TÉCNICO LISBOA

DroneSensor

Recolha de Informação de Sensores baseada em Drones

Ana Rita da Silva Pires

Dissertação para obtenção do Grau de Mestre em

Engenharia de Telecomunicações e Informática

Orientadores: Prof. Fernando Henrique Côrte-Real Mira da Silva
Prof. Teresa Maria Sá Ferreira Vazão Vasques

Júri

Presidente: Prof. Helena Isabel De Jesus Galhardas
Orientador: Prof. Fernando Henrique Côrte-Real Mira da Silva
Vogais: Prof. Rui Jorge Morais Tomaz Valadas

Outubro 2017

Agradecimentos

Expresso desta forma os mais sinceros agradecimentos por todo o apoio prestado a vários níveis durante a realização desta tese.

Aos orientadores Prof. Fernando Mira da Silva e Prof. Teresa Vazão pelo apoio prestado desde o início, tal como pelas críticas e sugestões, bastante importantes em todo o desenvolvimento.

À minha família, pelo apoio incondicional e dedicação demonstrados durante todo o percurso académico.

Aos meus amigos, que estiveram presentes em todos os momentos. Um especial agradecimento à Sónia e ao Tiago, por todo o apoio e compreensão nesta fase final.

Abstract

Delay Tolerant Networks result from the need to transfer information on wireless mobile networks in situations where traditional networks tend to fail. The main characteristic of these networks is the lack of continuous communication between their nodes and the difficulty of ensuring the delivery of the information to the recipient. This type of network is used especially in situations where the mobile nodes of the network are widely dispersed. This thesis proposes an implementation of a low cost system that allows the acquisition of sensor data installed in areas without infrastructure, using drones.

Keywords

Delay Tolerant Networks; Routing; Communication without Infrastructure

Resumo

As redes tolerantes a atraso, tipicamente designadas por *Delay Tolerant Networks* (DTNS), resultam da necessidade de realizar a transferência de informação em redes móveis sem fios, em situações em que as redes tradicionais falham. Estas redes têm como principal característica a falta de comunicação contínua entre os seus nós e a dificuldade de assegurar a entrega de informação ao destinatário. Este tipo de redes são usadas especialmente em situações em que os nós móveis da rede se encontram amplamente dispersos.

Esta tese propõe a implementação de um sistema de baixo custo que permite a recolha de dados de sensores, instalados em áreas sem infraestrutura, com recurso a *drones*.

Palavras Chave

Redes Tolerantes a Atraso; Encaminhamento; Comunicação sem Infraestruturas;

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

Acrónimos

DTN Delay Tolerant Networks

MANETs Mobile AdHoc Networks

TCP Transmission Control Protocol

TCP-CL TCP Convergence-Layer

BP Bundle Protocol

LTP Licklider Transmission Protocol

GPRS General Packet Radio Service

ISP Internet Service Provider

LTE Long-Term Evolution

IBR Institut für Betriebssysteme und Rechnerverbund

1

Introdução

Conteúdo

1.1 Enquadramento

Em diversas situações somos confrontados com a ausência de serviços de comunicação em determinadas áreas geográficas, em que a infraestrutura não existe ou onde há uma demanda por serviços de comunicação, mas nenhuma oferta adequada.

Como exemplo de situações como as acima descritas temos a recolha e transmissão de informação no mar. Pode existir a necessidade de recolher dados em determinados pontos do oceano, em que não existe infraestrutura que suporte a comunicação. Podemos ser ainda confrontados com ligações indisponíveis ou que devem ser evitadas por questões de segurança [1].

Uma outra situação em que esta falta de meios pode ser condicionante é na comunicação em meios rurais. Para monitorizar as condições do solo numa plantação de grandes dimensões, ou as condições atmosféricas numa zona remota em que o acesso à Internet não existe. Para efetuar a recolha de dados de sensores dispersos por áreas geográficas vastas, como referido acima, pode não ser viável a utilização de uma rede entre os nós e a sua comunicação direta com um nó central, pois para tal todos os nós teriam que estar ao alcance uns dos outros. Para que se possa monitorizar o estado do solo em diversos pontos do terreno, a distância entre os nós terá que ser de forma a que estes consigam comunicar entre eles, e com um nó central. A utilização de nós móveis que passam pelos sensores para recolher a informação e posteriormente por um nó central para transmitir essa mesma informação soluciona esta dificuldade de comunicação, na medida em que minimiza a distância entre os nós já que um nó móvel se desloca para que fique ao alcance do nó com o qual pretende comunicar. Para suportar o processo de recolha de dados, podem ser utilizadas as Delay Tolerant Networks (DTN). As redes tolerantes a atraso, usualmente designadas por DTN, resultam da necessidade de realizar a transferência de informação em redes móveis sem fios - Mobile AdHoc Networks (MANETs) - em situações em que a comunicação é esporádica, não sendo possível garantir conectividade continua entre os nós intervenientes [2]. As DTN têm como principal característica a falta de comunicação contínua entre os seus nós e a dificuldade de assegurar a entrega de informação ao destinatário. Este tipo de redes são usadas especialmente em situações em que os nós móveis da rede se encontram amplamente dispersos [3].

