOUSEFI et al., 2006), as aplicações são divididas em duas classes : segurança e conforto. Como alguns exemplos de aplicações de segurança temos a detecção de colisão, o aviso de problemas na estrada e a assistência para mudança de faixa e ultrapassagem. Os tipos de mensagens de segurança são enviados por meio do CCH. Elas podem ser de dois tipos: eventos do motorista ou periódicas. A primeira detecta eventos que podem ocasionar acidentes, por exemplo: um carro muito próximo de outro, velocidade alta, etc. No caso das mensagens periódicas, as mensagens contém informações preventivas as quais podem ser utilizadas por aplicações que também não são de segurança. Tais mensagens podem ser de monitoramento de tráfego, ou mensagens de protocolos de roteamento e *beacons*. Como exemplo de aplicações de conforto, temos o acesso a Internet. Os passageiros do veículo podem querer enviar e-mails e nesse caso é preciso RSUs que funcionem como pontos de acesso.

2.3.6 Vehicular Content Centric Network (VCCN)

Com o surgimento das CCNs, alguns trabalhos propõem juntar as tecnologias de VANET com CCN (AMADEO et al., 2012) (GRASSI et al., 2014). Em (BOUK et al., 2015) são abordadas questões referentes as pesquisas sobre VCCN. De acordo com o autor, a arquitetura da VCCN consiste em tirar a pilha TCP/IP e substituir por uma com as estruturas de CCN, conforme pode ser visto na Figura 14. O ambiente CCN dentro das VANETs deve buscar considerar as

Figura 14 – Arquitetura VCCN.



Fonte: (BOUK et al., 2015)

restrições do 802.11p, mobilidade , etc. No modo como o *I-Packet* será difundido, deve-se reavaliar questões de conectividade, pois rapidamente um link pode se quebrar dificultando que o *I-Packet* encontre quem tenha seu respectivo *D-Packet*.

Em (AMADEO et al., 2012) são apresentados benefícios que a CCN representa para VANET quando substitui a pilha tradicional da Internet. São eles:

- **Configuração zero da rede:** O paradigma CCN permite a comunicação entre os nós, sem a necessidade de configuração de parâmetros de rede, tais como endereço IP, nome do servidor. A vantagem é que isso ajuda muito em um ambiente no qual a rede é não estática e muda frenquentemente, tais como são as VANETs.
- **Correspondência com a natureza das aplicações:** Nas aplicações em geral, os veículos da rede estão interessados em obter informações como a situação da rodovia, os avisos de acidentes, etc. Tudo isso independentemente da identidade ou endereço IP de quem está gerando a informação, ou seja, o interesse está apenas no conteúdo.
- **Facilidade do uso de *cache*:** O uso de *cache* de CCN em VANET, é facilmente aplicável devido aos recursos de energia e armazenamento dos dispositivos das VANETs.
- **Manutenção de serviços mesmo quando os links são intermitentes:** Com o uso de *cache*, um determinado veículo pode conseguir um conteúdo mesmo que o produtor não

possa ser alcançado bastando para isso que um veículo intermediário possua cópia do conteúdo em *cache*.

2.4 MECANISMOS

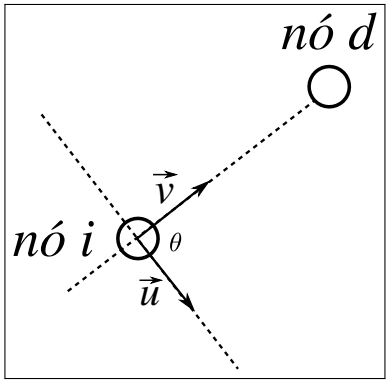
2.4.1 Trend Of Delivery (ToD)

O mecanismo ToD foi proposto e utilizado nos nossos trabalhos anteriores (VIEIRA et al., 2013) (FILHO et al., 2014). A finalidade desse mecanismo foi de auxiliar o protocolo de roteamento VDTN-ToD nas decisões de roteamento. Além disso, serviu para melhorar o desempenho dos protocolos *Spray-And-Wait* (SPYROPOULOS et al., 2005) e *PROPHET* (LINDGREN et al., 2003) em VANETs. Com o ToD um determinado nó da rede pode estimar quando é melhor: manter, encaminhar ou copiar um pacote, visando obter uma melhor taxa de entrega e diminuir o atraso da mensagem.

O valor do ToD é obtido a partir dos valores de mobilidade dos nós e a utilização da lógica *fuzzy*. O valor do ToD indica o quão bom o vizinho é para receber um determinado pacote. O ToD é baseado em três valores: o sentido ($\omega_{i,d}$), a distância ($\Psi_{i,d}$) e a velocidade ($\tau_{i,d}$), onde i é o nó que esta com a mensagem e d é o destino final da mensagem.

O sentido indica o quanto o sentido do nó se aproxima do destino, ou seja, o nó que terá maior valor será aquele que a direção tende a se aproximar mais do destino. Dessa forma, é calculado um ângulo θ formado entre o vetor direção $\overrightarrow{u}$ que indica o sentido de i e o vetor voltado para o destinatário $\overrightarrow{v}$(Figura 15). Assim, o ângulo entre eles indica quão bom ou ruim é o valor ($\omega_{i,d}$). Os valores associados são: péssimo, ruim, bom e ótimo.

Figura 15 – Representação do ângulo formado entre o vetor direção do nó e o vetor direção desejado.



Fonte: (VIEIRA et al., 2013)

Para a variável linguística distância foram definidas quatro categorias, são elas: muito perto, perto, longe e muito longe. As funções de pertinência de cada variável são definidas da seguinte forma:

1. muito perto: $x < R$
2. perto: $R < x \leq 2 \cdot R$
3. longe: $2 \cdot R < x \leq 3 \cdot R$
4. muito longe: $x > 3 \cdot R$

Onde R é o raio de transmissão.

A velocidade utilizada é a velocidade decomposta ($v_{i,x}$) do nó i, pois o valor dessa velocidade irá indicar o quão rápido o nó i se aproxima ou se afasta do destinatário. Ela é calculada conforme a Equação 2.1.

$$v_{i,x} = v_i cos(\theta) \tag{2.1}$$

Para velocidade são utilizadas três categorias: baixa, média e alta.

Com os valores calculados ($\omega_{i,d}$), ($\Psi_{i,d}$) e ($\tau_{i,d}$). O valores do ToD são definidos em sete: máximo (MA), ótimo (OT), muito bom (MB), bom (BO), ruim (RU), muito ruim (MR) e péssimo (PE). O valor do ToD será máximo quando a distância de um nó ao destino for menor que o alcance de transmissão. Para gerar as regras *fuzzy* seguiu-se os princípios:

1. Quanto mais próximo um nó estiver do destinatário, e quanto mais veloz ele se aproximar, melhor será seu valor de ToD.
2. Se um nó se distancia do destinatário, apesar disto representar um valor de ToD ruim, quanto mais próximo ele estiver do destinatário, e quanto mais lento ele se afastar, melhor será seu valor de ToD em relação aos outros nós com valor de ToD ruins.

A ideia básica desse princípio é que é nó deve se afastar lentamente do destinatário e se aproximar rapidamente. As regras são exibidas na Tabela 1, nela vemos o ToD sendo definido através dos três valores ($\omega_{i,d}$), ($\Psi_{i,d}$) e ($\tau_{i,d}$) em conjunto.

Utilizando o processo de defuzzificação centroide (ou centro de gravidade) é obtido o valor de ToD entre 0 e 1.

2.4.2 Aprendizagem de Máquina por Reforço Utilizando Q-Learning

Antes de falar da abordagem por reforço, precisamos definir o conceito de aprendizado e o de agente. Aprendizado denota mudanças no sistema que permitem que ao sistema

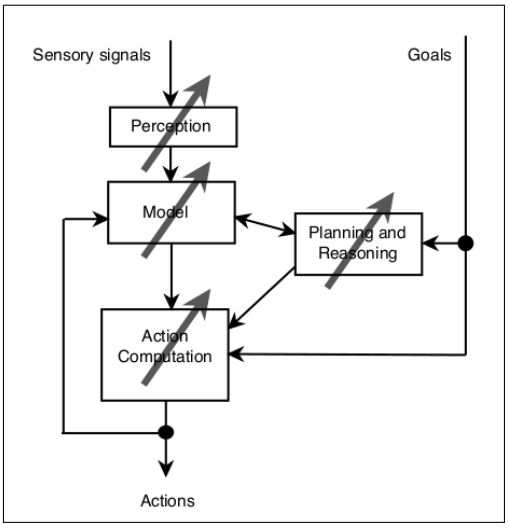
Tabela 1 – Regras *fuzzy* para definição do ToD (VIEIRA, 2012)

		Distância											
		Muito Perto			Perto			Longe			Muito Longe		
		Velocidade			**Velocidade**			**Velocidade**			**Velocidade**		
		Alta	Média	Baixa	Alta	Média	Baixa	Alta	Média	Baixa	Alta	Média	Baixa
Sentido	Ótima	MA	MA	MA	OT	OT	MB	MB	MB	BO	BO	RU	RU
	Boa	MA	MA	MA	MB	MB	BO	BO	BO	RU	BO	RU	RU
	Ruim	MA	MA	MA	RU	RU	BO	MR	MR	RU	MR	MR	MR
	Péssima	MA	MA	MA	MR	RU	RU	PE	MR	MR	PE	PE	PE

realizar a mesma tarefa ou demais oriundas da mesma população mais eficiente na próxima vez (SIMON, 1983). No caso da a aprendizagem de máquina, ela usualmente refere-se à mudanças em sistemas que executam tarefas associadas com inteligência artificial (IA). Tais tarefas podem envolver: reconhecimento, diagnóstico, planejamento, controle robótico, predição, dentre outras.

Para o entendimento sobre agente, vamos observar uma arquitetura típica de agente em IA. Um agente é algo que pode ser visualizado com percepção de seu ambiente através de sensores e atuando sobre o ambiente através de atuadores (RUSELL; NORVIG, 2003). Um agente humano tem olhos, ouvidos e outros órgãos para sensores. Mãos, pernas, boca e outras partes do corpo para atuadores. Um agente robótico deve ter câmeras, detectores infra-vermelhos para sensores e vários motores como atuadores. A Figura 16 mostra uma arquitetura típica de um

Figura 16 – Uma arquitetura típica de um agente em IA.



Fonte: (NILSSON, 2005)

agente em IA. Qualquer mudança em um dos componentes da Figura 16 pode ser considerada

como aprendizado. Dependendo de qual subsistema se queira alterar, é escolhido um diferente tipo de mecanismo de aprendizado. Neste trabalho, é utilizado o mecanismo de aprendizado por reforço utilizando o agente Q-Learning.

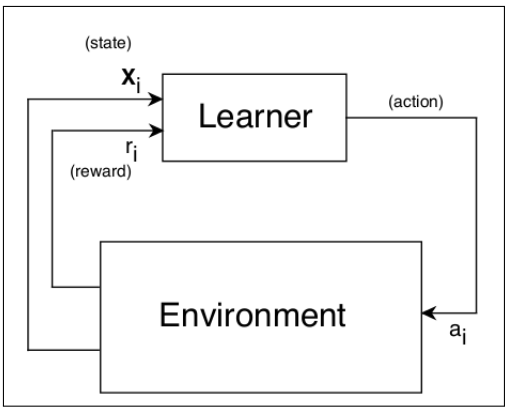
Na aprendizagem por reforço, o agente precisa aprender a ação que deve realizar em cada circunstância iterativa. O agente aprende por meio de um *feedback* que o ambiente fornece dada uma ação executada no mesmo por meio dele. Esse *feedback* é chamado de recompensa ou reforço. Exemplo: um agente que precisa aprender a jogar xadrez, pode ter uma recompensa positiva quando suas jogadas resultam em ganho e negativa quando resultam em perda. Através disso, o agente vai montando uma política para ganhar no jogo, ou seja, essa política é construída por tentativa e erro.

Formalmente, um modelo de aprendizagem por reforço consiste em:

- Um conjunto discreto *S* de estados do ambiente.
- Um conjunto discreto *A* de ações do agente.
- Um conjunto escalar *R* de sinais de reforço.

A Figura 17 mostra como é o funcionamento básico da aprendizagem por reforço. Inicialmente,

Figura 17 – Modelo de aprendizagem por reforço.

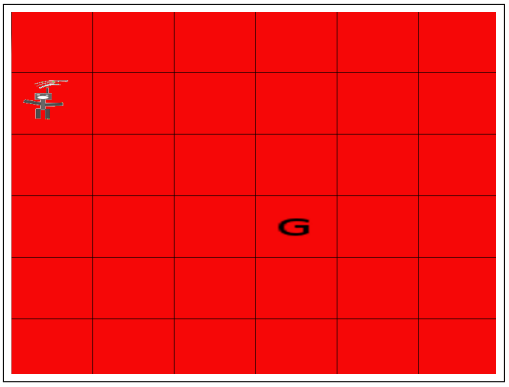


Fonte: (NILSSON, 2005)

o agente (*Learner*) não conhece nada do ambiente e possui um conjunto de ações que podem ser executadas. Dessa forma, ele executa uma determinada ação a_i no ambiente que responde com valor de recompensa r_i e um valor x_i com seu estado. Assim, o agente vai executando suas ações e aprendendo como ele deve agir no ambiente.

