Adaptação da Proposta de Rede Ad Hoc Veicular Centrada em Pontos de Interesse e Desenvolvimento de Aplicação Baseada no Framework Kura

*

Leandro Costa de Andrade
PESC/COOPE/UFRJ
CPS 731
Rio de Janeiro, Brasil
leandrolca@cos.ufrj.br

***Abstract* - These instructions give you the basic guidelines for preparing papers for IEEE conference proceedings.**

Index Terms - List key index terms here. No mare than 5.

I. INTRODUÇÃO

O objetivo deste relatório é demonstrar os progressos obtidos pelo autor na integração do protocolo RADNet à estrutura de software (*framework*) Kura. Ademais, embora não se tenha ainda concluído a implementação de um protótipo, este relatório propõe uma adaptação do protocolo RADNet para redes veiculares, abstendo-se da possibilidade de modificar o código fonte do protocolo RADNet, como propõe o artigo da referência [1] mas, ao invés disso, desenvolver uma proposta alternativa baseada no mesmo algoritmo proposto em ambiente simulado e fazendo-o através de pacotes de *software* (*bundles*) construídos sobre o *framework* Kura, em Java.

O desafio será encontrar uma forma eficaz de além de agregar valor ao protocolo RADNet já implementado, utilizando o algoritmo da referência [1], agregar também as facilidades e funcionalidades disponíveis no *framework* Kura.

II. O DESENHO DA RADNET

As RADNets, originalmente, suportam modelos de comunicação P2P e arquiteturas cliente-servidor e podem também, de forma geral, ser usadas em outras aplicações para redes móveis.

As RADNet são redes Ad Hoc dispensam a necessidade de endereçamento IP (Internet Protocol) e funcionam através do Protocolo REP, baseado em Prefixos Ativos (AP). A versão existente do REP foi implementada em ambiente Linux como um módulo de serviço denominado REPD.

Foram criadas, em complemento, interfaces de acesso (API) nas linguagens C, Phyton e Java para a plataforma Android, com o objetivo de viabilizar a construção de aplicações capazes de acessar o serviço REPD.

A Figura 1 apresenta um diagrama que ilustra o funcionamento do serviço REPD. Mostra como este serviço faz a ligação entre as aplicações que usam os modos de comunicação por interesse (IG) e comunicação origem-destino (S2D) – veja item B ambos, modos de endereçamento de mensagens.

O serviço REPD encapsula a mensagem que uma aplicação deseja enviar conforme a arquitetura do Protocolo REP e depois, empacota essa mesma mensagem em *Frames Ethernet*, esteja o dispositivo sob uma rede cabeada, ou rede sem fio.

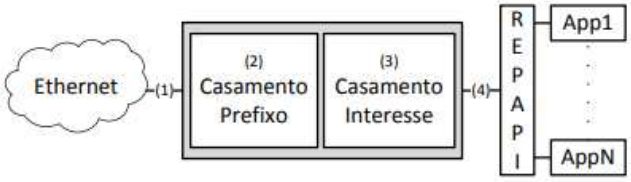

Fig. 1 Diagrama de Funções do Serviço REPD

Da mesma forma, o serviço REPD recebe mensagens da rede, as retira dos *Frames Ethernet* e inicia o processo de filtragem da mensagem recebida. O Serviço REPD analisa se existe correspondência em relação ao Prefixo P_i (S2D), ou o campo desse prefixo esta preenchido com *null*, isto é indicando uma mensagem em modo *broadcast* (IG). Faz esse trabalho, através de um filtro de casamento, conforme mostrado na Figura 1(2). Em seguida o serviço analisa se o interesse constante no campo *I* é coincidente com algum dos interesses declarados por cada aplicação, então repassa a mensagem para a(s) aplicação(ões), cujos interesses declarados são correspondentes ao interesse *I,* constante no cabeçalho da mensagem, Figura 1(3). As aplicações locais (App1-AppN) se comunicam com o serviço REPD, como mostrado na Figura 1(4).

A mensagem provida pelo serviço REPD terá no máximo 1476 Bytes e a string representando o interesse, no máximo 255 caracteres.

A partir desse momento, primeiramente, vai-se descrever o conceito de Prefixo Ativo (AP) e como é a sua organização arquitetural interna, onde nós construídos de maneira distribuída, são usados para comunicação e implementam o

* This work is partially supported by NSF Grant #2003168 to H. Simpson and CNSF Grant #9972988 to M. King.

encaminhamento de mensagens. Depois, será mostrado como os nós usam o conceito de Prefixos Ativos (APs) para endereçar mensagens para outros nós, ou para grupos de interesse.

A. *Prefixos Ativos e o Encaminhamento de Mensagens*

A Figura 2 mostra os dois componentes do Prefixo Ativo (AP) e a estrutura do cabeçalho de uma mensagem que o utiliza em uma rede RADNet típica.

Como se pode ver, na Figura 2(a) um Prefixo Ativo (AP) é uma estrutura de dados simples implementada em um nó, no nível da camada de rede (*network layer*). Ele é composto por um Prefixo do nó (P_i) e por um interesse de aplicação (I), o qual é usado por cada aplicação que está sendo executada para configurar nesta estrutura o seu interesse atual.

Já a Figura 2(b) mostra o cabeçalho de uma mensagem de Prefixo Ativo, conforme desenhado na versão atual do protocolo RADNet: a versão do protocolo, o limite do número de saltos permitidos, o tamanho do cabeçalho, o identificador da mensagem, dois prefixos de nó, identificando os prefixos do originador e do destinatário da mensagem e o interesse da aplicação.