Existem ainda outros domínios e exemplos de aplicação das DTN, sendo que as aplicações mais comuns passam pela gestão de frotas de veículos, redes de comunicação para zonas rurais sem infraestruturas e operações de busca e salvamento [4].

1.2 Motivação

Para combater as dificuldades mencionadas na secção anterior, existem diversas opções possíveis. Ao utilizar tecnologias de rede móvel celular, tais como o Long-Term Evolution (LTE), o 3G e o General

Packet Radio Service (GPRS), é possível que nós distantes comuniquem entre si. Estas tecnologias, cuja essência é possibilitar o tráfego de dados por pacotes para que a ligação à Internet possa ser efetuada através da rede celular, permitem que os dispositivos enviem dados entre si ou para um servidor comum (um nó central). Esta solução é uma opção viável para a comunicação, no entanto é necessário o envolvimento de um Internet Service Provider (ISP), o que envolve custos elevados. Para além dos custos, para esta solução é necessário que os dispositivos mantenham uma conexão constante à rede móvel celular, o que leva a um elevado consumo de bateria. Outra das formas de mitigar o problema da comunicação a longas distâncias sem infraestrutura que a suporte, pode passar pela utilização de um nó que se movimente, com o auxílio de um *drone* (nó móvel), para recolher e entregar os dados no destino. Num cenário em que o objetivo é efetuar a recolha de dados de sensores pré-estabelecidos, e a posterior entrega num nó central que faz a sua gestão, o recurso a *drones* resolve os problemas de falta de conectividade entre os nós. O *drone* permite que os nós estejam dispersos visto que se pode deslocar para que fique ao alcance dos sensores e assim estabelecer a ligação para transmitir os dados. Este nó móvel pode efetuar a recolha, deslocar-se até ao nó central para transmitir os dados, e carregar a bateria até à próxima recolha. Deste modo, e com recurso a uma DTN para suportar a comunicação entre todos os nós, visto que sofre interrupções, é possível transmitir dados entre nós dispersos sem envolver custos elevados. Esta solução pode ser aplicada nas situações referidas na secção anterior, tais como a monitorização de condições atmosféricas e monitorização do solo em campos agrícolas.

1.3 Contribuições da tese

Esta tese descreve a implementação de um sistema que visa a recolha de dados de sensores em áreas vastas e sem infraestruturas, nas quais não é possível a ligação de vários nós através de uma rede fixa, recorrendo a tecnologias *open-source* e de baixo custo. Para além destas características, a solução pode ser utilizada em diversos contextos, bastando para tal adaptar os sensores utilizados aos dados que queremos recolher. A solução pode ser implementada nas mais remotas áreas, sem qualquer infraestrutura necessária.

1.4 Estrutura do Documento

O documento está organizado em capítulos de acordo com a seguinte estrutura. O capítulo 2 aborda o estado da arte relativamente a protocolos de rede utilizados em redes tolerantes a atraso, projetos já implementados neste âmbito e ainda às tecnologias usadas para os concretizar. O capítulo seguinte apresenta a arquitetura da solução, tendo em conta os resultados do capítulo anterior. O capítulo

4 aborda a implementação da solução proposta a nível tanto de *hardware* como de *software*, e os resultados obtidos com esta implementação. Por fim é apresentada a síntese do trabalho efetuado, as conclusões e o trabalho a desenvolver no futuro.