Podemos utilizar a aprendizagem por reforço para o ambiente da Figura 18 como exemplo. Nesse modelo temos uma grade 6x6. Nele, o robô tem o objetivo de alcançar o quadrado com G da maneira mais rápida possível. Podemos então fazer a modelagem da seguinte

Figura 18 – Exemplo com uma grade 6x6.



Fonte: Elaborado pelo autor

forma:

- $S = \{(i, j) \mid \text{i,j} = 1, 2, 3, 4, 5, 6\}$
- $A = \{\text{Direita, Esquerda, Cima, Baixo}\}$
- $R = \{R_{i,j} \mid \text{i,j} = 1, 2, 3, 4, 5, 6 \mid R_{i,j} = 1 \text{ ou } R_{i,j} = -1\}$

Onde o conjunto de estados S depende da posição atual (i, j) do robô. As ações possíveis são mover nas 4 direções em cada quadrado. No caso, a recompensa será positiva quando o robô se aproximar do objetivo e negativa quando se afastar.

O tipo de agente utilizado nesse trabalho é o Q-Learning. Agentes Q-Learning aprendem a função ação-valor (função-Q), dada uma utilidade esperada de uma ação em um determinado estado. Quando é utilizado o aprendizado de diferença temporal, o Q-Learning não necessita conhecer o modelo de transições de estado do ambiente. Neste caso, o ambiente fornece a conexão entre os estados vizinhos sob a forma de transições observadas. A desvantagem do Q-Learning é que ele converge de forma mais lenta que outras abordagens, tal como o Agente *Adaptive Dynamic Programming* (ADP) (RUSELL; NORVIG, 2003), porém o Q-Learning evita ter que aprender o modelo de transição.

Para o agente Q-Learning definir a ação que irá tomar no ambiente, é utilizada a notação $Q(a,s)$ para denotar o valor de utilidade da ação a no estado s. Dessa forma, a ação é escolhida conforme a Equação 2.2.

$$a_{escolhida} = Max_a Q(a,s) \tag{2.2}$$

O valor da função $Q(a,s)$ é atualizada conforme a Equação 2.3.

$$Q(a,s) = Q(a,s) + \alpha(R(s) + \gamma Max_{a'} Q(a',s') - Q(a,s)) \tag{2.3}$$

Onde:

- a' e s' representam o próximo estado e a próxima ação.
- $R(s)$ é a recompensa no estado s.
- $\gamma = [0,1]$ é o fator de desconto temporal.
- $\alpha = [0,1]$ é a taxa de aprendizado.

Esse cálculo é feito sempre que a ação a é executada no estado s levando para o estado s'.

3 TRABALHOS RELACIONADOS

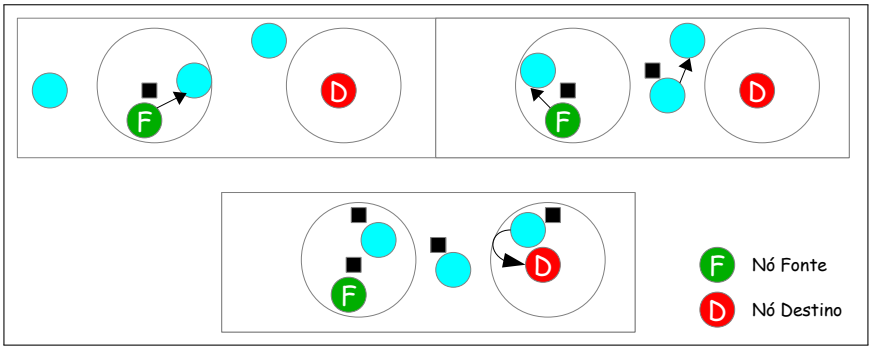
3.1 PROTOCOLOS DE ROTEAMENTO DTN

Vários protocolos de roteamento DTN já foram propostos na literatura. Dentre eles, os protocolos clássicos Epidêmico (VAHDAT et al., 2000) e *Spray-And-Wait* (SPYROPOULOS et al., 2005). Um protocolo mais recente é o Delay Tolerant Reinforcement-Based (DTRB) (ROLLA; CURADO, 2013). Em seguida, são descritos esses três protocolos de roteamento DTN.

3.1.1 Protocolo Epidêmico

O protocolo Epidêmico funciona tentando espalhar o máximo possível de cópias do pacote para aumentar a possibilidade do pacote chegar no nó destino. Quando um nó *A* encontra um nó *B*, eles trocam a lista de pacotes que eles possuem. Dessa forma, *A* copia para *B* todos os pacotes que *B* não possui e vice-versa. O funcionamento do Epidêmico pode ser visto na Figura 19.

Figura 19 – Operação do protocolo Epidêmico.



Fonte: (FILHO, 2014)

A abordagem do protocolo Epidêmico pode gerar uma ótima taxa de entrega (quando os recursos da rede não são limitados), mas ele pode lotar de maneira desnecessária o *buffer*, gerando desperdício de recursos da memória. Desta forma, o protocolo não é escalável.

Adaptando o protocolo Epidêmico para a o módulo de roteamento da arquitetura VDTCON, foi implementado o VDTCON-Epidemic.

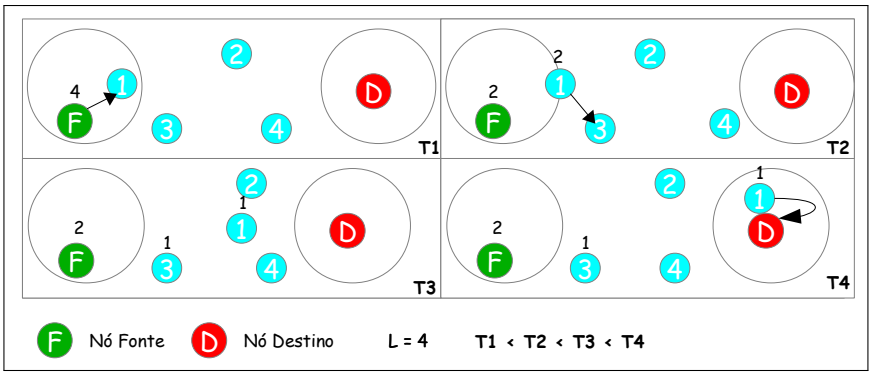
3.1.2 Protocolo Spray-And-Wait

Semelhante ao protocolo Epidêmico, o *Spray-And-Wait* espalha várias cópias dos pacotes, todavia ele estabelece um valor L que irá limitar o número de vezes que ele poderá copiar. O funcionamento acontece em duas fases:

- Fase *Spray*: Nessa fase, quando um nó F gera um pacote, ele o inicializa com valor L. Assim, toda vez que ele encontra um outro nó V que não possui o pacote, F diminui o valor de L e copia o pacote para V. Dessa forma, V passa a ter também a custódia do pacote com valor de L diminuído. Quando o valor de L chegar a 1, então o pacote não poderá ser mais copiado, ele entra na fase *wait*.
- Fase *Wait*: Nessa fase, o pacote só será transferido quando for encontrado o nó destino.

Uma das abordagens utilizada para saber quantos pacotes serão retidos e quantos copiados é a chamada binária. Seu funcionamento é mostrado na Figura 20. No instante T1 o

Figura 20 – Funcionamento *Spray-And-Wait* Binário.



Fonte: (FILHO, 2014)

nó fonte gera um pacote e L é inicializado com 4, quando ele encontra o nó 1, o nó fonte deve reter L/2 cópias e transmitir as L/2 restante. Nesse caso, cada um ficará com duas cópias. Em T2 o nó 1 encontra o nó 3, então o mesmo processo é executado e ambos agora ficam cada um com apenas uma cópia e entram no modo *wait*. Como o nó 1 entrou em modo *wait*, então no instante T3 em que ele encontra os nós 2 e 4, ele não copia o pacote para nenhum dos dois, ele só transfere o pacote quando finalmente encontra o destino no instante T4. Apesar de usar uma abordagem semelhante ao do Epidêmico, o *Spray-And-Wait* consegue uma boa escalabilidade.

Adaptando o Spray-And-Wait para o módulo de roteamento da arquitetura VDTCON, foi implementado o VDTCON-Spray.

3.1.3 Protocolo Delay Tolerant Reinforcement-Based Routing - DTRB

O protocolo DTRB foi proposto para um ambiente urbano com redes sem fio 802.11. Ele utiliza uma técnica de multiagente de aprendizagem por reforço (KAELBLING et al., 1996) para aprender sobre rotas na rede e replicar os pacotes DTN que produzem melhores recompensas. Para montar as informações necessárias aos seus cálculos, os nós DTRB trocam conhecimento por meio de mensagens de controle em *broadcast* que carregam dois pedaços de informações: *tabela de distância* (que indica a distância em função de tempo do nó fonte para todos os nós conhecidos) e a *recompensa* oferecida para uma determinada troca de dados.

No DTRB o modelo de multiagente de aprendizagem por reforço consiste em:

- Um conjunto discreto de S estados do ambiente, cada nó $s \in S$ na rede é considerado um estado do agente. O conjunto de todos os nós da rede é o espaço de estado.
- Um conjunto discreto de A ações dos agentes. No DTRB, cada pacote DTN é um agente indexado por nó fonte e destino. O possível conjunto de ações permitidos em cada nó é o conjunto de vizinhos em um determinado tempo.
- Um conjunto escalar de reforços R que possui um desvanecimento exponencial.

O algoritmo de aprendizado DTRB é inspirado no Q-Learning nele cada nó mantém uma tabela de valores-Q $Q(d,x)$, onde d é o nó destino do pacote e x é o próximo salto. As informações necessárias para o cálculo de Q são trocadas nas mensagens de controle. Dessa forma, $Q_s(d,x)$ é o valor estimado de praticabilidade de entrega da mensagem que sai de s para chegar em d através de x. Quando o nó s recebe uma mensagem de controle do nó x, então s atualiza seu $Q_s(d,x)$ conforme a Equação 3.1.

$$Q_s(d,x) = (1-\alpha)Q_s(d,x) + \alpha\{R + \gamma[max_y \in N_x Q_x(d,y)]\} \tag{3.1}$$

O valor da recompensa é calculada conforme a Equação 3.2.

$$R = \begin{cases} e^{-k}, & \text{se } 0 < D(x,y) < k. \\ 0, & \text{se } D(x,y) > k, \text{ onde } y \in N_x. \end{cases} \tag{3.2}$$

Onde $D(x,y)$ é a distância temporal entre os nós x e y, e k é um valor de tempo determinado em segundos. Um ponto importante é que o nó s faz um envelhecimento do seus valores Q, conforme a Equação 3.3.

$$Q_s(d,x) = Q_s(d,x)\omega^{\mu} \tag{3.3}$$

Onde ω é a constante de envelhecimento e μ é quantidade de tempo passado desde a última medida.

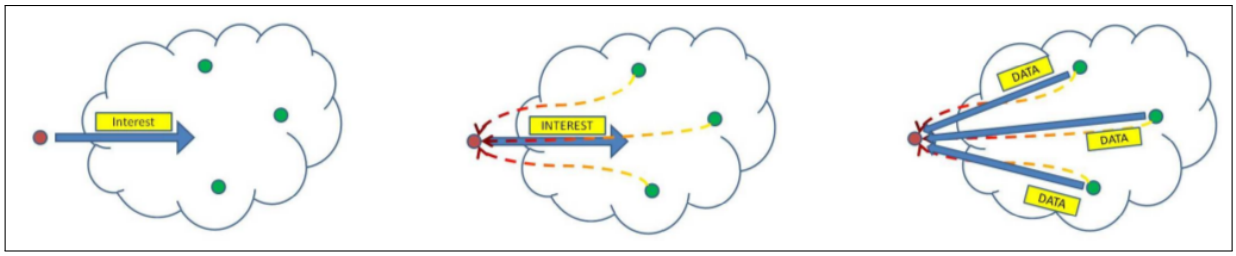
Neste trabalho, foi desenvolvido o protocolo VDTCON-QLR que também utiliza aprendizado por reforço com agente Q-Learning, mas a abordagem do cálculo do reforço é feita de maneira diferente. Para calcular o valor de reforço, é utilizado um valor referente ao contato com nó destinatário e também é utilizado o valor da métrica *Trend Of Delivery* (ToD) (VIEIRA et al., 2013).