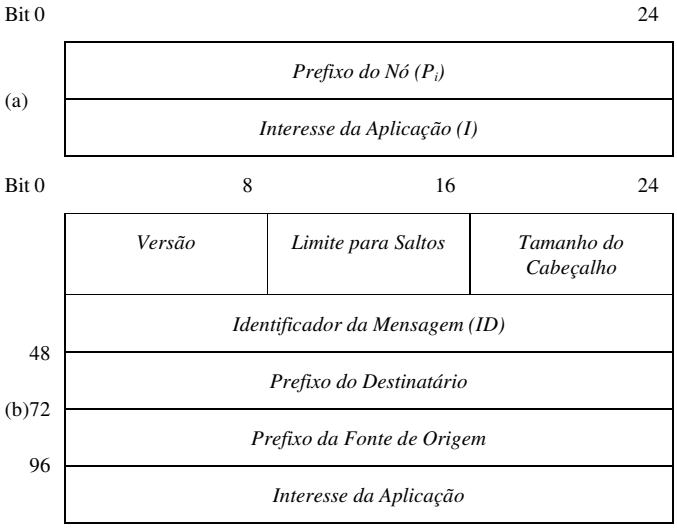

Fig. 2 Elementos do Protocolo RADNet: (a) Prefixo Ativo e (b) Cabeçalho da Mensagem de Prefixo Ativo

Em uma dada rede, os valores de n e m são configurados nos nós para gerar o seu próprio prefixo P_i com n campos e cada campo com m *bits*, de tal forma que n x m *bits* estabelecem a identificação de de cada nó para fins de endereçamento e filtro de correspondência para encaminhamento de mensagens.

Mais especificamente, um nó gera uma sequência de n campos, onde para cada campo o nó atribui um valor com m bits, a partir da aplicação de uma função de distribuição de probabilidade.

Em seguida, resulta um conjunto de valores de n campos que, na mesma ordem em que foram criados, constituirão o prefixo P_i do nó. Então esses n valores criados e armazenados nos respectivos n campos são concatenados na mesma sequencia de criação e formam o Prefixo de identificação do nó (P_i) (Ex: Pi_A = (1,2,4,0,3,8,9,5) => P_{iA} = (12403895)).

Por uma questão de simplicidade, usamos o P_i do nó e a identificação dos nós de maneira intercambiável.

O protocolo REP de comunicação, em sendo executado em um nó, implementa o encaminhamento de mensagens usando os campos P_i do nó.

Mensagens transmitidas no modo S2D (veja item B para esclarecer o significado de S2D e IG) tem o campo Prefixo da Fonte de Origem preenchido com o Prefixo P_i do nó de origem e o campo Prefixo do Destinatário preenchido com o P_i do nó de destino. O Protocolo REP, ao longo do processo de recebimento e filtragem da mensagem, verifica se existe correspondência entre o Prefixo do Destinatário da Mensagem e o Prefixo do próprio nó. Caso positivo, o Protocolo REP verifica o campo de interesse da mensagem e entrega a mensagem à(s) aplicação(ões) que registrou(ram) o mesmo interesse e não retransmite a mensagem porque ela atingiu o seu destinatário. Caso não ocorra correspondência integral entre o Prefixo do Destinatário e o Prefixo do próprio nó, o algoritmo do Protocolo REP busca uma correspondência de pelo menos um par (mesmo numeral na mesma posição (Ex: Pi_Dest = (1**2**403895) => $P_{i\_Nó}$ = (9**2**316754)). Caso positivo, a mensagem é retransmitida aos nós vizinhos. Caso contrário, é descartada.

Mensagens transmitidas no modo IG tem o campo Prefixo do Destinatário preenchido com *null*. Nesse caso, o algoritmo do Protocolo REP ao perceber o *null*, passa a verificar se o interesse da mensagem recebida é coincidente com o interesse registrado por alguma das aplicações que estão em execução no próprio nó. Caso positivo, ele faz uma cópia dessa mensagem e a entrega à aplicação, cujo interesse declarado é o mesmo que o interesse constante na mensagem recebida e retransmite essa mensagem aos nós vizinhos. Caso no nó não haja qualquer aplicação cujo interesse registrado seja o mesmo, a mensagem é descartada.

Além disso, o identificador da mensagem (*ID*) e o prefixo de origem são armazenados nos nós e usados para comparar com os campos correspondentes das próximas mensagens recebidas, para que as mensagens duplicadas sejam detectadas e não sejam encaminhadas.

Observe que o uso da transmissão de mensagens por interesse permite a uma aplicação que está sendo executada em um nó da rede, identificar diretamente outras instâncias da mesma aplicação em outros nós da rede, na mesma área de cobertura. Enquanto isso, o conteúdo transmitido da mensagem se presta para compartilhar informações entre as instâncias da mesma aplicação.

Quando cada nó cria o seu próprio identificador P_i, a partir de valores probabilísticos, dois objetivos importantes são alcançados: o primeiro, a estrutura de dados proposta de n campos com m *bits* de maneira distribuída implica em que a probabilidade de duplicidade do Prefixo do Nó seja menor que $(1/2)^{m \times n}$. Além disso, é possível ajustar os tamanhos de m e n de sorte que a probabilidade de coincidência de Prefixos P_i

entre dois nós, seja ajustada para adequar-se ao número estimado de nós que se pretende conectar em uma mesma rede; o segundo, o trabalho da referência [2] cita que já foi demonstrado que protocolos baseados em fofoca atingem altos níveis de entrega de mensagens e que a RADNet atingiu níveis de entrega equivalentes.