2

Trabalho Relacionado

Conteúdo

2.1 DTNs - Delay Tolerant Networks

As redes permitem a conexão de um grande leque de dispositivos em largas distâncias. Hoje em dia é possível conectar um telemóvel a milhões de servidores de todo o mundo. Apesar do sucesso destas redes, a sua utilização não é possível em todos os locais, e nalguns casos envolve um custo muito elevado. Esta limitação deve-se a um conjunto de premissas que não são verdadeiras em todos as situações e locais. A primeira premissa é referente à conexão ponto-a-ponto, que existe entre a fonte e o destino possivelmente através de nós intermediários. Esta é quebrada com facilidade com a mobilidade dos nós, bastando um dos mesmos ficar fora de alcance da rede para que a conexão seja perdida. As DTN são uma tentativa de ampliar as redes sem fio atualmente usadas, permitindo a comunicação em redes com nós a longas distâncias, redes de sensores, redes móveis ad-hoc, e redes *low-cost*, recorrendo a protocolos que tenham a capacidade de gerir a desconexão e o atraso. [5]

O conceito DTN surgiu inicialmente como uma abordagem à comunicação no espaço ou numa escala inter-planetária. Neste tipo de ambientes é inevitável a existência de uma grande latência, por vezes medida em horas ou até dias, e a perda de ligação. No entanto, estes problemas também se aplicam a locais não tão distantes mas nos quais não existe infraestrutura que permita a comunicação utilizando os protocolos existentes anteriormente [4]. As propriedades bastante peculiares das DTN, tais como a possível perda de conexão e o atraso na entrega da informação causam alguns problemas no que toca à sua criação e configuração, daí não ser possível a utilização dos protocolos já existentes até então. Estes problemas passam pela alocação de recursos, espaço de buffer, gasto energético, confiabilidade e segurança.

Os protocolos de rede, tais como o Transmission Control Protocol (TCP) não podem ser utilizados para este tipo de redes sem uma adaptação, visto que são protocolos orientados à conexão e requerem a existência de um atraso mínimo entre a fase de estabelecimento da conexão e a transmissão de dados, atraso esse que nas redes descritas acima pode ser de horas, dias ou até meses.

As secções seguintes apresentam o estado da arte relativamente a protocolos de comunicação utilizados neste tipo de redes, a tecnologia utilizada para a concretização de projetos nesta área e exemplos práticos de utilização das redes tolerantes a atraso.

2.2 Protocolos de Encaminhamento

As DTN são um caso especial das redes ad-hoc, que são um tipo de rede descentralizada que não necessitam de uma infraestrutura pré-existente, como um *router* ou um ponto de acesso para que exista comunicação, pois esta é feita diretamente entre os dispositivos da rede. Uma vez que o foco das DTN é ser tão eficiente quanto possível, e devido às suas características específicas referidas anteriormente, os típicos protocolos ad-hoc, como é o caso do MANETs, falham ao estabelecer caminhos

em DTN. Para aplicar o conceito referido acima, existe uma grande escolha no que toca a protocolos de encaminhamento. Estes dividem-se essencialmente em duas famílias [5]:

- Flooding Families
- Forwarding Families

A família de protocolos *flooding family* funciona num esquema de réplica. Os protocolos pertencentes a esta família tipicamente criam réplicas da mensagem, e propagam-nas pela rede até que chegue ao seu destino. Já os protocolos da família *forwarding family* baseiam-se no encaminhamento da própria mensagem. Tirando partido das vantagens de cada uma destas famílias, temos ainda o *Bundle Protocol*. É um protocolo que agrupa os dados em pacotes, e transmite-os utilizando *store-and-forward*, que implementa características pertencentes a cada uma delas. As secções seguintes apresentam alguns protocolos de ambas as famílias, bem como a descrição do *Bundle Protocol*.

2.2.1 Flooding Families

Os protocolos pertencentes às Flooding Families funcionam num esquema de réplica (criação de algumas cópias da mensagem ou pacote original, e a sua posterior entrega). Cada nó tem uma cópia de cada mensagem, que transmite para um conjunto de nós denominados *relays*. Todos os nós mantém cópias, que guardam num buffer, e que serão transmitidas aquando conexão estabelecida. Os protocolos baseados em replicação permitem uma melhor taxa de entrega de mensagens quando comparados com os protocolos baseados em encaminhamento, e não requerem conhecimento prévio da topologia da rede. A grande desvantagem dos protocolos desta família está relacionada com o espaço de armazenamento que cada nó tem que conter, uma vez que mantém cópias das mensagens a enviar.