3.2 TRABALHOS DE DTN COM CONTENT-ORIENTED NETWORKS

3.2.1 STIgmergy Routing

Em (NGUYEN et al., 2010) é proposto um protocolo de roteamento para *Content-Centric Delay-Tolerant Networks* (CCDTN) que utiliza um mecanismo que explora a estigmergia entre os nós móveis da rede. Esse protocolo é chamado *STIgmergy Routing* (STIR). O STIR funciona em cima de três mecanismos: difusão de interesses, gerenciamento de gradiente e difusão de conteúdo, conforme a Figura 21.

Figura 21 – Os três mecanismos do STIR.



Fonte: (NGUYEN et al., 2010)

Na difusão de interesses, o objetivo é fazer o interesse chegar ao produtor do conteúdo de maneira eficiente. Ele foi inspirado no protocolo de roteamento DTN clássico *Spray-And-Wait* Binário (SPYROPOULOS et al., 2005). A ideia consiste em limitar o número de vezes que um interesse será difundido. O Algoritmo 1 mostra como a difusão de interesses é feita.

O gerenciamento de gradiente tenta manter com coerência a função de utilidade $U(c,n)$ que é a utilidade do conteúdo c no nó n. Esse valor reflete a proximidade em tempo e espaço entre o usuário do conteúdo e o nó n considerado, em outras palavras, ele indica a frequência com que o nó n encontra o usuário do conteúdo. O mecanismo de gerenciamento de

Algoritmo 1: Algoritmo de difusão de interesses

```
Entrada: Número de cópias de interesses = L
início
    while Não receber dados do
        para Cada contato faça
            se O outro nó esta carregando dados então
                Requesite e receba o dado;
            fim
            senão
                se Número de cópias de interesses > 1 então
                    Compartilhe metade das cópias com o nó encontrado;
                fim
            fim
        fim
    end
fim
```

gradiente mantém:

- Que o usuário de conteúdo c mantém o valor máximo de $U_{max}(c)$.
- O produtor do conteúdo c mantém o valor mínimo de $U_{min}(c)$.
- Os nós que carregam dados mantém o valor no intervalo dentro de $[U_{min}(c), U_{max}(c)]$.

Quando ocorre um contato entres os nós i e j, o valor de utilidade é atualizado conforme a Equação 3.4.

$$U_{novo}(c,i) = \alpha.U_{contato}(c,j) + (1-\alpha).U_{antigo}(c,i) \tag{3.4}$$

Onde $\alpha \in [0,1]$ é o coeficiente de reforço, $U_{novo}(c,i)$ é o valor atualizado de utilidade de i, $U_{old}(c,i)$ é o valor antigo de utilidade de i e $U_{contato}(c,j)$ é o valor de utilidade do nó j encontrado. O valor de utilidade também sofre envelhecimento a cada contato, ele ocorre conforme a Equação 3.5.

$$U_{novo}(c,i) = U_{old}(c,i).\beta^t \tag{3.5}$$

Onde $\beta \in [0,1]$ é a constante de envelhecimento e t é o tempo decorrido desde o último contato.

No mecanismo de difusão de conteúdo também é utilizada a abordagem do *Spray-And-Wait* Binário. No caso da decisão de roteamento de um nó n, um determinado nó v vizinho

só irá receber a cópia de um conteúdo de *n* se *v* tiver um valor de utilidade maior que o de *n*. O Algoritmo 2 mostra esse mecanismo.

Algoritmo 2: Algoritmo de difusão de conteúdo

```
Entrada: Número de cópias de conteúdo = M
início
    para Cada contato faça
        se O outro nó possui um valor de gradiente maior ou igual então
            se Número de cópias de conteúdo > 1 então
                Compartilhe metade das cópias com o nó encontrado;
            fim
            senão
                Transfira o conteúdo;
            fim
        fim
    fim
fim
```

3.2.2 Swarm-based Intelligent Routing (SIR)

Em (NGUYEN et al., 2011) é proposto outro protocolo para CCDTN chamado de SIR. Esse protocolo utiliza as métricas atraso e número de saltos para estabelecer o gradiente para decisão de roteamento. O atraso consiste na soma de tempo entre os contatos dos links que constituem o caminho entre dois nós. O número de saltos é a quantidade de links que constituem o caminho entre dois nós. O roteamento do SIR consiste das fases: difusão de interesses e difusão de conteúdos.

Na difusão de interesses, como acontece no STIR, o SIR utiliza a abordagem do *Spray-And-Wait* Binário para espalhar um número limitado de *I-Packets* pela rede. Nessa fase os nós *Relays* (nós intermediário que não são nem produtores nem o usuário do conteúdo) fazem atualização da função de utilidade utilizando os seguintes comportamentos:

- Se o nó *Relay* encontra o usuário do conteúdo, então ele reinicia o valor do atraso para 0 e atualiza o valor do número de saltos para 1.
- Se dois nós *Relays* se encontram, o nó com maior valor de atraso recebe o valor do outro nó. O nó que tiver o maior valor de atraso incrementa o número de saltos em 1.

- Se dois nós *Relays* se encontram e possuem o mesmo atraso, então eles mantém seus valores de utilidades inalterados.

Na difusão de conteúdos, o SIR conduz o *D-Packet* pelo menor caminho até o usuário do conteúdo. A abordagem do *Spray-And-Wait* Binário também é utilizada para limitar o número de cópias. O mecanismo de encaminhamento consiste em sempre selecionar o nó *Relay* com menor atraso (o que indica que estará mais próximo em tempo do usuário do conteúdo). Se os atrasos forem iguais, então o conteúdo é encaminhado para o nó *Relay* com menor número de saltos (mais próximo do usuário do conteúdo em espaço).

3.2.3 Disruption-Tolerant Information-Centric Ad-Hoc Network

Em (YU et al., 2014) é proposto a arquitetura *Disruption-Tolerant Information-Centric Ad-Hoc Network* (DT-ICAN). O DT-ICAN engloba ambas as famílias de redes P2P de disseminação de conteúdos, em que os interesses são propagados de maneira epidêmica. Tendo também características da família de redes baseadas em conteúdo em que o dado em cache é identificado singularmente em blocos. Os interesses são difundidos periodicamente em *broadcast* por meio de mensagens de controle. Com a finalidade de diminuir o consumo da banda, as mensagens de controle de interesse foram divididas em duas: *Node-Interest* que indica quais interesses de dados o nó quer e *Request* que representa o interesse de dados que estão sendo pedidos pelos nós vizinhos, esses pedidos ficam armazenados no *Request Store*. Além disso, existe mensagens de controle de *cache summary* para auxiliar o estabelecimento de prioridades e garantir a compactação de pedidos. Elas também são transmitidas periodicamente.

Uma *Node-Interest* sempre é propagada com prioridade, assim ela sempre será enviada primeiro. Uma *Request* também pode ser ativada quando um novo contato acontece. Nesse caso, é realizado imediatamente um *broadcast* do *Request* atual. Quando uma transmissão de dado é ativada por um *Request* a ordem com que os pedaços de conteúdos disponíveis serão enviados para o nó requisitante será feita de maneira aleatória. Para cada pedaço de dado, o nó que está enviando primeiramente envia um *Request-To-Send-Block* (RTSB) carregando o nome do pedaço. Quando o nó alvo recebe o RTSB, ele então devolve um RTSB-Reply indicando se ele aceita ou não. O pedaço só será enviado no caso de aceite. Esse procedimento é utilizado para eliminar transmissões redundantes.

Em nosso trabalho propomos a arquitetura VDTCON que diferente da arquitetura DT-ICAN, utiliza protocolos de roteamento *unicast* de DTN.

4 PROPOSTA - ARQUITETURA VDTCON

Este trabalho tem como proposta a arquitetura VDTCON que possui o objetivo de fornecer uma rede tolerante a atrasos orientada a conteúdo para aplicações de VANETs. A arquitetura possui as seguintes características:

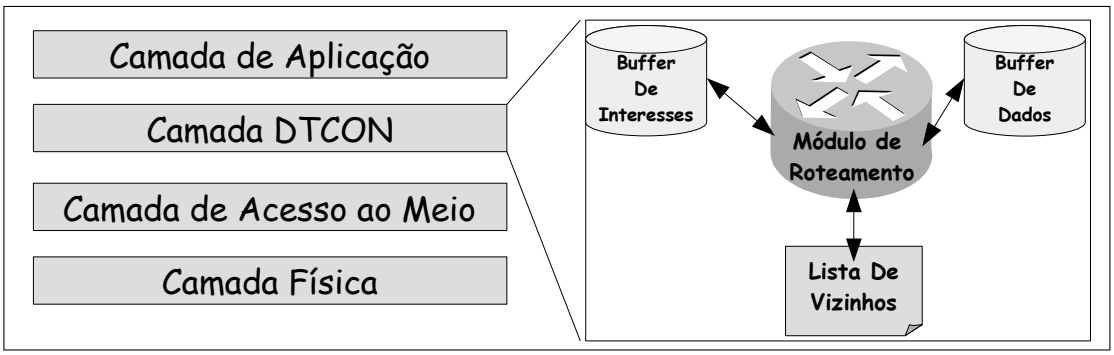
- **Utiliza o paradigma de DTN armazena-carrega-e-encaminha:** Com isso, a arquitetura VDTCON é capaz de lidar com as constantes quebras de conexão que ocorrem em uma VANET. Além disso, é capaz de manter a rede funcionando, mesmo que ela esteja esparsa.
- **Utiliza a nomeação hierárquica de NDN:** Como foi visto anteriormente na seção de VANETs, com uso de endereçamento por nomes, retira-se a necessidade do uso de endereçamento IP. Dessa forma, a comunicação é facilitada, pois quando um determinado nó solicita um conteúdo ele não irá procurar por um IP específico vinculado a um nó, mas irá buscar o conteúdo por um nome que pode estar em qualquer nó na rede.
- **Utiliza I-Packets e D-Packets para a comunicação:** Da mesma forma que ocorre na proposta de NDN, um determinado conteúdo é dividido em vários pedaços.
- **Possui quatro módulos em sua camada DTCON:** Buffer de Interesses, Buffer de Dados, Lista de Vizinhos e Módulo de Roteamento.

Na seção seguinte essas características são detalhadas.

4.1 ESTRUTURAS DO VDTCON

Na Figura 22, pode ser observada uma pilha de camadas genérica para DTCON.

Figura 22 – Pilha de camadas genérica para DTCON.



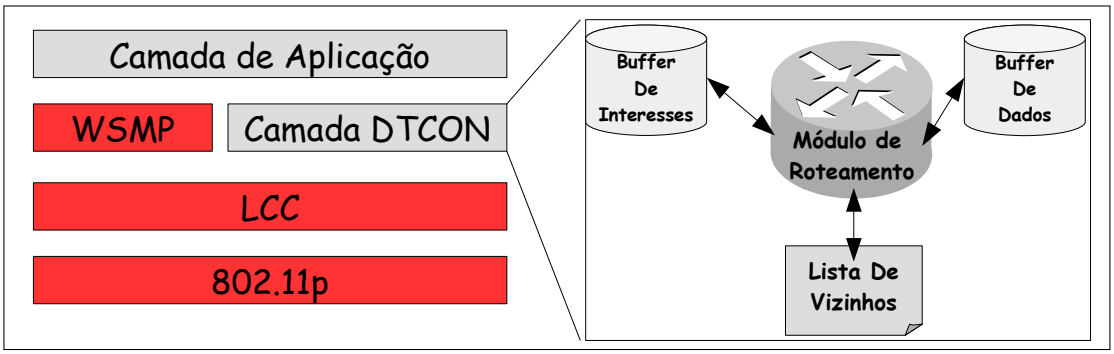
Fonte: Elaborado pelo autor

Na pilha de camadas, pode-se observar que são mantidas as camadas de aplicação, de acesso ao meio e a camada física. Todas as três possuem os papéis semelhantes de quando elas

estão no TCP/IP. Na camada de aplicação o nó consumidor gera os *I-Packets*. Cada aplicação consumidora que estiver sendo executada em um *host* possui um número de processo, através desse valor a camada DTCON sabe para qual aplicação deve entregar o *D-Packet*, quando um determinado *I-Packet* for satisfeito em um nó consumidor. No caso do nó produtor, ele possui uma única aplicação que recebe o *I-Packet* da camada DTCON. Através do nome que vem nesse *I-Packet* a aplicação do nó produtor sabe com qual *D-Packet* ele deve responder. Em relação a camada de acesso ao meio, ela sempre entrega para a camada DTCON qualquer pacote que chegue.

Na Figura 23, podemos ver a pilha quando são utilizadas as camadas de VANETs da pilha WAVE (EICHLER, 2007). Pode ser observado que semelhante ao que é feito em VCCN são removidas as camadas de transporte e de rede, mas no caso da arquitetura proposta, elas são substituídas pela camada DTCON.

Figura 23 – Pilha de camadas para VDTCON.