Há três vantagens a se considerar quando se usa prefixos ao invés de interesses para se transmitir mensagens em uma RADNet: a primeira, são evitados problemas relacionados pelo surgimento de nomes, usando interesse como uma sequência de caracteres de tamanho variável para um filtro de correspondência; a segunda vantagem, interesses muito populares podem causar uma inundação de mensagens na rede, enquanto outros interesses menos demandados pelas aplicações, podem dificultar ou impedir que as mensagens cheguem aos seus destinatários; a terceira, o uso de prefixos pode reduzir o isolamento de um nó, causado por interesses não coincidentes com os dos demais nós da vizinhança, de modo que esses nós mais próximos possam retransmitir mensagens mesmo que não haja interesses coincidentes.

Quando um nó preenche os campos P_i da mensagem, passa a retransmitir mensagens automaticamente, mesmo que não se tenha ainda configurado os respectivos interesses das aplicações em execução no campo Interesse da aplicação (*I*).

Ainda, pode acontecer que um nó tenha um prefixo P_i, tal que nenhum dos campos do seu prefixo coincida com os dos nós vizinhos. Nesse caso todas as mensagens enviadas pelo respectivo nó serão descartadas pelos nós vizinhos e vice-versa.

Assim, uma vez tendo sido detectado esse problema o nó pode ajustar o seu prefixo, por exemplo copiando e colando em um dos campos correspondentes do seu Prefixo, o valor de um do mesmo campo do Prefixo de um dos nós vizinhos, visando que passe a existir a requerida correspondência e as suas mensagens possam ser retransmitidas.

B. Modos de Endereçamento de Mensagens

Segundo a arquitetura das RADNets, cada aplicação em execução em um nó específico, configura o seu campo de Interesse da Aplicação (*I*) com as palavras chave apropriadas para identificar o seu respectivo interesse corrente.

É possível que uma certa implementação do Protocolo REP da rede RADNet suporte mais de um interesse declarado por aplicação. Porém, nesse caso, considera-se por simplicidade que cada aplicação declare apenas um interesse.

As RADNets usam o Prefixo Ativo para disponibilizar dois modos distintos de endereçamento: um chamado de origem-para-destino (Source-to-Destination – S2D); o outro modo de endereçamento é o modo Grupo de Interesse (Interest-Group – IG).

No modo S2D, o nó de origem necessita ter o descritor do P_i do nó de destino para enviar as mensagens. Já no modo de endereçamento IG, em contraste, o nó de origem precisa apenas informar o interesse a ser posto no campo *I* destinado a um grupo de nós que compartilha o mesmo interesse, enquanto o campo P_i é configurado como *null*, o que implica que a mensagem será transmitida em *broadcast*.

O cabeçalho da mensagem é usado como um filtro de correspondência com os Prefixos Ativos (APs) dos demais nós intermediários, a medida em que a mensagem é enviada de nó para nó ao longo da rede, até que a mensagem encontre o grupo de nós de destino, ou que seja descartada.

Caso ocorra uma correspondência de interesse em um nó intermediário, a mensagem é copiada e entregue à aplicação local que declarou o interesse enviado através da mensagem.

O nó de origem da mensagem, e o grupo ao qual se destina a mensagem pode continuar a se comunicar no modo IG, embora também seja possível que um dos nó do grupo possa mudar o modo de comunicação para o S2D, uma vez que o P_i do destinatário passe a ser conhecido.

III. REDE VEICULAR *AD HOC* - RADNET – VE

Entre os modelos arquitetônicos encontrados na literatura de redes centradas em informação (ICN), as redes centradas em conteúdo (CCN) ganharam proeminência no trabalho em redes veiculares.

Embora as CCNs sejam mais promissoras para ambientes veiculares do que os modelos centrados em IP, há algumas limitações que impedem sua adoção em projetos para redes veiculares - VANET.

Esses algoritmos são vulneráveis em ambientes veiculares altamente dinâmicos devido à intermitência inerente à variação das rotas.

Cenários em cujas aplicações necessitam trocar uma grande quantidade de dados sensíveis ao atraso ainda não foram exaustivamente estudados. Aplicações mais adequadas a esta finalidade, fazem uso de serviços de dados não armazenáveis em *cache*.

O que se vai apresentar neste capítulo é a descrição arquitetural de um novo modelo de ICN para VANETs. Trata-se de uma variante da RADNet (*Iterest-Centric Mobile Ad Hoc Network*) para ambientes veiculares (RADNet-VE).

No modelo proposto, assim como na RADNet, cada nó usa um Prefixo Ativo (AP) para compensar a falta de um protocolo centrado em no *Internet Protocol* (IP).

Essa proposta traz a vantagem de eliminar a necessidade de um identificador único para cada nó e a sobrecarga de manutenção devido à informação de roteamento na rede.

O Prefixo Ativo (AP) já foi definido no capítulo anterior e a descrição de sua estrutura e funcionamento descritas anteriormente, permanecem em vigor.

Além disso, este modelo, proposto pelo artigo da referência [1] sugere modificações de certas estruturas de dados e mecanismos estabelecidos na primeira versão da RADNet: no cabeçalho da mensagem, no mecanismo para registrar interesses no nível da camada de rede e no mecanismo para reencaminhar mensagens a outros nós.

O Protocolo RVEP da RADNet-VE pode encaminhar mensagens, considerando o prefixo da origem, o interesse declarado na mensagem, a posição relativa do nó de origem da mensagem, o identificador de vias e a direção de propagação.