Direct Contact - O protocolo *Direct Contact* é o caso especial das *flooding families*, uma vez que o conjunto de *relays* apenas contém o nó destino. É considerado como membro desta família de protocolos pois não requer nenhum conhecimento prévio da topologia da rede. [6] Este protocolo apenas permite que a transmissão da mensagem ocorra quando o nó fonte entra em contacto com o nó destino, ou seja, que os dados sejam transmitidos num só *hop*. Devido à sua simplicidade não consome muitos recursos no que toca a armazenamento e usa exatamente uma transmissão da mensagem quando a fonte fica diretamente ligada ao destino. À medida que o número de nós aumenta, a probabilidade de entrega da mensagem reduz para valores próximos de 0, pois o nó fonte pode nunca entrar em contacto direto com o nó destino [5].

Two-Hop Relay - O protocolo *Two-Hop Relay* baseia-se numa transmissão com dois *hops* entre a fonte e o destino. Se existirem n nós à volta da fonte, diretamente conectados, então devem ser geradas

n cópias da mensagem na fonte, e ser entregues a esses mesmos nós. A fonte e os *relays* guardam a mensagem até que entrem em contacto com o nó destino, para que esta seja entregue. Como nesse momento existem (n+1) cópias da mensagem na rede, há um maior consumo de largura de banda e de espaço de armazenamento.
Esta estratégia apresenta uma probabilidade de entrega da mensagem bastante superior à do protocolo descrito na secção anterior. Assumindo que cada nó entra em contacto com o nó destino da mensagem com uma probabilidade independente p, então a probabilidade de entrega de cada mensagem é 1-(1-p)(n+1), que para valores baixos de p é aproximadamente (n+1)p [5].
Ao aumentar o número de cópias da mensagem na rede, a latência média de entrega diminui visto que a mensagem é entregue quando um dos n+1 nós que possuem a mensagem entrar em contacto com o nó destino. Este protocolo sofre a mesma grande limitação que o protocolo *Direct Contact*: se nenhum dos nós que possuem a mensagem entrar em contacto direto com o destino, a mensagem não será entregue, embora neste exista uma maior probabilidade de entrega [6].

Tree-Based Flooding - O protocolo *Tree-Based Flooding* transmite os pacotes baseado numa estrutura de árvore, daí o seu nome. O protocolo descrito anteriormente, *Two-Hop Relay*, pode ser considerado como *Tree-Based Flooding* de profundidade 2. Neste caso, quando uma mensagem é copiada para um *relay* existe indicação de quantas cópias o próprio *relay* deve fazer. Para fazer estas cópias, existem diversas formas possíveis. A mais simples é permitir que cada nó faça cópias ilimitadas, mas restringir a sua viagem a um número máximo de *n hops* desde a fonte. Podemos ainda limitar o número de cópias que cada nó pode fazer, e assim limitamos a largura e profundidade da árvore. Este protocolo, ao contrário dos referidos acima, permite a entrega da mensagem a um destino que está a mais que um *hop* de distância, contudo definir os parâmetros pode ser um desafio. Se os valores forem demasiado conservadores, poderão ser feitas demasiadas cópias e causar uma grande sobrecarga na rede, caso contrário, se os valores forem demasiado restritos, a mensagem poderá nunca chegar ao seu destino [5].

Epidemic Routing - No protocolo *Epidemic Routing* todos os nós podem ser portadores da mensagem, ou seja, quando uma mensagem é enviada, é guardada no buffer local e etiquetada com um ID único. Quando dois nós se conectam enviam ao outro a lista de todos os IDs que têm nos seus buffers, denominada *summary vector*. Através da utilização destes vetores, os nós trocam as mensagens que não têm e quando a operação termina, têm as mesmas mensagens nos seus buffers. Isto garante uma probabilidade elevada de entrega de mensagens. O protocolo descrito é bastante simples na medida em que não é necessário conhecimento prévio da topologia da rede [5], embora consuma muitos recursos em termos de buffer space e cause uma grande sobrecarga na rede.

Probabilistic Routing - Quando uma mensagem chega a um nó que não tem contacto com outros nós, esta é guardada no buffer até que seja estabelecida conexão. É estabelecido um threshold nos nós. Um nó só pode receber uma dada mensagem se a sua probabilidade de entrega for superior ao threshold definido [3].