Fonte: Elaborado pelo autor

A camada DTCON é composta dos seguintes módulos:

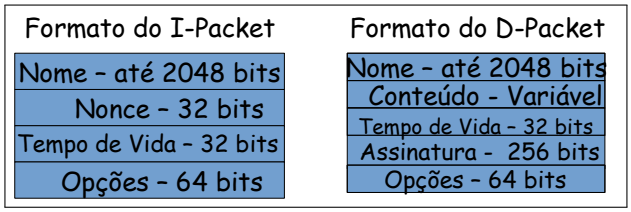
- **Buffer de Interesses:** Sua função é armazenar os *I-Packets* (pacotes de interesse) que foram recebidos ou gerados por um nó. No caso do *I-Packet* que está armazenado no Buffer de Interesses do nó que o gerou, o *I-Packet* possui uma entrada com o número que identifica a aplicação que o criou. Dessa forma, quando o *D-Packet* correspondente chegar, a camada DTCON sabe para qual aplicação deve entregar o *D-Packet*. Quando um determinado nó gera um *I-Packet* e quer obter o conteúdo correspondente, primeiramente, ele precisa estar conectado com algum nó vizinho. Para saber quais vizinhos ele possui, ele checa sua Lista de Vizinhos. Então, o protocolo de roteamento implementado no Módulo de Roteamento decide qual vizinho irá receber o *I-Packet*. Além disso, é definido se o *I-Packet* vai ser copiado ou encaminhado para o vizinho escolhido. Outro ponto importante

é que quando o *D-Packet* (pacote de dado) correspondente ao *I-Packet* que está no Buffer de Interesses chega ao nó, então o *I-Packet* é satisfeito, logo o *I-Packet* é apagado do Buffer de Interesses. Isso é semelhante ao que acontece na estrutura PIT da NDN (JACOBSON et al., 2009) que apaga a entrada de um *I-Packet* quando ele é satisfeito.

- **O Buffer de Dados:** Sua função é armazenar os *D-Packets* que foram recebidos ou gerados por um nó. Ele funciona de maneira similar ao *cache* do CS da NDN. Quando um nó gera um *I-Packet*, ele checa se já existe o *D-Packet* correspondente no seu *buffer*. Nesse caso, o *I-Packet* é automaticamente satisfeito. Quando o *D-Packet* é recebido em um nó, ele fica armazenado para ser novamente encaminhado ou copiado em um contato futuro. Se o *D-Packet* satisfez um *I-Packet* no nó solicitante, então o *D-Packet* é entregue a aplicação. A decisão se este *D-Packet* permanece ou não armazenado no *buffer* fica por conta do Módulo de Roteamento. Em ambos os *buffers*, a política de descarte (no caso de lotação) é definida também no Módulo de Roteamento.
- **Lista de Vizinhos:** Armazena os MACs dos vizinhos que permanecem conectados ao nó. Mensagens *Hellos* são trocadas periodicamente para saber a conectividade dos vizinhos, além disso, servem para a troca de informações utilizadas pelo protocolo de roteamento implementado no Módulo de Roteamento. O formato dos pacotes *Hellos* é definido no Módulo de Roteamento. A Lista de Vizinhos é utilizada pelo Módulo de Roteamento. É dessa lista que sai a escolha de qual vizinho irá receber cópia de algum pacote. Observe que essa decisão faz um papel semelhante da FIB da NDN, definindo por qual *face* (nesse caso, o conjunto de faces de um nó será o seu conjunto de vizinhos) o *I-Packet* será copiado ou encaminhado.
- **Módulo de Roteamento:** Nele é implementado o protocolo de roteamento. Nesse módulo, pode ser colocado qualquer protocolo de roteamento DTN, bastando adaptá-lo para a arquitetura. A partir da interação com os outros módulos, as decisões de roteamento devem ser tomadas. O protocolo deve definir um ou mais tipos de pacote *Hello* para troca de informações periódicas entre os nós. Uma vez que um determinado nó pode possuir N vizinhos e P pacotes em seus *buffers*, o protocolo deve decidir em que ordem irá copiar/encaminhar os pacotes e para qual vizinho irá copiar, ou seja, escolher a ordem dos pares $< P_i, N_j >$. Isso será melhor entendido na descrição dos protocolos. Outra decisão que o protocolo de roteamento terá que tomar é a política de descarte dos pacotes quando um novo pacote chega e o *buffer* está cheio.

A Figura 24 mostra o formato dos pacotes da arquitetura, com o *I-Packet* possuindo

Figura 24 – Formato dos pacotes.



Fonte: Elaborado pelo autor

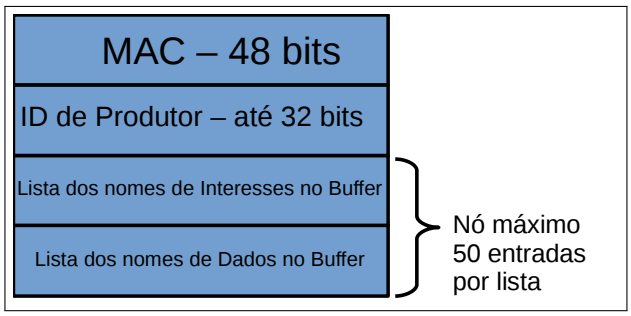
quatro campos e o *D-Packet* cinco. Cada *I-Packet* possui um Nome, para com ele saber como encontrar o seu *D-Packet* correspondente. O formato do nome é o mesmo que utilizado na NDN, ou seja, com nomeação hierárquica. O campo Nonce é um valor inteiro aleatório que cada *I-Packet* recebe. Com esse valor, podemos diferenciar *I-Packets* que possuem o mesmo nome. O Tempo de Vida é um valor inteiro que indica quanto tempo um *I-Packet* pode ficar armazenado no Buffer de Interesses. Se o protocolo de roteamento necessitar de alguma informação adicional, ela deverá ser colocada no campo Opções. O *D-Packet* também possui o campo Nome para sua idenficação correspondente com algum *I-Packet*. O campo Conteúdo é um pedaço do dado solicitado pelo consumidor. O Tempo de Vida indica quanto tempo o *D-Packet* poderá ficar armazenado no Buffer de Dados. O campo Assinatura contém dados para aspectos de segurança. Finalmente, o campo Opções poderá conter informações adicionais para o protocolo de roteamento.

As decisões tomadas pelo Módulo de Roteamento são definidas pelo protocolo de roteamento. Para realizar experimentos com a arquitetura, foram adaptados os protocolos *Epidêmico* e *Spray-And-Wait*, no qual foram nomeados para VDTCON-Epidemic e VDTCON-Spray. Além disso, foi desenvolvido o protocolo VDTCON-QLR que utiliza Q-Learning para tomar decisões de roteamento. Todos os três são abordados nas seções seguintes.

4.2 PROTOCOLO VDTCON-EPIDEMIC

O protocolo VDTCON-Epidemic é adaptado do protocolo de roteamento DTN clássico, o Epidêmico (VAHDAT et al., 2000). A Figura 25 mostra o formato do pacote *Hello*. O campo MAC serve para que um vizinho possa ser identificado na Lista de Vizinhos. o ID de produtor é uma parte associada ao nome do conteúdo. Dessa forma, quando um determinado nó K possui um vizinho que é produtor de algum interesse de K, então K imediatamente solicita

Figura 25 – Pacote Hello do VDTCON-Epidemic.

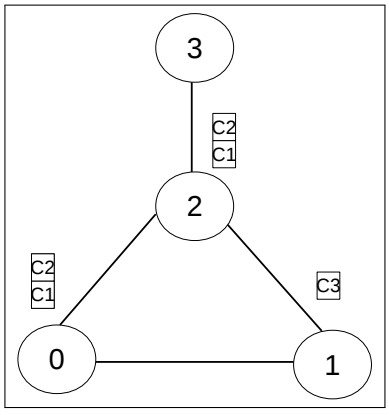


Fonte: Elaborado pelo autor

o *D-Packet* correspondente. As listas de dados e interesses servem para que um nó identifique quais pacotes seus vizinhos já possuem, evitando assim copiá-los desnecessariamente. No campo Opções do *I-Packet* e do *D-Packet* é adicionado o campo NumCop (Número de Cópias com 16 bits) que irá indicar quantas vezes o pacote foi replicado.

Para entender o funcionamento do protocolo, considere a rede mostrada na Figura 26. Nela, os nós 0 e 2 já possuem em seus Buffers de Dados $C1$ e $C2$ que são produzidos pelo

Figura 26 – Rede utilizada no exemplo.

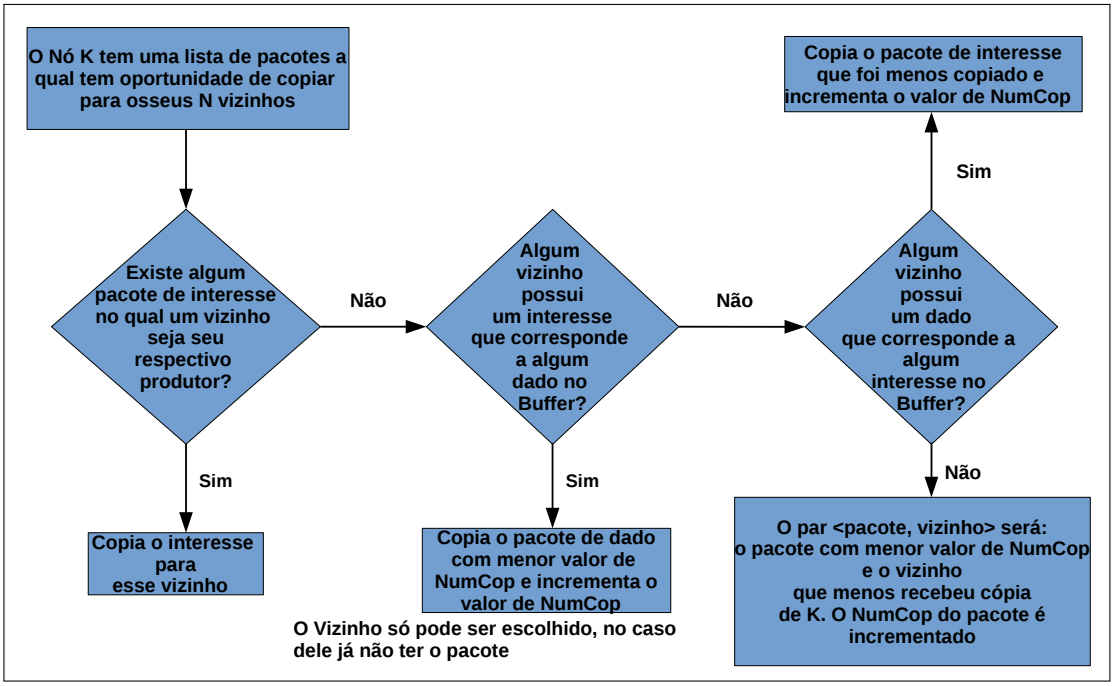


Fonte: Elaborado pelo autor

nó 3. O nó 1 possui $C3$ que também é produzido pelo nó 3. Supondo que todos os vizinhos já estão na Lista de Vizinhos de todos os nós e que o nó 0 já sabe quais pacotes estão em 1 e 2 e que essas informações foram obtidas na troca de mensagens *Hellos*, imagine que o nó 0 gera o *I-Packet* $I3$, cujo o *D-Packet* é o $C3$. Enquanto tiver conectividade, o nó 0 irá copiar os pacotes do seus *buffers*, mas como o ambiente é uma VANET, então podem ocorrer rápidas quebras de conexão. Nesse caso, o protocolo precisa definir qual pacote será copiado primeiro e qual vizinho irá receber primeiro a cópia.

A decisão é tomada conforme o fluxograma da Figura 27. Inicialmente, o nó K tem

Figura 27 – Fluxograma do protocolo VDTCON-Epidemic.



Fonte: Elaborado pelo autor

uma lista de pacotes para enviar aos seus *N* vizinhos. Então ele checa se existe algum produtor para algum de seus *I-Packets*. Caso sim, o *I-Packet* é escolhido imediatamente para seu produtor correspondente. Caso não, então o nó *K* checa se algum vizinho possui algum *I-Packet* que possa ser satisfeito por algum dos seus *D-Packets*. Caso sim, para cada par *<D-Packet, Vizinho>* é escolhido o par cujo *D-Packet* foi menos copiado. Caso não, o nó *K* checa se algum vizinho possui algum *D-Packet* que corresponda à algum de seus *I-Packets*. Caso sim, da mesma forma que anteriormente, para cada par *<I-Packet, Vizinho>* é escolhido o par cujo seu *I-Packet* foi menos copiado.

No caso de nenhuma das condições descritas no fluxograma da Figura 27 serem atendidas, então o par escolhido *<Pacote, Vizinho>* será o *Pacote* que foi menos copiado por *K* e o *Vizinho* que menos recebeu cópia de *K*. Dessa forma, a prioridade fica com os pacotes que foram menos copiados e as cópias ficam mais espalhadas entre os vizinhos.