Além disso, o Protocolo RVEP da RADNet-VE é capaz de limitar o escopo da comunicação de acordo com o identificador da via por onde um nó está passando e, em se tratando de mensagens para grupos de interesse, o protocolo pode limitar o número de saltos de uma mensagem e pode também implementar serviços de associação (*membership services*) de maneira distribuída, usando interesses (*I*) em mensagens de rede.

Os nós do RADNet-VE não armazenam dados em cache e não executam processamento na rede.

O Protocolo RVEP foi desenhado com base nos requisitos de comunicação de categorias de aplicação propostos pela referência [6].

Em uma rede RADNet-VE, todos os nós devem ser inicializados antes que possam transmitir mensagens. Durante a inicialização, as seguintes variáveis de controle e estruturas de dados são inicializadas:

1) contador de mensagens enviadas;
2) prefixo do nó;
3) tabela de prefixos de origem e identificadores de mensagens recebidas ($idTable_i$);
4) tabela de interesses e número máximo de saltos ($interestTable_i$);
5) tabela de posição relativa dos vizinhos dentro do raio de comunicação da rede de rádio ($posTable_i$); e
6) lista de identificadores de vias.

Seguindo o processo de inicialização, o nó registra os interesses das aplicações, conjuntamente com os respectivos números máximos de saltos permitidos.

Finalmente, se o nó é uma unidade fixada na borda de uma via (tipo 1), a lista de identificadores de vias pode ter mais de uma via registrada, porque esta unidade pode estar posicionada em um ponto por onde passe mais de uma via. Caso contrário, se o nó é um veículo (tipo 2), a lista de identificadores de vias só poderá conter uma via e será atualizada todas as vezes que o veículo mudar de via, ou se o nó é responsável por notificar obstáculos ao longo de uma via (tipo 3), a lista de identificadores de estrada deve ter só uma entrada, para que o nó publique exclusivamente os dados relativos às condições de somente uma via.

Para um nó transmitir uma mensagem, o protocolo de comunicação deve receber três itens de dados:

1) o prefixo de destino;
2) o interesse da aplicação; e
3) a direção da mensagem.

Antes de enviar a mensagem para os vizinhos, o nó monta a mensagem da seguinte forma:

1) configura o campo de versão com a versão do protocolo em um dado momento;
2) configura o número de saltos para zero (0). Assim, este valor será incrementado a cada vez que a mensagem for retransmitida;
3) configura o tamanho do cabeçalho da mensagem com um valor inteiro;
4) configura o campo do identificador da mensagem com o valor do contador da mensagem em um dado momento;
5) configura o prefixo do destinatário da mensagem;
6) configura o prefixo do nó de origem da mensagem com o prefixo do nó;
7) configura o campo de interesse da aplicação com o dado correspondente e de acordo com o interesse declarado da aplicação que está enviando a mensagem, ou com o já que estava registrado na mensagem que está sendo retransmitida;
8) configura os dados de posição com os dados obtidos do sistema GPS embarcado no dispositivo;
9) configura o campo de direção com o dado correspontente;
10) configura o identificador da via com o dado correspondente.

Na segunda etapa da comunicação, segundo o Protocolo RVEP, ao receber a mensagem msg_j o seguinte algoritmo é executado no nó:

1) O algoritmo verifica se o identificador da mensagem ($msg_j.id$) existe na tabela de prefixos de origem e identificadores de mensagens recebidas ($idTable_i$);
2) Se existir, o nó descarta a mensagem, caso contrário, o nó registra o msg_j em $idTable_i$ e incrementa o número de saltos da mensagem (+1);
3) Se o número de saltos no campo de número de saltos da mensagem ($msg_j.hopLimit$) é 1, o algoritmo insere a posição relativa do originador da mensagem na tabela de posição relativa dos vizinhos ($posTable_i$). Este procedimento permite a atualização das posições relativas dos vizinhos a um salto de distância;
4) Em seguida, o algoritmo verifica se o valor do identificador da via existe na lista de identificadores da via. Se não existir, o nó descarta a mensagem msg_j. Por outro lado, caso exista, ele calcula a posição do nó em relação ao nó de origem da mensagem. O resultado desse cálculo deve ser +1 ou -1, tornando possível determinar se o nó que originou a mensagem está a frente ou atrás do nó que recebeu a mensagem.
5) Baseado na posição relativa na Tabela $posTable_i$ o nó pode determinar se ele, ou qualquer de seus vizinhos está mais ou menos distante do nó de origem da mensagem. O algoritmo então extrai essa informação e a armazena na variável *fwdPrefix*.
6) Nos próximos dois passos, o algoritmo executa a filtragem da mensagem herdada do algoritmo da RADNet. Primeiro, ele verifica se a tabela de interesses do nó *i* ($interestTable_i$) tem uma entrada igual ao interesse da mensagem recebida ($msg_j.interest$);

7) Então, o algoritmo determina se o prefixo da origem ($msg_j.srcPrfx$) da mensagem RADNet-VE tem correspondência entre um ou mais pares dos campos, ou se o prefixo de destino é nulo.

8) Se ocorre uma correspondência entre um interesse, o algoritmo verifica se o nó i é a destinação da mensagem msg_j, ou se o prefixo de destino é nulo. Se for, o algoritmo cria uma cópia da mensagem msg_j e a encaminha para a aplicação associada com $msg_j.appIntrst$.