Os protocolos descritos acima apresentam vantagens e desvantagens, dependendo do objectivo da sua aplicação. O protocolo *Direct Contact* é um protocolo vantajoso na medida em que consome poucos recursos, e a sua configuração é bastante simples. A sua grande desvantagem prende-se com o facto de apenas entregar a mensagem quando os dois nós entram em contacto, o que reduz a probabilidade de acontecer no caso em que temos muitos nós na rede. O protocolo *two-hop relay* tem um comportamento semelhante ao descrito anteriormente, no entanto quando existe um número elevado de nós, ocupa muito espaço de armazenamento. O protocolos *tree-based flooding* oferece uma grande probabilidade de entrega da mensagem, no entanto também requer maior espaço de armazenamento, tal como o protocolo *Epidemic Routing*, que é vantajoso na medida em que garante a entrega de uma mensagem ao nó destino, no entanto, ao enviar uma cópia para todos os nós da rede, cria sobrecarga na mesma e todos os nós terão que guardar uma cópia da mensagem, ocupando espaço desnecessário. Por último, o protocolo *Probabilistic Routing* partilha as vantagens do anterior, mas apenas envia a mensagem para nós que tenham uma elevada probabilidade de entregar a mensagem ao destino. Para que tal seja possível, necessita de existir um padrão no movimento dos nós móveis para que estes cálculos probabilísticos sejam aplicados corretamente, logo é necessário um pouco de conhecimento acerca da movimentação do nó móvel, como a sua velocidade e aceleração, para que possa prever o padrão. De acordo com o objetivo definido para a solução proposta, em que o espaço de armazenamento é restrito e existe conhecimento prévio da localização dos nós, permitindo que os nós entrem em contacto quando necessário, o protocolo *Direct Contact* é o mais vantajoso desta família, primando pela sua simplicidade.

2.2.2 Forwarding Families

As Forwarding Families baseiam-se no encaminhamento de uma mensagem, em vez da sua cópia e distribuição. Estas utilizam a informação já conhecida acerca da topologia da rede para escolher o melhor caminho, e então a mensagem é encaminhada nó após nó até ao destino. Os protocolos pertencentes a esta família baseiam o encaminhamento da mensagem no conhecimento adquirido acerca da topologia da rede, de forma a enviar apenas uma cópia da mensagem pelo melhor caminho, não sendo necessário recorrer a replicação.

Location-Based Routing - Recorre a uma função de distância para estimar o custo de entrega das mensagens de um nó para outro. A localização dos nós podem ter significado físico, tais como as coordenadas GPS, ou apenas na topologia da rede. De um modo geral, a mensagem é encaminhada para o próximo nó se esse estiver mais próximo do destino que o nó atual. Tem como principal vantagem o facto de requerer pouca informação acerca da rede, mas, por outro lado, independentemente da distância entre dois nós, não há garantia que estes consigam comunicar entre si. Para além disso, as coordenadas dos nós estão sujeitas a alterações [6].

Source Routing - No *Source Routing* apenas o nó fonte necessita de conhecer a topologia da rede. Este está encarregue de toda a transmissão, e determina o caminho baseado na topologia da rede antes de a mensagem chegar ao nó [6]. O nó fonte tem que possuir inteligência suficiente para tomar a decisão de qual o caminho mais curto para que exista melhor performance. Apresenta uma boa performance apenas para os casos em que a fonte e o destino estão próximos.

Per-Contact Routing - A tabela de routing é recalculada de cada vez que um novo nó está disponível, em vez de calcular o próximo hop. Isto garante que cada decisão de routing é feita com a informação atualizada. Esta solução torna muito difícil evitar loops [5].

Hierarchical Routing - Este tipo de encaminhamento baseia-se numa rede hierárquica, que funciona como uma abstração da topologia da rede. Os nós são agrupados, e por cada grupo é definido um nó que tem conhecimento acerca dos outros do seu grupo, e dos restantes nós que representam os outros grupos. O nó fonte, ao enviar uma mensagem, tem que conhecer o endereço dos nós responsáveis pelos grupos, para que o encaminhamento possa ser efetuado. A decisão de encaminhamento para o próximo nó é feita pelo responsável do grupo aquando a receção. É um “hop-by-hop” routing, em vez de “source routing”. É um protocolo escalável para padrões de tráfego localizado, e não necessita de informação de localização. Tendo em conta que a informação de contacto é variável no tempo, é necessário um método auxiliar para agregar informação relativa ao tempo [5].