Podemos observar duas restrições em relação aos nós da Lista de Vizinhos: a primeira é que para um determinado pacote (seja *D-Packet* ou *I-Packet*), um vizinho só poderá ser escolhido no caso dele não possuir um pacote equivalente. Assim, é evitada a transmissão desnecessária de cópias. Na segunda restrição, temos a variável *nonce*. Ela contém um valor aleatório que é colocado no *I-Packet* no momento em que ele é gerado, da mesma forma que ocorre no NDN original (JACOBSON et al., 2009). É importante observar que podem haver

dois *I-Packets* com o mesmo nome, mas com valores de *nonce* diferentes (por exemplo, dois nós geraram o mesmo *I-Packet*). Assim, é garantido que o nó *K* só copie *I-Packets* repetidos para um mesmo vizinho no caso dos *I-Packets* terem valores *nonce* diferentes.

Voltando ao exemplo da Figura 26, quando o nó 0 executa o algoritmo, a decisão tomada por ele será copiar primeiro o *I*3 para o nó 1. Quando o nó 1 recebe o *I*3, ele imediatamente responde com *C*3 para o nó 0. A Figura 28 mostra um exemplo de uma rede executando o fluxograma da Figura 27.

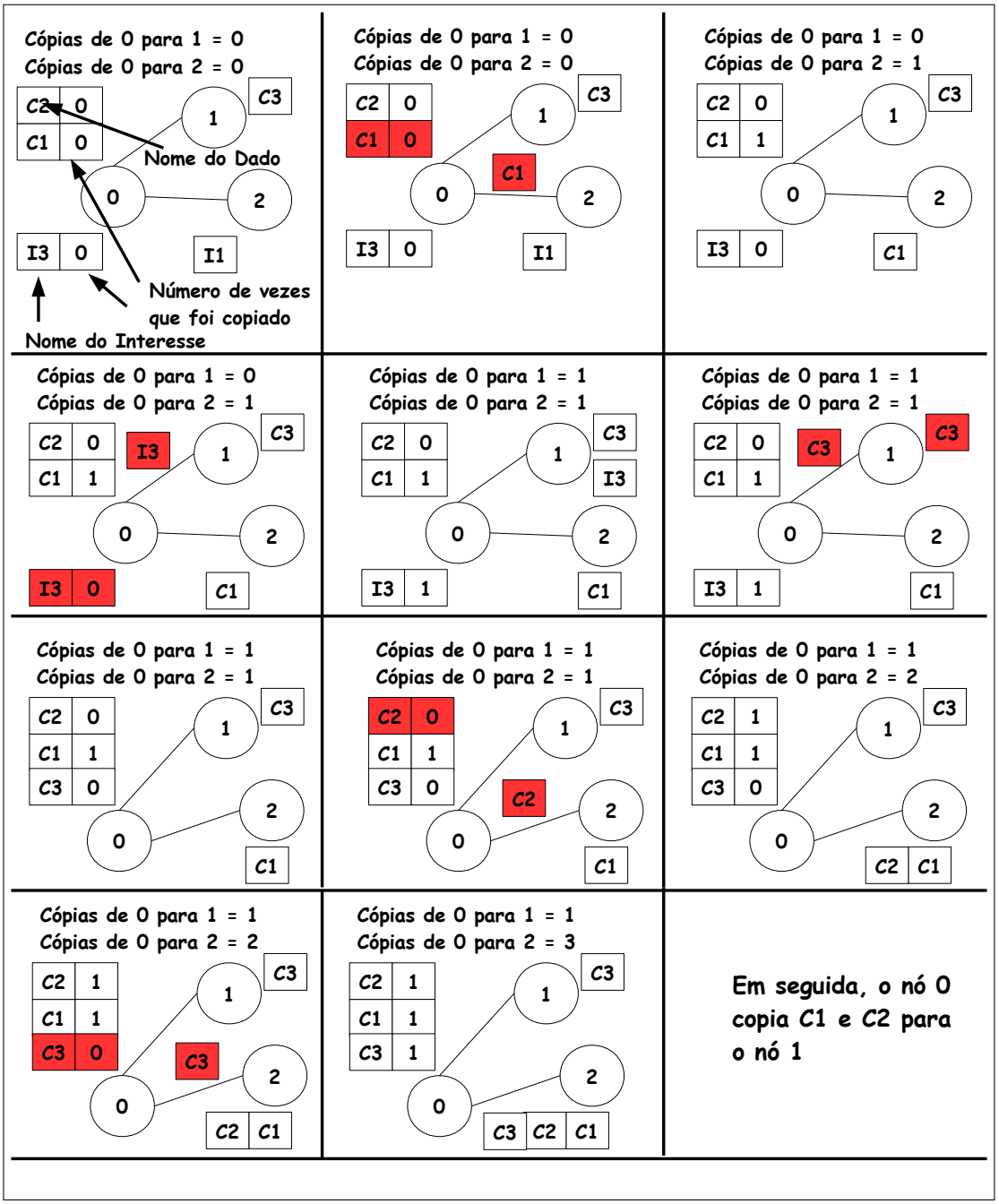
4.3 PROTOCOLO VDTCON-SPRAY

O protocolo VDTCON-Spray é adaptado do protocolo de roteamento DTN clássico *Spray-And-Wait* (SPYROPOULOS et al., 2005). O formato do pacote *Hello* é o mesmo do VDTCON-Epidemic (Figura 25). No campo Opções do *I-Packet* e do *D-Packet*, é adicionado, além do NumCop (16 bits), o campo NumRep (Número de Replicações com 16 bits) que é inicializado com um valor constante *L*. Esse campo irá indicar quantas replicações ainda restam para o pacote. A Figura 29 mostra o fluxograma do VDTCON-Spray. Como pode ser visto, ele é semelhante ao fluxograma do VDTCON-Epidemic, mudando apenas o quadrado em vermelho. A diferença é que o número de cópias é limitada pelo valor de *L*. Cada pacote possui um valor NumRep que é inicializado com *L*. Todas as vezes que um pacote é copiado, o valor de NumRep é divido por 2. Quando o valor de NumRep chega em 1, o pacote só será copiado quando algum vizinho possuir o pacote correspondente. Outro detalhe é que o valor de NumRep sempre é reinicializado para *L* quando chega em outro nó. Dessa forma, cada nó replica no máximo $\log_2(L)$ vezes cada pacote. Assim, o protocolo faz o controle de replicação, amenizando o *overhead* na rede.

4.4 PROTOCOLO VDTCON-QLR

O VDTCON-QLR utiliza QLearning para auxiliar na decisão da escolha do par <Pacote, Vizinho>. Em semelhança ao VDTCON-Epidemic, o VDTCON-QLR não limita o número de cópias que poderá ser feito de um pacote. Todavia, se para um nó *n* qualquer nenhum vizinho possuir o valor da função de utilidade *Q* maior ou igual ao de *n*, então não será realizada a replicação.

Figura 28 – Executando o fluxograma do VDTCON-Epidemic, a ordem das figuras vai da esquerda para direita.



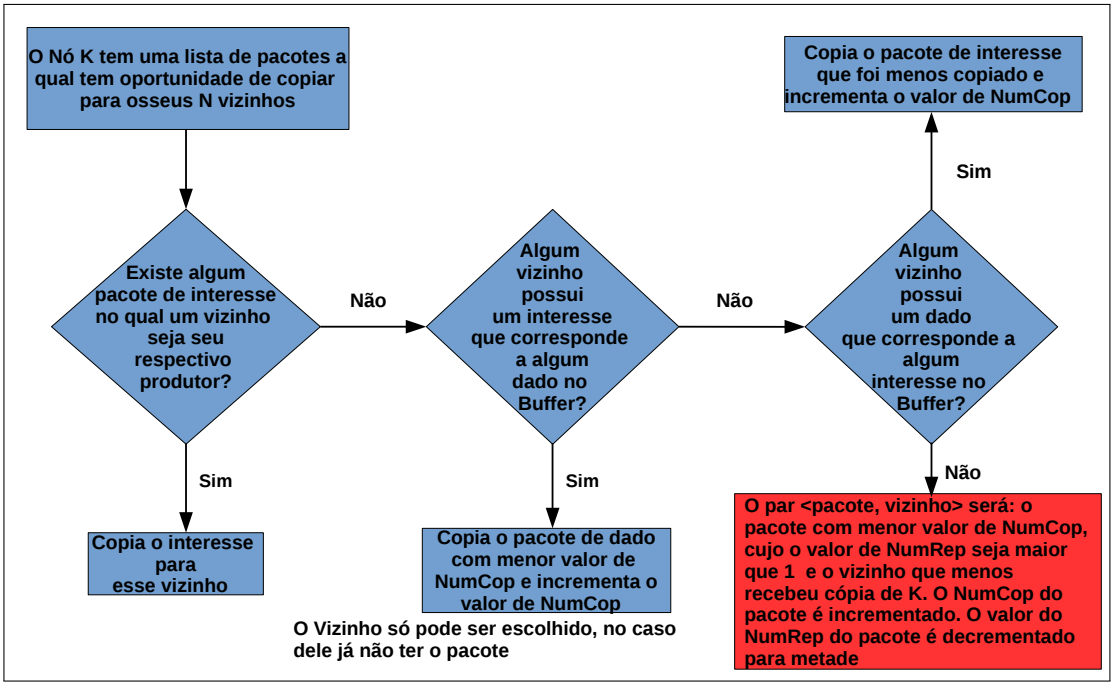
Fonte: Elaborado pelo autor

4.4.1 Modelo de Aprendizagem por Reforço

Para o protocolo, é preciso definir um modelo para a aplicação da aprendizagem por reforço. Nessa proposta o modelo é da seguinte forma:

- Cada nó *n* representa um estado *N*.
- Cada vizinho *v* de um nó *n* qualquer representa uma possível ação do agente, ou seja, a

Figura 29 – Fluxograma do protocolo VDTCON-Spray.



Fonte: Elaborado pelo autor

ação realizada é copiar um pacote para *v* cujo destino final é um nó identificado por *ID*.

- Cada *ID* que identifica o nó destinatário representa um agente que está aprendendo.

Dependendo do nó em que o pacote esteja, o valor de reforço pode variar, além disso, o agente vai sempre de um nó para outro, por isso modelamos cada nó *n* como sendo um possível estado do agente. A ação escolhida pelo agente também reflete no valor do reforço. Por exemplo, o valor utilizado no cálculo do reforço depende do valor do ToD para o nó destino da ação escolhida. Dessa forma, no modelo proposto, o conjunto de ações é formado pela possibilidade de cópias que podem ser feitas para cada vizinho que o agente tem quando está no estado *n*. Como numa rede pode existir mais de um destinatário, os valores de aprendizado mudam para cada *ID* de nó destino, ou seja, para cada agente. O *ID* de nó destino para o *I-Packet* é uma parte associada ao nome do produtor. Nesse protocolo, cada *I-Packet* deve carregar o MAC do dispositivo e a geolocalização do consumidor que o gerou para que quando o *D-Packet* correspondente for gerado, o seu ID de destino seja configurado com este MAC e também com a geolocalização do nó consumidor alvo.

4.4.2 Cálculo do Reforço

O cálculo do reforço é auxiliado por dois valores: pelo ToD e pelo valor de contato VC que indica o quanto um nó teve de contato com o nó destino. O VC recebe o valor de contato máximo $CONMAX$ todas as vezes que um nó entrar em contato com um nó destinatário. No caso de dois nós intermediários i e j entrarem em contato, o valor de VC de i, receberá metade do valor do VC de j, caso VC de j seja duas vezes maior que o VC de i e vice-versa.

Nessa proposta, o reforço é calculado sempre que é necessário atualizar o valor da função Q. O cálculo do reforço em um nó i qualquer é feito conforme a Equação 4.1:

$$R(i,ID) = Val(ToD(i,Pos(ID))) + \frac{VC(i,ID)}{CONMAX} \tag{4.1}$$

O valor $Val(ToD(i,Pos(ID)))$ fica entre 0 e 1. Por isso, o valor $VC(i,ID)$ é divido por $CONMAX$ para que $VC(i,ID)$ fique no mesmo intervalo de $Val(ToD(i,Pos(ID)))$. Após não haver mais contato entre os nós, o valor de VC para cada nó destinatário sofre envelhecimento e assim o seu valor vai diminuindo. A Equação 4.2 mostra como o envelhecimento é feito:

$$VC(i,ID) = VC(i,ID)\mu^T \tag{4.2}$$

Onde $\mu = [0,1]$ e T é o tempo decorrido desde o último contato.

4.4.3 Cálculo da Função Q

Assim que o protocolo inicia a execução em um nó n, o valor de Q é inicializado com 0. Após isso, a cada S segundos, o valor de Q é atualizado. Essa atualização é feita conforme a Equação 4.3:

$$Q_n(n,ID) = Q_n(n,ID) + \alpha(R(n,ID) + \gamma Max(Q_n(v,ID)) - Q_n(n,ID)) \tag{4.3}$$

Onde ID identifica o nó destinatário, v é um vizinho de n, γ é o fator de desconto temporal, α é a taxa de aprendizado e $Max(Q_n(v,ID))$ é o valor máximo de Q dos vizinhos para ID no nó n. Os valores Q dos vizinhos são obtidos e atualizados via mensagem *Hello*.