9) Antes de encaminhar a mensagem para outros nós, o algoritmo verifica se ocorreu uma correspondência de prefixo ou de interesse. Caso contrário, descarta a mensagem. Caso positivo, o nó *i* aguarda por um pequeno período de tempo através de uma função de distribuição uniforme entre 0 e 1, cujo valor aleatório de tempo obtido é dividido pela distância entre o nó receptor e o originador da mensagem antes de encaminhar a mensagem msg_j para os seus vizinhos. Entretanto, a retransmissão só ocorrerá se as condições a seguir forem satisfeitas:
   i. O nó *i* não é o destino da mensagem RADNET-VE;
   ii. O nó *i* é o nó mais distante do nó que originou a mensagem, segundo a tabela ($posTable_i$);
   iii. A msg_j não chegou ao limite máximo de saltos.

IV. Proposta de Adaptação dos Protocolos RADNet-VE e RADNet para o Framework Kura

Nesta seção será proposta uma adaptação do protocolo RVEP para a rede RADNet-VE.

A motivação dessa proposta vem da percepção de limitações inerentes às propostas feitas até o momento para os Protocolos REP e RVEP, tendo em vista necessidades operacionais, relativas ao controle e a gestão do tráfego em redes veiculares – VANET de alta densidade.

A seguir são listadas, a título de exemplo, algumas necessidades vislumbradas em um primeiro momento:

1) O protocolo RVEP precisa estar instalado e inicializado em todos os nós antes que a comunicação seja iniciada. Existe um problema logístico para operacionalizar a instalação e ativação remota em larga escala de instâncias do protocolo em cada nó de uma rede de alta densidade. O framework Kura pode oferecer uma solução viável para o gerenciamento do ciclo de vida do protocolo e aplicações RADNet;
2) Para aplicar o protocolo RVEP em cada nó é necessário garantir a atualização precisa de dados sensíveis ao tempo, por exemplo a atualização da via em que cada nó está posicionado à medida em que ele se desloca.

Algorithm 1: RAdNet-VE communication protocol

```
Input: msg_j
If msg_j.ID ∈ idTable_i[msg_j.srcPrfx] then
    Discard msg_j;
else
    Insert msg_j.ID into idTable_i[msg_j.srcPrfx];
    msg_j.hopLimit := msg_j.hopsLimit + 1;
    If msg_j.hopLimit = 1 then
        Insert msg_j.position into posTable_i[msg_j.srcPrfx];
    end
    If msg_j.roadId ∈ roadIdentifiers_i then
        positioning_i := calcPositioning(position_i, msg_j.position,
        msg_j.roadId);
        fwdPrfx := prefix_i;
        dist := calculateDistance(position_i, msg_j.position, msg_j.roadId);
        foreach position ∈ posTable_i do
            If calcPositioning(position, msg_j.position, msg_j.roadId) =
            msg_j.direction then
                neighborDist := calculateDistance(position,
                msg_j.position, msg_j.roadId);
                If neighborDist > dist ∧ neighborDist ≤ radioRange/2
                then
                    fwdPrfx := Neighbor's prefix;
                    dist := neighborDist;
                end
            end
        end
        intrstMtch := msg_j.interest ∈ intrstTable_i;
        prfxMtch := |prefix_i ∩ msg_j.srcPrfx| > 0 ∨ msg_j.destPrfx = null;
        If intrstMtch = true then
            If msg_j.destPrfx = null ∨ msg_j.destPrfx = prefix_i then
                Send a copy of msg_j to application;
            end
        end
        If prfxMtch = true ∨ intrstMtch = true then
            fwdMsg := msg_j.destPrfx = null ∨ msg_j.destPrfx ≠ prefix_i;
            nodePos := fwdPrfx = prefix_i ∧ (positioning = msg_j.direction
            ∨ msg_j.direction = 0);
            fwdHops := false;
            If intrstMtch = true then
                fwdHops := msg_j.hopLimit <
                intrstTable_i[msg_j.interest];
            else
                fwdHops := msg_j.hopLimit < maxHopLimitOfProtocol;
            end
            If (fwdMsg ∧ nodePos ∧ fwdHops) = true then
                Wait uniform(0, 1)/dist;
                Send msg_j to all neighbors_i ;
            else
                Discard msg_j;
            end
        else
            Discard msg_j;
        end
    else
        Discard msg_j;
    end
end
```

3) O gerenciamento do fluxo de veículos seria facilitado se houvesse aplicações auxiliares sendo executadas nas bordas da via para auxiliar cada nó a calcular a sua posição relativa em relação aos demais nós em uma mesma via;
4) É necessária uma aplicação para verificar eventuais coincidências de prefixos P_i dos nós em uma mesma via. Embora improváveis, poderiam levar a falhas catastróficas implicando em acidentes;
5) É interessante considerar a possibilidade de integração de aplicações RADNet com outras aplicações que usem protocolos baseados em endereçamento IP para comunicação de mensagens de gerenciamento com pontos mais distantes além do alcance da rede local, viabilizando, por exemplo, a comunicação antecipada de acidentes, obstáculos, engarrafamentos, etc;

6) Visando a utilização de algoritmos de otimização de fluxo de tráfego, a plataforma Kura torna possível a comunicação de informações úteis para sistemas de gerenciamento de fluxo responsáveis por áreas mais extensas.

Na primeira etapa deste trabalho, a proposta preliminar desse trabalho foi aproveitar o protocolo REP, já implementado e transformá-lo em mais um serviço do *framework* Kura. Para isso, seria preciso transformar o Serviço REPD escrito em linguagem C, em um componente de serviço compatível com o ambiente Kura e fazer a sua instalação no framework Kura, em uma placa Raspberry.