Os protocolos listados acima apresentam prós e contras na sua aplicação. O protocolo *Location-Based Routing* é um protocolo bastante complexo na sua aplicação, visto necessitar de uma função auxiliar para calcular o custo de entrega de uma mensagem, ainda assmi não garantindo a entrega da mensagem ao destino. O *Source Routing*, é um protocolo que consome poucos recursos, visto que apenas o nó fonte necessita de conhecimento prévio da rede, e é este que calcula o melhor caminho, no entanto a probabilidade de entrega da mensagem diminui com redes muito extensas. O protocolo *Per-Contact Routing* consome muitos recursos a nível de processamento devido à necessidade de recalcular a tabela de encaminhamento. Apesar da atualização constante da tabela, visto que os nós

podem estar em movimento, é muito provável a ocorrência de *loops*. Por fim, o protocolo *Hierarchical Routing* tem uma taxa de sucesso na entrega da mensagem elevado, mesmo para redes com muitos nós, no entanto necessita de métodos auxiliares para calcular o melhor caminho, e assim causa um maior consumo de recursos dos nós.

O protocolo *Source Routing* é o que apresenta menos desvantagens tendo em conta os pontos avaliados anteriormente. Em oposição aos restantes protocolos desta família, consome poucos recursos para determinar o caminho, tem uma implementação e preparação mais simples e não necessita de meios auxiliares para calcular o melhor caminho.

2.2.3 Bundle Protocol

O protocolo *Bundle Protocol* agrupa blocos de dados em pacotes, *bundles*, e transmite-os utilizando *store-and-forward*. A utilização deste protocolo permite que os dados sejam guardados durante largos períodos de tempo até que seja possível transmiti-los, mesmo em situações em que a conectividade à rede não é constante, e que existam quebras na largura de banda. Ainda assim, a entrega do pacote ao nó destino é garantida [7].

Este protocolo requer a utilização de um *Convergence-Layer adapter*[1] para enviar e receber bundles utilizando as ligações, redes ou protocolos já existentes. [8]

A implementação desenvolvida pelo Institut für Betriebssysteme und Rechnerverbund (IBR), e assim denominado IBR-DTN, deste protocolo inclui quatro módulos de encaminhamento distintos, e podem ser utilizados de forma concorrente. O primeiro módulo, *Static*, faz o encaminhamento entre rotas estáticas definidas à priori no ficheiro de configuração, as quais assume sempre disponíveis. O módulo *Neighbor* encaminha os pacotes para os nós vizinhos descobertos pelo *Discovery Agent*. Caso seja utilizado IPND - IP Neighbor Discovery, são alcançados os nós presentes na mesma sub-rede. Existe ainda uma implementação do protocolo *Epidemic Routing*, definida no módulo *Epidemic*. Este tipo de encaminhamento é uma variação de *flooding*, referida acima. Tem como objetivo o encaminhamento dos *bundles* por todos os nós participantes, e caso os recursos sejam ilimitados, garante a eventual entrega de uma determinado *bundle*. Quando um bundle é entregue ao destino, o mesmo é eliminado da rede para que não sejam recebidos bundles duplicados. Para tal são utilizada a informação constante nos *summary vectors*. O último módulo especificado nesta implementação é o módulo *Retransmission*, que como o próprio nome indica trata da re-transmissão de *bundles* caso ocorra algum erro no seu envio e consequente chegada ao destino. A retransmissão depende da natureza do erro ocorrido, permanente ou temporário.

Estes módulos recebem os eventos do *Discovery Agent* e interagem com o armazenamento do dispositivo. Os quatro módulos de encaminhamento descritos acima permitem a utilização de diversos

[1] Permite o envio e a recepção de blocos de dados utilizando as ligações nativas de uma rede ou protocolo

protocolos de encaminhamento de acordo com a preferência do utilizador.

- **Default routing (Direct Contact)** - Apenas efetua a entrega dos *bundles* aos vizinhos e aos nós estáticos.
- **Epidemic routing** - Envia para todos os vizinhos disponíveis.
- **Flooding routing** - Como o epidemic, mas não envia os *own summary vector* para os vizinhos.
- **Prophet routing** - Encaminha baseado na probabilidade de encontrar outros nós.
- **No routing**

2.2.4 Síntese e discussão

Ao analisar o protocolo mais relevante de cada família para a solução proposta, o *Direct Contact* e o *Source Routing*, verificou-se uma grande semelhança no que toca às suas vantagens e desvantagens. Ambos necessitam de pouco espaço de armazenamento, e pior performance no que toca a situações em que existem muitos nós, e nós muito distantes. O protocolo pertencente às *forwarding families*, *Direct Contact*, é o que melhor se adequa a esta solução devido à simplicidade de implementação e configuração.