O valor da função Q sofre o mesmo envelhecimento da variável VC conforme a Equação 4.4.

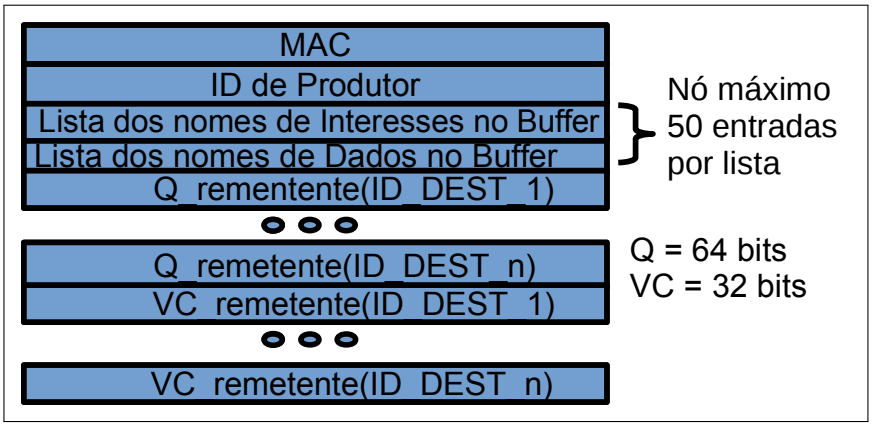
$$Q_n(n,ID) = Q_n(n,ID)\mu^T \tag{4.4}$$

Esse envelhecimento ocorre em cada atualização da função Q. O valor da função Q sempre é mantido no máximo em qualquer nó consumidor no qual o pacote a ser enviado é de dado. Isso é feito para aumentar a quantidade de *D-Packets* no *buffer* dos nós consumidores, facilitando que os *I-Packets* correspondentes sejam automaticamente satisfeitos.

4.4.4 Pacote Hello

A Figura 30 mostra o formato do pacote *Hello*. Basicamente, cada nó envia os seus

Figura 30 – Pacote Hello do VDTCON-QLR.



Fonte: Elaborado pelo autor

valores Q e VC referentes a cada nó destino. Dessa forma, os nós da rede podem manter uma tabela com valores Q e VC de cada vizinho e assim, decidir para quais vizinhos devem ser enviados os pacotes.

4.4.5 Exemplo dos Cálculos Feito pelo Protocolo

À medida que um nó n VDTCON-QLR vai recebendo mensagens *Hello* dos seus vizinhos, ele vai montando e atualizando sua tabela de valores Q. O Quadro 1 mostra a tabela de valores Q no nó n com dois vizinhos e dois IDs de destino. Observe que o próprio nó n está na sua tabela de vizinhos, ou seja, a tabela contém os valores de Q e VC do próprio nó n. Esses valores são comparados com os dos vizinhos na hora de decisão de roteamento.

Suponha agora que houve uma atualização da função Q em um nó n qualquer e que o tempo de atualização a função Q é 0.2 segundos, $\alpha = 0.7$, $\gamma = 0.8$, $\mu = 0.98$, $Val(ToD(n, Pos(ID1)))$ = 0.75, $Val(ToD(n, Pos(ID2)))$ = 0.95 e $CONMAX$ = 10. O nó n atualiza seus valores Q conforme as contas abaixo:

Quadro 1 – Tabela montada nó n.

Vizinho	ID de Destino	Valor Q	VC
n	ID1	5.2	8.0
n	ID2	2.0	6.5
$v1$	ID1	4.2	9.2
$v1$	ID2	3.0	7.5
$v2$	ID1	1.4	9.2
$v2$	ID2	3.4	7.5

Fonte: Elaborado pelo autor

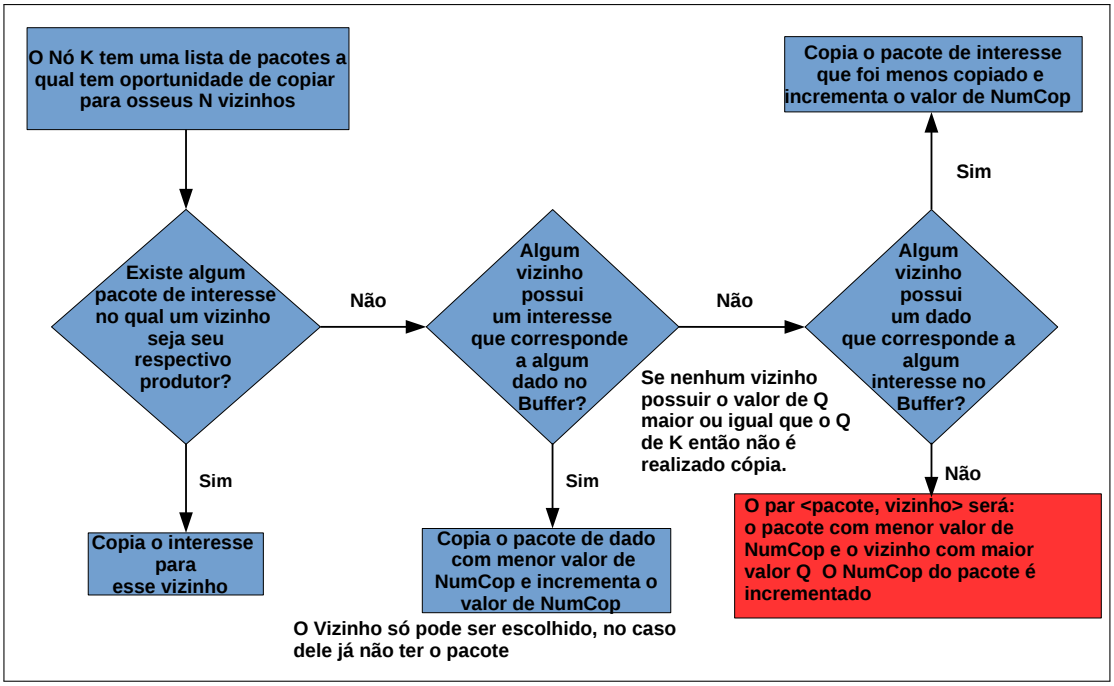
$$R(n,ID1) = Val(ToD(n,Pos(ID1))) + \frac{VC(n,ID1)}{CONMAX}$$
$$R(n,ID1) = 0.75 + \frac{8.0}{10.0} = \mathbf{1.55}$$
$$Q_n(n,ID1) = Q_n(n,ID1) + \alpha(R(n,ID1) + \gamma Max(Q_n(v,ID1)) - Q_n(n,ID1))$$
$$Q_n(n,ID1) = 5.2 + 0.7 * (1.55 + 0.8 * 4.2 - 5.2) = \mathbf{4.99}$$
$$Q_n(n,ID1) = Q_n(n,ID1)\mu^T$$
$$Q_n(n,ID1) = 4.99 * 0.98^{0.2} = \mathbf{4.79}$$
$$R(i,ID2) = Val(ToD(n,Pos(ID2))) + \frac{VC(n,ID2)}{CONMAX}$$
$$R(n,ID2) = 0.95 + \frac{6.5}{10.0} = \mathbf{1.6}$$
$$Q_n(n,ID2) = Q_n(n,ID2) + \alpha(R(n,ID2) + \gamma Max(Q_n(v,ID2)) - Q_n(n,ID2))$$
$$Q_n(n,ID2) = 2.0 + 0.7 * (1.6 + 0.8 * 3.4 - 2.0) = \mathbf{3.62}$$
$$Q_n(n,ID2) = Q_n(n,ID1)\mu^T$$
$$Q_n(n,ID2) = 3.62 * 0.98^{0.2} = \mathbf{3.47}$$

Os outros nós também atualizam seus valores e trocam mensagens *Hello* para que possam manter os valores Q dos vizinhos atualizados.

4.4.6 Algoritmo de Decisão de Roteamento

O algoritmo de decisão é mostrada no fluxograma da Figura 31. Ela é semelhante ao do VDTCON-Epidemic (Figura 27) e ao do VDTCON-Spray (Figura 29), mudando apenas o quadrado final do fluxograma. Quando o fluxograma chega nesse quadrado em um nó n qualquer, é escolhido o pacote que foi menos copiado. Então é selecionado para receber a cópia o vizinho que possui o maior valor Q para esse pacote. Caso nenhum vizinho possua um valor Q maior ou igual que ao de Q do nó n, então o pacote não é copiado. O Algoritmo 3 mostra como é tomada essa decisão.

Figura 31 – Fluxograma do protocolo VDTCON-QLR.



Fonte: Elaborado pelo autor

Algoritmo 3: Decisão de roteamento tomado por um nó n

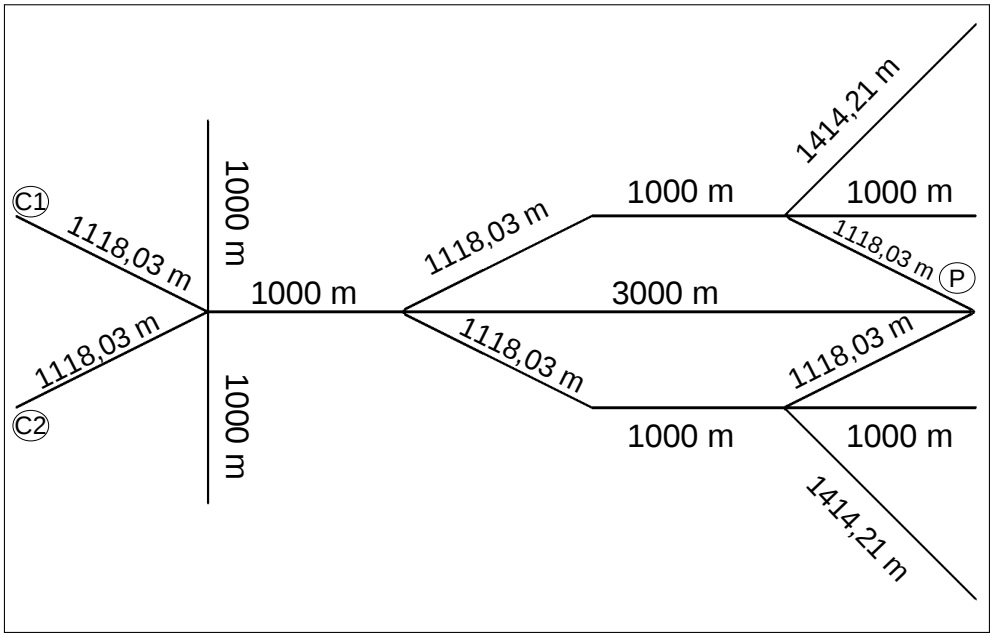
```
begin
    p = pacote_menos_copiado(Buffer de Interesses, Buffer de Dados);
    ID_DEST = GetIDByPacket(p);
    Q_max = 0;
    vCopy = null;
    for cada vizinho v do
        if Q_n(nei, ID_DEST) > Q_max && v não possui o pacote p then
            Q_max = Q_n(v, ID_DEST);
            vCopy = v;
        end
    end
    if Q_max >= Q_n(n, ID_DEST) then
        n copia o pacote p para vCopy;
    end
    else
        n não realiza cópia;
    end
end
```

5 EXPERIMENTOS E RESULTADOS

5.1 DESCRIÇÃO DO CENÁRIO

O mapa do cenário utilizado para os experimentos é mostrado na Figura 32. Ele foi

Figura 32 – Cenário utilizado na simulação.



Fonte: Elaborado pelo autor, gerado no SUMO.

elaborado utilizando o Simulation of Urban MObility (SUMO). Nele cada pista possui 4 faixas (duas de ida e duas de volta). O cenário consiste de dois consumidores $C1$ e $C2$ que possuem interesses que podem ser obtidos no produtor P. Durante 200 segundos, $C1$ gera pacotes de interesses. Após passar esse tempo, $C2$ começa a gerar os mesmos interesses, também durante 200 segundos.

A aplicação desse cenário é semelhante ao KioskNet (GUO et al., 2011). Os nós consumidores recebem os interesses solicitados da região e então os propagam utilizando os veículos como carregadora de dados. Quando o produtor recebe um interesse, ele manda o pacote de dados utilizando também os veículos. Os tipos de veículos foram categorizados em: ônibus, trem, táxi e veículos comuns. Todos eles com padrões de mobilidades conforme descrito em (CHENG et al., 2008). Outras informações do cenário podem ser vistas na Tabela 2. Os valores das constantes foram escolhidas empiricamente.