Entretanto, tendo em vista a premência de tempo e dificuldade técnica de aprendizado do *framework* Kura e de um serviço de integração de um protocolo complexo a este *framework*, optou-se por manter o Serviço REPD, funcionando em segundo plano, no ambiente Linux.

O Serviço REPD é executado em segundo plano e pode ser acessado por aplicações compatíveis com o ambiente Linux, desde que estas usem adequadamente as Interfaces de Aplicação (API), disponíveis nas linguagens C, Phyton ou Java.

A instalação do Protocolo REP se dá em sobre o Sistema operacional Raspibian uma placa Raspberry a partir dos seguintes passos:

1) Executar o comando: sudo apt-get install make gcc autoconf;
2) Você precisa compilar a instalar a biblioteca estática de utilidades para compilar o daemon. Então, ir ao diretório repa-c-utils e executar o comando: ./autogen.sh;
3) Em seguida, executar o comando: ./configure;
4) Depois, o comando: make;
5) Finalmente, o comando: sudo make install;
6) Nesse momento, passa a ser possível compilar o daemon. Então, ir ao diretório repa-daemon e executar o comando:make clean;
7) Depois, execute o comando: make.

Feita a instalação, haverá uma pasta apps/[arch] onde estará o binário "repad": daemon da Radnet. Esse binário deve estar sendo executado em cada nó enquanto se queira usar a RADNet.

O serviço é chamado à execução da seguinte forma: execute o comando "./repad -t" e será possível ver na tela os logs referentes ao funcionamento do serviço.

Os arquivos “librepa": são bibliotecas necessárias para se usar a rede RADNet.

Caso se faça a instalação, usando o comando "make install", pode ser executado o binário daemon, ou podem ser compilados programas da biblioteca sem se usar a pasta repa-daemon.Neste caso, o daemon pode ser acionado de qualquer lugar com o comando: "repad -t", lembrando que talvez seja necessário utilizar o usuário root.

Qualquer programa escrito em linguagem C que se queira compilar necessitará de incluir no compilador gcc e no código fonte o arquivo "repa.h", disponível na pastar hdr da pasta repa-daemon.

Caso este arquivo já tenha sido instalado, ele estará disponível para compilação do sistema, e só será necessário incluir o header no código fonte e a biblioteca dinâmica na linha de comando do gcc "-lrepa".

Na segunda etapa do trabalho, buscou-se portar uma das Interfaces de aplicação do Protocolo REP, aquela originalmente construída para o Sistema operacional Android escrita em Java para bundle no *framework* Kura.

Então, foi criado um bundle ufrj.coppe.lcp.repd_1.0.0.201906212239.jar, equivalente ao arquivo librepa.jar disponível para o ambiente Android, de modo que se possa usá-lo no *framework* Kura, importando-o como um pacote de quem depende um outro plug-in de uma aplicação ou como um plug-in requerido por outro.

Alternativamente, verificou-se a possibilidade de agregar ao Java Build Path as bibliotecas librepa.jar e librepa.so necessárias ao desenvolvimento das aplicações.

Na terceira etapa, foram pensadas três classes de aplicações descritas anteriormente como: tipo 1 - se o nó é uma unidade fixada na borda de uma via e a lista de identificadores de vias pode ter mais de uma via registrada; tipo 2 - o nó é um veículo; e tipo 3 - o nó é responsável por notificar obstáculos ao longo de uma via e sendo assim, a lista de identificadores de estrada deve ter só uma entrada, para que o nó publique exclusivamente os dados relativos às condições de somente uma via.

A arquitetura de software foi proposta considerando que o Protocolo REP permaneceria inalterado e que as funções do protocolo RVEP, proposto e testado em ambiente de simulação fossem implementadas na aplicação veículo, como um bundle.

Dessa forma, as variáveis e estruturas de dados não suportadas pelo protocolo REP, seriam construídas em um *bundle* auxiliar, que trabalhasse em complemento ao protocolo REP, em cada nó e que registrasse seus serviços e disponibilizasse interfaces adequadas para que as aplicações cliente tipos 1, 2 e 3, pudessem bem utilizá-los.

Durante a inicialização dos nós para operar com o Protocolo RADNet-VE, seriam inicializadas no *bundle* auxiliar ($RVEP_2$) e no Protocolo REP ($RVEP_1$), na camada de rede, as seguintes variáveis e estruturas:

1) ($RVEP_2$) - contador de mensagens enviadas;
2) ($RVEP_1$) - prefixo do nó;
3) ($RVEP_2$) - tabela de prefixos de origem e identificadores de mensagens recebidas ($idTable_i$);
4) ($RVEP_2$) - tabela de interesses e número máximo de saltos ($interestTable_i$);
5) ($RVEP_2$) - tabela de posição relativa dos vizinhos dentro do raio de comunicação da rede de rádio ($posTable_i$); e
6) ($RVEP_2$) - lista de identificadores de vias.

Seguindo o processo de inicialização, o nó registra os interesses das aplicações ($RVEP_1$) e em seguida, os respectivos números máximos de saltos permitidos ($RVEP_1$).

Segundo o Protocolo RVEP, para um nó transmitir uma mensagem, o protocolo de comunicação deve receber três itens de dados:

1) o prefixo de destino;
2) o interesse da aplicação; e
3) a direção da mensagem.

A direção da mensagem deve ser definida como +1 (enviada para os nós à frente do nó de origem, segundo a Tabela ($posTable_i$). Nesse caso, uma proposição é que a direção da mensagem (+1 ou -1), seja acrescentada como primeiros dois bytes da *string* de conteúdo da mensagem e sejam avaliados no *bundle* auxiliar, depois de a mensagem ter sido aceita pelo REP e repassada à aplicação.

Antes de enviar a mensagem para os vizinhos o nó, através do *bundle* auxiliar, monta a mensagem da seguinte forma:

1) configura o campo de versão com a versão do protocolo em um dado momento na camada de rede, protocolo REP ($RVEP_1$);
2) configura o número de saltos para zero (0) no terceiro byte disponível da *string* de conteúdo ($RVEP_1$). Assim, este valor será incrementado a cada vez que a mensagem chegar ao *bundle* auxiliar e for retransmitida;
3) configura o tamanho do cabeçalho da mensagem com um valor inteiro (RVEP1);
4) configura o campo do identificador da mensagem (RVEP1);
5) configura o prefixo do destinatário da mensagem (RVEP1);
6) configura o prefixo do nó de origem da mensagem com o prefixo do nó ($RVEP_1$);
7) configura o campo de interesse da aplicação com o dado correspondente e de acordo com o interesse declarado da aplicação que está enviando a mensagem, ou com o já que estava registrado na mensagem que está sendo retransmitida;
8) configura os dados de posição com os dados obtidos do sistema GPS embarcado no dispositivo na sequência, a partir do quarto byte da string de conteúdo;
9) configura o campo de direção com o dado correspontente nos primeiros dois bytes da string de conteúdo ($RVEP_1$);
10) configura o identificador da via com o dado correspondente, na sequência, após os dados de posição obtidos do sistema GPS ($RVEP_1$).

Na segunda etapa da comunicação, propõe-se a seguinte adaptação ao Protocolo RVEP, após o recebimento da mensagem e durante o seu tratamento pelo *bundle* auxiliar. a mensagem msg_j o seguinte algoritmo é executado no nó:

1) No modo S2D, o Protocolo REP, ao longo do processo de recebimento e filtragem da mensagem, verifica se existe correspondência entre o Prefixo do Destinatário da Mensagem e o Prefixo do próprio nó. Caso positivo, o Protocolo REP verifica o campo de interesse da mensagem e entrega a mensagem à(s) aplicação(ões) que registrou(ram) o mesmo interesse e não retransmite a mensagem porque ela atingiu o seu destinatário.
2) Caso não ocorra correspondência integral entre o Prefixo do Destinatário e o Prefixo do próprio nó, o algoritmo do Protocolo REP busca uma correspondência de pelo menos um par (mesmo numeral na mesma posição (Ex: Pi_Dest = (1**2**403895) => $P_i\_Nó$ = (9**2**316754)).
3) Caso positivo, a mensagem é retransmitida aos nós vizinhos. Caso contrário, é descartada.
4) No modo IG o campo Prefixo do Destinatário é preenchido com *null*. Nesse caso, o algoritmo do Protocolo REP ao perceber o *null*, passa a verificar se o interesse da mensagem recebida é coincidente com o interesse registrado por alguma das aplicações que estão em execução no próprio nó.
5) Caso positivo, ele faz uma cópia dessa mensagem e a entrega à aplicação, cujo interesse declarado é o mesmo que o interesse constante na mensagem recebida e retransmite essa mensagem aos nós vizinhos. Caso no nó não haja qualquer aplicação cujo interesse registrado seja o mesmo, a mensagem é descartada.
6) Além disso, o identificador da mensagem (*ID*) e o prefixo de origem são armazenados nos nós e usados para comparar com os campos correspondentes das próximas mensagens recebidas, para que as mensagens duplicadas sejam detectadas e não sejam encaminhadas.
7) Recebida a mensagem e encaminhada ao *bundle* auxiliar, o algoritmo verifica se o identificador da mensagem (msgj.id) existe na tabela de prefixos de origem e identificadores de mensagens recebidas (idTablei);
8) Se existir, o nó descarta a mensagem, caso contrário, o nó registra o msgj em idTablei e incrementa o número de saltos da mensagem (+1);
11) Se o número de saltos no campo de número de saltos da mensagem é 1, o algoritmo insere a posição relativa do originador da mensagem na tabela de posição relativa dos vizinhos (posTablei). Este procedimento permite a atualização das posições relativas dos vizinhos a um salto de distância;
12) Em seguida, o algoritmo verifica se o valor do identificador da via existe na lista de identificadores da via. Se não existir, o nó descarta a mensagem msgj. Por outro lado, caso exista, ele calcula a posição do nó em relação ao nó de origem da mensagem. O resultado desse cálculo deve ser +1 ou -

1, tornando possível determinar se o nó que originou a mensagem está a frente ou atrás do nó que recebeu a mensagem.

13) Baseado na posição relativa na Tabela posTablei o nó pode determinar se ele, ou qualquer de seus vizinhos está mais ou menos distante do nó de origem da mensagem. O algoritmo então extrai essa informação e a armazena na variável fwdPrefix.

V. Óbices e Questionamentos a Serem Respondidos

Há ainda dúvida em relação à estrutura de dados e sobre como é processada, no protocolo RADNet, a contagem do número de saltos da mensagem sendo transmitida, e sobre como é configurado o número máximo de saltos permitido para uma mensagem e se o número máximo de saltos é estabelecido por mensagem, por grupo de interesse ou é padronizado para uma dada rede.

Que variável é usada para contar o número de saltos no protocolo RADNet? Como e em que momento é feita a comparação do número de saltos de uma certa mensagem e do limite estabelecido para o número máximo de saltos?