Tabela 2 – Configuração dos experimentos

Parâmetro	Configuração
Raio de transmissão	300 m
Memória	Dispositivos com memória abundante
Velocidade máxima dos nós	27.77 m/s
Número de nós	2 trens, 14 ônibus, 6 táxis e 20 carros
Modelo de propagação	Nakagami
Modelo de mobilidade	carFollowing-Krauss (Padrão SUMO)
Tamanho do D-Packet	1000 bytes
Tamanho dos Nomes	64 bits
Tempo de simulação	2700 segundos
Tempo de Warm Up	2000 segundos
Tempo de vida do I-Packet	500 segundos
Tempo de vida do D-Packet	700 segundos
Quantidade de simulações por cenário	33
Intervalo de confiança	95%
$\alpha - QL$, $\mu - QL$, $\gamma - QL$	0.7, 0.98, 0.8
$\alpha - STIR$, $\beta - STIR$	0.8, 0.98
Camada MAC	802.11p (conforme implementado no NS 3.16)

5.2 MÉTRICAS

Neste trabalho são utilizadas três métricas: taxa de interesses satisfeitos (TIS), atraso médio e overhead. Para as métricas, considere *IG* como número de *I-Packets* gerados durante a simulação, *IS* como número de *I-Packets* satisfeitos no nós consumidores, *IC* como número de *I-Packets* copiados, *DC* como número de *D-Packets* copiados e IT_i como tempo levado para que o *I-Packet i* seja satisfeito no consumidor.

A métrica TIS indica qual a porcentagem de *I-Packets* satisfeitos nos nós consumidores. A TIS é calculada conforme a Equação 5.1:

$$TIS = \ (\frac{IS}{IG}.100)\% \tag{5.1}$$

O *overhead* é calculado segundo a Equação 5.2. Ela calcula o custo gerado pelos *I-Packets* e *D-Packets* copiados pela rede.

$$Overhead \ = \ IC + DC \tag{5.2}$$

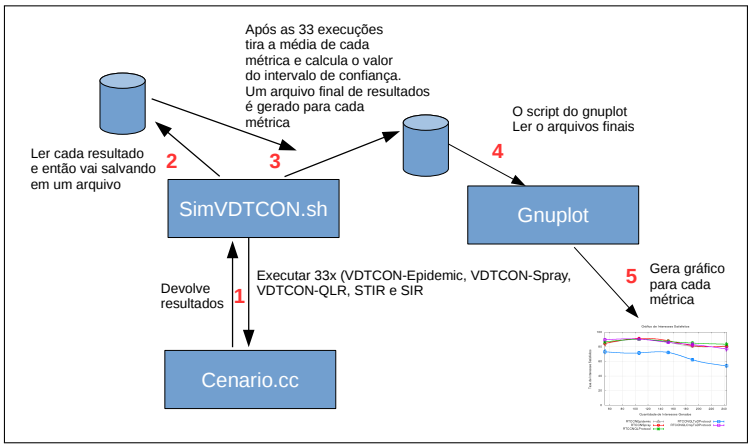
O atraso médio calcula quanto tempo em média leva para que um *I-Packet* seja satisfeito no nó consumidor. Nesse trabalho, o atraso médio é calculado conforme a Equação 5.3.

$$Atraso\ Médio = \frac{\sum_{i=1}^{IS} IT_i}{IS} \tag{5.3}$$

5.3 SIMULAÇÃO

Para a simulação, foi criado um script em *shell* que executa o cenário no NS3 33 vezes para cada protocolo. Dessa forma, são gerados arquivos com os resultados de cada métrica. Esses arquivos são passados para um outro script do *gnuplot* que gera então os gráficos de resultados. Essa abordagem é mostrada na Figura 33.

Figura 33 – Arquitetura de como a simulação foi feita.



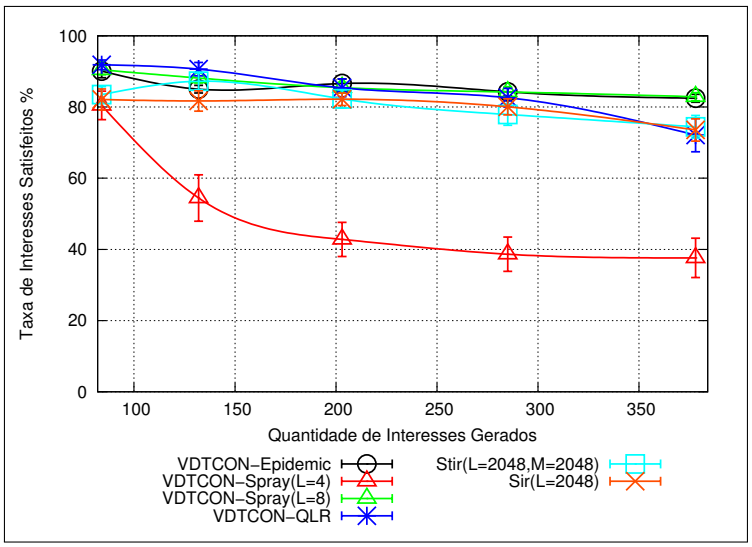
Fonte: Elaborado pelo autor.

5.4 ANÁLISE DE RESULTADOS

Analisamos primeiramente a taxa de interesses satisfeitos. É importante que o protocolo consiga satisfazer o máximo possível de interesses, maximizando a possibilidade de que os dados cheguem ao nó consumidor que os solicitou.

Na Figura 34 pode ser visto o comportamento da taxa de interesses satisfeitos nos protocolos. Percebe-se que todos mantiveram aproximadamente uma taxa entre 72% e 90%, com exceção do VDTCON-Spray(4) que chegou a atingir uma taxa inferior à 40%. Essa taxa baixa ocorreu devido ao o número baixo de cópias que VDTCON-Spray(4) fez, diferentemente do VDTCON-Spray(8) que espalhou bem mais cópias e alcançou, por isso, taxas altas. Como existe bastante espaço na memória dos veículos, o VDTCON-Epidemic conseguiu atingir taxas altas, pois ele tenta espalhar o máximo possível de pacotes, facilitando as entregas. Apesar do VDTCON-Spray(8) espalhar menos cópias que o VDTCON-Epidemic, ele conseguiu taxas no mesmo intervalo. A vantagem do VDTCON-Spray(8) em relação ao VDTCON-Epidemic foi manter o *overhead* mais baixo. Os protocolos SIR, STIR e VDTCON-QLR conseguiram taxas

Figura 34 – Avaliando desempenho da taxa de interesses satisfeitos.



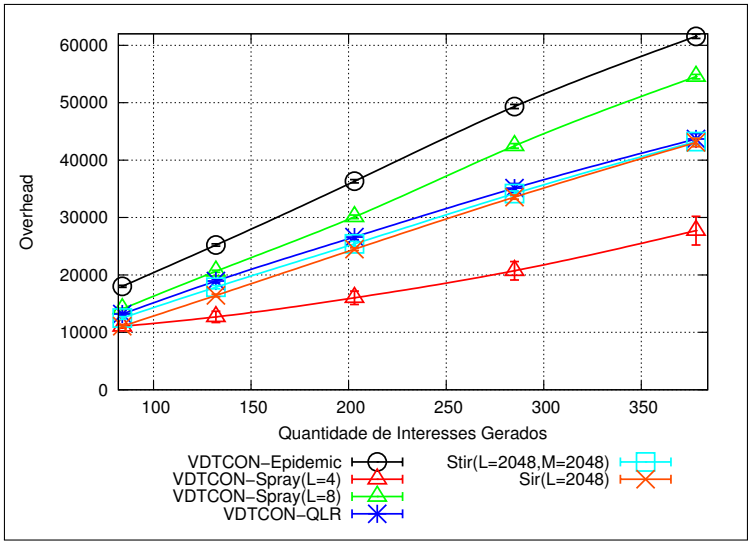
Fonte: Elaborado pelo autor.

próximas das do VDTCON-Epidemic, mas mantendo um controle mais rígido para espalhar cópias dos pacotes, visando atingir *overhead* mais baixo. O SIR e STIR fazem isso estabelecendo o limite de cópias. O VDTCON-QLR faz isso espalhando cópias apenas quando um vizinho possui o valor de função Q maior. Os seguintes pontos cooperam para que os protocolos consigam taxas altas:

- Todos os protocolos realizam cópias dos *I-Packets* e *D-Packet*, ou seja, sempre os pacotes serão espalhados pela rede, pelo menos enquanto o tempo de vida dos pacotes não expirar.
- Como os *D-Packets* são mantidos no *buffer* de nós intermediários, muitos *I-Packets* podem ser satisfeitos mesmo sem chegar no nó produtor.
- Um *I-Packet* pode ser automaticamente satisfeito assim que ele é gerado, bastando que *D-Packet* correspondente já esteja no *buffer*.

O valor do *overhead* diz qual o custo necessário para atingir as taxas da Figura 34. Os valores são mostrados na Figura 35. Como já esperado, o VDTCON-Epidemic alcançou o maior *overhead*, uma vez que ele sempre espalha pacotes sem nenhum controle. O VDTCON-Spray(8) manteve o segundo maior *overhead*. Ele mais é baixo que o do VDTCON-Epidemic devido ao fato do VDTCON-Spray(8) limitar um número de cópias. O VDTCON-Spray(4) obteve o *overhead* mais baixo, porém com a taxa de interesses satisfeitos mais baixa dentre todos. Os protocolos SIR, STIR e VDTCON-QLR mantiveram praticamente o mesmo *overhead*. O *overhead* do STIR e do SIR poderiam ser diminuídos, bastando limitar mais o número de cópias, porém isso causaria uma menor taxa de interesses satisfeitos e um maior atraso médio dos pacotes.

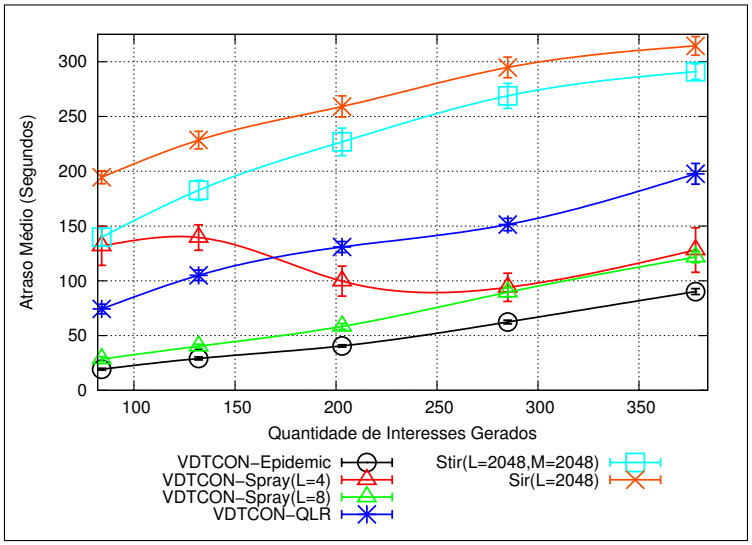
Figura 35 – Avaliando desempenho de overhead.



Fonte: Elaborado pelo autor.

O atraso médio na Figura 36 mostra quanto tempo, em média, levou para um interesse ser satisfeito. Nesse caso, os menores atrasos ficaram com os protocolos que espalham mais pacotes, uma vez que quantos mais pacotes são espalhados, mais rápido eles chegarão ao seu destino. Dessa forma, o VDTCON-Epidemic e VDTCON-Spray(8) atingiram os menores atrasos.

Figura 36 – Avaliando desempenho de atraso médio.



Fonte: Elaborado pelo autor.

A curva do VDTCON-Spray(4) teve um comportamento diferente devido sua variação mais alta na taxa de interesses satisfeitos, uma vez que o atraso médio é calculado em cima dos *I-Packets* que foram satisfeitos. Dentre os protocolos SIR, STIR e VDTCON-QLR, o último ficou com o

menor atraso. A vantagem do VDTCON-QLR é que além de ele usar como parâmetro para a função Q o contato com o destino, ele também utiliza a métrica ToD.

6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho propôs a arquitetura VDTCON baseada em DTN e NDN para fornecer uma rede tolerante a atrasos orientado a conteúdo para aplicações VANETs. Na arquitetura foi feita uma estrutura chamada de módulo de roteamento, onde pode ser adaptado e inserido qualquer protocolo roteamento DTN. Foram adaptados os protocolos Epidêmico (adaptação nomeada de VDTCON-Epidemic) e *Spray-And-Wait* (adaptação nomeada de VDTCON-Spray). Foi desenvolvido o protocolo VDTCON-QLR e implementados os protocolos STIR e SIR. Todos eles sendo colocados no módulo de roteamento da arquitetura VDTCON para que fossem realizados os experimentos.

Nas avaliações dos experimentos, os protocolos VDTCON-Epidemic e VDTCON-Spray alcançaram altas taxas de interesses satisfeitos e atraso médio baixo, porém com *overhead* mais alto que os demais. O VDTCON-QLR desenvolvido ficou com a mesma taxa de interesses satisfeitos e *overhead* dos protocolos SIR e STIR, porém superou-os no atraso médio.