Esse limite para o número máximo de saltos, sendo um limite padrão, poderia ser transmitido para um certo conjunto de nós pela via um serviço provido pela plataforma Kura?
Qual o critério mais eficiente para definir o número máximo de saltos de uma certa mensagem ou grupo de mensagens? Por tipo de mensagem? Por categoria de interesse?

Havendo mais de uma aplicação em execução no nó, como são registrados os interesses das respectivas aplicações? No nível da camada de rede do nó o protocolo REP registra o interesse de uma aplicação por vez, ou existe mais de uma instância do protocolo REP em execução respectivamente para cada aplicação em execução?

A princípio, há apenas uma instância do protocolo REP sendo executada por nó e as respectivas aplicações devem guardar em variáveis e estruturas de dados próprias as informações importantes, que serão usadas na montagem de cada mensagem de acordo com a necessidade de cada aplicação.

Será necessário se pensar no estabelecimento de mais de uma instância do Protocolo REP em execução concomitante para dar mais agilidade de comunicação ao nó? Qual será a correlação da necessidade de respostas sensíveis ao tempo das aplicações em relação a arquitetura cliente servidor ou P2P proposta pela arquitetura do REP?

Relativo a (2): o número de saltos é configurado para zero (0) em que campo da mensagem? Quando a mensagem é retransmitida, o valor referente ao número de saltos incrementado em que campo? Limite para Saltos? Nesse caso, em que variável ou campo é armazenado o valor correspondente ao número máximo de saltos permitido para a mensagem? Na Aplicação?

Relativo a (4): configura o campo do identificador da mensagem com o valor do contador da mensagem em um dado momento; Pode-se considerar a possibilidade de subdividir o campo do identificador da mensagem usando a primeira parte para categorizar a mensagem e a segunda parte dos 24 bits para registrar o número de saltos como um contador para a mensagem, enquanto o campo Limite para Saltos receberia o valor máximo de saltos autorizado para a mensagem?

Como caracterizar o campo (*ID*) identificador da mensagem?

É necessário criar um algoritmo para calcular a posição relativa do nó em relação ao nó de origem da mensagem. O resultado desse cálculo deve ser +1 ou -1, tornando possível determinar se o nó que originou a mensagem está à frente, ou atrás do nó que recebeu a mensagem.
Também é necessário, com base na posição relativa da Tabela $posTable_i$ o nó determinar se ele, ou qualquer de seus vizinhos está mais ou menos distante do nó de origem da mensagem. O algoritmo então extrai essa informação e a armazena na variável *fwdPrefix*.

Como calcular a distância relativa entre o nó receptor e os demais nós vizinhos e entre o nó originador da mensagem somente utilizando os resultados (+1) e (-1)? Como com base na tabela proposta, definir um sequenciamento dos nós considerando, por exemplo, a direção e o sentido do movimento do tráfego em uma via?

V. Conclusão

Nesse trabalho, os objetivos propostos, quais sejam os de integração do Protocolo REP ao *framework* Kura e da implementação parcial do Protocolo RVEP através de *bundles* em Java, foram parcialmente atingidos. Foi possível conhecer, estudar e instalar o Protocolo REP, estudar o Protocolo RVEP e compará-lo ao REP e propor uma arquitetura, que facilite a implementação parcial do RVEP, mas proporcione, em compensação, outras vantagens operacionais para resolver o problema da gestão de fluxo de tráfego urbano de alta densidade.

Não foi possível concluir a implementação do código necessário para iniciar ensaios e analisar a performance de protótipos em um cenário de teste a serem definidos. Também será necessário definir uma metodologia para testar a performance do sistema.

Ainda comentando sobre a melhor forma de verificar a qualidade do trabalho, outro estudo correlato indicou a possibilidade de se aplicar uma metodologia de verificação de software ao longo do desenvolvimento, visando garantir a qualidade dos processos de desenvolvimento, o que pode ser considerado em um ambiente comercial para o desenvolvimento de aplicações críticas.

Finalmente, é importante ressaltar nessa conclusão que, essa proposta objetivou estudar a possibilidade de se integrar uma rede centrada em informação (ICN), a partir de uma adaptação do Protocolo REP, desenvolvida por meio de *bundles*, para ser aplicada a sistemas digitais críticos, altamente dependentes comunicação, cujas aplicações são sensíveis ao atraso das mensagens, operando sobre redes móveis e cujos nós estão em constante movimento, percorrendo rotas.

Referências

[1] F. Goncalves, F. Franca and C. de Amorim, "Interest-centric vehicular ad hoc network," in 2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), New York, NY, USA, 2016 pp. 1-10.

[2] R. Dutra, H. Moraes and C. Amorim, "Interest-Centric Mobile Ad Hoc Networks," in 2013 IEEE 12th International Symposium on Network Computing and Applications, Cambridge, MA, USA USA, 2012 pp. 130-138.

[3] H. F. Moraes, R.C. Dutra, and C. L. Amorim, "On developing interest-centric applications for ad hoc networks," in Proceedings of the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimidia Networks – WoWMoM 2012 – To appear. Los Alamitos, CA, USA: IEEE Computer Society, 2012.

[4] C. Sommer, R. German, and F. Dressler, IEEE Transactions on Mobile Computing 10, 3 (2011).

[5] C. Somer. Veins Disponível em: http://veins.car2x.org/. Acessado em: June 23, 2019, 2014.

[6] Y. Huo, W. Tu, Z. Sheng and V. Leung, "A survey of in-vehicle communications: Requirements, solutions and opportunities in IoT," in 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, 2015 pp. 132-137.