A contribuição desse trabalho é todo o ambiente da arquitetura VDTCON com seus protocolos de roteamento. Nesse ambiente, novos protocolos de roteamento podem ser desenvolvidos e comparados com os já implementados. Novos cenário podem ser elaborados para outros testes. Além disso, pode ser feita uma comparação com outros trabalhos de DTN com redes orientadas a conteúdo, tais como DT-ICAN (YU et al., 2014), mostrando em que situações um ou outro é melhor.

Para trabalhos futuros, consideramos também adaptar para a o módulo de roteamento da arquitetura VDTCON protocolos de roteamento VANET/DTN , tais como o VDTN-ToD (VIEIRA et al., 2013), VADD (ZHAO; CAO, 2008) e GeOpps (LEONTIADIS; MASCOLO, 2007). Assim, avaliar e comparar seus desempenhos quando eles são adaptados para a arquitetura.

REFERÊNCIAS

AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. Wireless sensor networks: a survey. **Computer networks**, Elsevier, v. 38, n. 4, p. 393–422, 2002.

AL-SULTAN, S.; AL-DOORI, M. M.; AL-BAYATTI, A. H.; ZEDAN, H. A comprehensive survey on vehicular ad hoc network. **Journal of Network and Computer Applications**, Elsevier, 2013.

ALVES, R.; CAMPBELL, I.; COUTO, R.; CAMPISTA, M.; MORAES, I.; RUBINSTEIN, M.; COSTA, L.; DUARTE, O.; ABDALLA, M. Redes veiculares: Princípios, aplicações e desafios. **Minicursos do Simpósio Brasileiro de Redes de Computadores, SBRC**, 2009.

AMADEO, M.; CAMPOLO, C.; MOLINARO, A. Crown: Content-centric networking in vehicular ad hoc networks. **Communications Letters, IEEE**, IEEE, v. 16, n. 9, p. 1380–1383, 2012.

BOUK, S. H.; AHMED, S. H.; KIM, D. Vehicular content centric network (vccn): a survey and research challenges. In: ACM. **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. [S.l.], 2015. p. 695–700.

BRITO, G. M. de; VELLOSO, P. B.; MORAES, I. M. Redes orientadas a conteúdo: Um novo paradigma para a internet. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC**, v. 2012, p. 211–264, 2012.

CAESAR, M.; CONDIE, T.; KANNAN, J.; LAKSHMINARAYANAN, K.; STOICA, I. Rofl: routing on flat labels. In: ACM. **ACM SIGCOMM Computer Communication Review**. [S.l.], 2006. v. 36, n. 4, p. 363–374.

CARZANIGA, A.; ROSENBLUM, D. S.; WOLF, A. L. **Content-based addressing and routing: A general model and its application**. [S.l.], 2000.

CHENG, P.-C.; WENG, J.-T.; TUNG, L.-C.; LEE, K. C.; GERLA, M.; HAERRI, J. Geodtn+ nav: A hybrid geographic and dtn routing with navigation assistance in urban vehicular networks. **MobiQuitous/ISVCS**, 2008.

EICHLER, S. U. Performance evaluation of the ieee 802.11 p wave communication standard. In: IEEE. **Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th**. [S.l.], 2007. p. 2199–2203.

FALL, K. A delay-tolerant network architecture for challenged internets. In: ACM. **Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications**. [S.l.], 2003. p. 27–34.

FALL, K.; FARRELL, S. Dtn: an architectural retrospective. **Selected Areas in Communications, IEEE Journal on**, IEEE, v. 26, n. 5, p. 828–836, 2008.

FILHO, J. G. **Tendência de Entrega Aplicada a Protocolos de Roteamento DTN em Ambientes de Redes Veiculares**. Graduação — Universidade Estadual do Ceará, UECE, 2014.

FILHO, J. G.; JÚNIOR, J. C.; PATEL, A. A new performance efficient trend of delivery mechanism applied to dtn routing protocols in vanets. In: **AICT 2014, The Tenth Advanced International Conference on Telecommunications**. [S.l.: s.n.], 2014. p. 98–104.

FU, Z.; MENG, X.; LU, S. How bad tcp can perform in mobile ad hoc networks. In: IEEE. **Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on**. [S.l.], 2002. p. 298–303.

GRASSI, G.; PESAVENTO, D.; PAU, G.; VUYYURU, R.; WAKIKAWA, R.; ZHANG, L. Vanet via named data networking. In: **IEEE INFOCOMM 2013 NOMEN Workshop**. [S.l.: s.n.], 2014.

GUO, S.; DERAKHSHANI, M.; FALAKI, M. H.; ISMAIL, U.; LUK, R.; OLIVER, E. A.; RAHMAN, S. U.; SETH, A.; ZAHARIA, M. A.; KESHAV, S. Design and implementation of the kiosknet system. **Computer Networks**, Elsevier, v. 55, n. 1, p. 264–281, 2011.

JACOBSON, V.; SMETTERS, D. K.; THORNTON, J. D.; PLASS, M. F.; BRIGGS, N. H.; BRAYNARD, R. L. Networking named content. In: ACM. **Proceedings of the 5th international conference on Emerging networking experiments and technologies**. [S.l.], 2009. p. 1–12.

JAIN, M.; PATRA, R. Implementing delay tolerant networking. 2003.

JUANG, P.; OKI, H.; WANG, Y.; MARTONOSI, M.; PEH, L. S.; RUBENSTEIN, D. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. **ACM Sigplan Notices**, ACM, v. 37, n. 10, p. 96–107, 2002.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of artificial intelligence research**, p. 237–285, 1996.

KAI, C. Rf and microwave wireless systems. **ISBN: 0-47135199-7, John Willey and Sons Inc., New York**, 2000.

KARP, B.; KUNG, H.-T. Gpsr: Greedy perimeter stateless routing for wireless networks. In: ACM. **Proceedings of the 6th annual international conference on Mobile computing and networking**. [S.l.], 2000. p. 243–254.

KHABBAZ, M. J.; ASSI, C. M.; FAWAZ, W. F. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. **Communications Surveys & Tutorials, IEEE**, IEEE, v. 14, n. 2, p. 607–640, 2012.

KOPONEN, T.; CHAWLA, M.; CHUN, B.-G.; ERMOLINSKIY, A.; KIM, K. H.; SHENKER, S.; STOICA, I. A data-oriented (and beyond) network architecture. **ACM SIGCOMM Computer Communication Review**, ACM, v. 37, n. 4, p. 181–192, 2007.

KRAJZEWICZ, D.; HERTKORN, G.; ROSSEL, C.; WAGNER, P. SUMO (Simulation of Urban MObility); An open-source traffic simulation. In: **4th Middle East Symposium on Simulation and Modelling (MESM2002)**. [S.l.: s.n.], 2002. p. 183–187.

LEONTIADIS, I.; MASCOLO, C. Geopps: Geographical opportunistic routing for vehicular networks. In: IEEE. **World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a**. [S.l.], 2007. p. 1–6.

LINDGREN, A.; DORIA, A.; SCHELÉN, O. Probabilistic routing in intermittently connected networks. **ACM SIGMOBILE Mobile Computing and Communications Review**, ACM, v. 7, n. 3, p. 19–20, 2003.

NGUYEN, A.-D.; SÉNAC, P.; DIAZ, M. et al. Stigmergy routing (stir) for content-centric delay-tolerant networks. In: **LAWDN-Latin-American Workshop on Dynamic Networks**. [S.l.: s.n.], 2010.

NGUYEN, A. D.; SÉNAC, P.; RAMIRO, V.; DIAZ, M. Swarm-based intelligent routing (sir): a new approach for efficient routing in content centric delay tolerant networks. In: ACM. **Proceedings of the 9th ACM international symposium on Mobility management and wireless access**. [S.l.], 2011. p. 137–142.

NILSSON, N. J. Introduction to machine learning an early draft of a proposed textbook department of computer science. **Standorf University**, 2005.

NSNAM. 2015. Acessado em 19-06-2015. Disponível em: <https://www.nsnam.org/>.

OH, S.-Y.; LAU, D.; GERLA, M. Content centric networking in tactical and emergency manets. In: IEEE. **Wireless Days (WD), 2010 IFIP**. [S.l.], 2010. p. 1–5.

OLIVEIRA, C. T. de; MOREIRA, M. D.; RUBINSTEIN, M. G.; COSTA, L. H. M.; DUARTE, O. C. M. Redes tolerantes a atrasos e desconexoes. **SBRC Simpósio Brasieliro de Redes de Computadores e Sistemas Distribuidos**, 2007.

PERINO, D.; VARVELLO, M. A reality check for content centric networking. In: ACM. **Proceedings of the ACM SIGCOMM workshop on Information-centric networking**. [S.l.], 2011. p. 44–49.

PROJETO-IPN. 2016. Acessado em 15-04-2016. Disponível em: <http://ipnsig.org/>.

PROJETO-NDN. 2015. Acessado em 01-01-2015. Disponível em: <http://named-data.net/>.

RICE, J. Seaweb acoustic communication and navigation networks. 2005.

ROLLA, V. G.; CURADO, M. A reinforcement learning-based routing for delay tolerant networks. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 26, n. 10, p. 2243–2250, 2013.

RUSELL, S.; NORVIG, P. Artificial intelligent: A modern approach. Prentice hall, 2003.

SARKAR, T.; JI, Z.; KIM, K.; MEDOURI, A.; SALAZAR-PALMA, M. A survey of various propagation models for mobile communication. **IEEE Antennas and Propagation Magazine**, v. 45, n. 3, p. 51–82, 2003.

SCHMIDT-EISENLOHR, F.; TORRENT-MORENO, M.; MITTAG, J.; HARTENSTEIN, H. Simulation platform for inter-vehicle communications and analysis of periodic information exchange. In: **Wireless on Demand Network Systems and Services, 2007. WONS'07. Fourth Annual Conference on**. [S.l.: s.n.], 2007. p. 50–58.

SIMON, H. A. Why should machines learn? In: **Machine learning**. [S.l.]: Springer, 1983. p. 25–37.

SINGH, S.; AGRAWAL, S. Vanet routing protocols: Issues and challenges. In: IEEE. **Engineering and Computational Sciences (RAECS), 2014 Recent Advances in**. [S.l.], 2014. p. 1–5.

SPYROPOULOS, T.; PSOUNIS, K.; RAGHAVENDRA, C. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: ACM. **Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking**. [S.l.], 2005. p. 252–259.

TALIWAL, V.; JIANG, D.; MANGOLD, H.; CHEN, C.; SENGUPTA, R. Empirical Determination of Channel Characteristics for DSRC Vehicle-to-vehicle Communication. In: ASSOCIATION FOR COMPUTING MACHINERY. **Proceedings of the First ACM International Workshop on Vehicular Ad Hoc Networks: October 1, 2004, Philadelphia, Pennsylvania, USA; Held in Conjunction with MobiCom 2004**. [S.l.], 2004. p. 88.

VAHDAT, A.; BECKER, D. et al. **Epidemic routing for partially connected ad hoc networks**. [S.l.], 2000.

VIEIRA, A. **Uma Aplicação de Auxílio à Mudanças de Faixa em Redes Veiculares**. Graduação — Universidade Estadual do Ceará, UECE, 2010.

VIEIRA, A. **VDTN-TD: Protocolo de roteamento Vanet/Dtn baseado em tendência de entrega**. Dissertação (Mestrado) — MACC, Universidade Estadual do Ceará, UECE, 2012.

VIEIRA, A. S. de S.; FILHO, J. G.; JÚNIOR, J. C.; PATEL, A. Vdtn-tod: Routing protocol vanet/dtn based on trend of delivery. In: **AICT 2013, The Ninth Advanced International Conference on Telecommunications**. [S.l.: s.n.], 2013. p. 135–141.

YOUSEFI, S.; MOUSAVI, M.; FATHY, M. Vehicular ad hoc networks (vanets): challenges and perspectives. In: IEEE. **ITS Telecommunications Proceedings, 2006 6th International Conference on**. [S.l.], 2006. p. 761–766.

YU, Y.-T.; JOY, J.; FAN, R.; LU, Y.; GERLA, M.; SANADIDI, M. Dt-ican: A disruption-tolerant information-centric ad-hoc network. In: IEEE. **Military Communications Conference (MILCOM), 2014 IEEE**. [S.l.], 2014. p. 1021–1026.

ZHANG, L.; AFANASYEV, A.; BURKE, J.; JACOBSON, V.; CLAFFY, k.; CROWLEY, P.; PAPADOPOULOS, C.; WANG, L.; ZHANG, B. Named data networking. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 44, n. 3, p. 66–73, Jul 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2656877.2656887>.

ZHAO, J.; CAO, G. Vadd: Vehicle-assisted data delivery in vehicular. **Vehicular Technology, IEEE Transactions on**, IEEE, v. 57, n. 3, p. 1910–1922, 2008.