2.3 Projetos

As DTN tem vários domínios de aplicação. Nas secções seguintes serão estudados diversos casos de uso deste tipo de redes.

2.3.1 SeNDT

O projecto SeNDT - Sensor Network with Delay Tolerance - foi desenvolvido por investigadores do Departamento de Engenharia Mecânica do *Trinity College Dublin* em 2007, tendo como objetivo inicial a monitorização da qualidade da água nas zonas rurais da Irlanda e a monitorização de ruído ambiental em áreas urbanas e auto-estradas nacionais [9].

O projeto tem por base a utilização de redes de sensores, neste caso sensores químicos e detetores de ruído, que em conjunto com dispositivos móveis formam os nós da rede. A transmissão dos dados dos sensores é efetuada através de nós móveis, os chamados *Data Mules*, que quando conectados aos sensores recolhem a informação e transmitem para um nó de armazenamento, usualmente designado por *data sink*. No caso da monitorização aquática o nó móvel será instalado num barco enquanto

que, para a poluição sonora, será instalado em camiões e outros veículos com rotas previsíveis. De momento, é referido pelos autores que ainda é usado um processo de recolha de dados manual.

Os sensores utilizados tem como característica comum o facto de terem uma potência elevada. Esta opção conduz a mais custos com o consumo energético, mas, em alternativa, permite que os sensores sejam mais precisos e que as distâncias de comunicação possam ser superiores, tornando assim a rede mais robusta [10]. Os sensores químicos são utilizados para obter informação sobre poluição aquática e os detetores de ruído para medir a poluição sonora. Em ambas as situações, estes sensores não estão em constante funcionamento, uma vez que são suportados por baterias, que recarregam com o auxílio de um painel solar. Para que os consumos energéticos sejam baixos, as unidades de monitorização aquáticas acordam durante 2 minutos de 30 em 30 minutos, enquanto as unidades de monitorização de ruído funcionam em ciclos de uma hora, registando a informação em intervalos de tempo de 5 minutos, 15 minutos ou 1 hora [10].

Tal como referido anteriormente, a informação coletada pelos sensores é transmitida para um nó móvel que a transportará posteriormente para o *data sink*. Para o efeito, são usados dois protocolos - o Bundle Protocol (BP), referido na secção anterior, ou o Licklider Transmission Protocol (LTP) -, que garantem a entrega dos dados em situações em que as redes convencionais falham.

O LTP é um protocolo de transmissão ponto-a-ponto (não tem nós intermediários), conhecido pela sua utilização em comunicações espaciais. O seu uso mais comum é o caso de uma ligação *single-hop*, no espaço, entre veículos espaciais e a estação Terrestre. Estas ligações, para além de extensas, sofrem de uma grande atraso condicionado pelos horários da estação terrestre e podem ainda sofrer interrupções [11]. Apesar de ser a utilização mais comum, não é exclusiva, pois pode ser aplicado em casos como o deste projeto. Tendo em conta que, como referido acima, o LTP é um protocolo de transmissão ponto-a-ponto, não existe *routing* nem controlo de colisões, os *bytes* são simplesmente transferidos entre dois pontos, sem intermediários [9].

A escolha entre os dois protocolos depende da dimensão da rede, tendo em conta a característica peculiar do protocolo LTP. No caso do projeto SeNDT, o protocolo escolhido pelos autores é o LTP. A difusão da informação pela rede tem um atraso que pode ir de um período de dias até meses. A recolha destes dados é feita de forma um pouco aleatória, dependendo dos requisitos dos dados recolhidos [12].

2.3.2 KioskNet

O projeto *KioskNet*, publicado em 2004, tem como objetivo possibilitar o uso da Internet em zonas rurais em países em desenvolvimento e em zonas com menos recursos económicos. Estes quiosques têm como fim providenciar uma variedade de serviços tais como certidões de nascimento, morte e casamento, e apoio nas áreas da medicina e agricultura. Para que a missão seja cumprida, a rede *KioskNet* é composta por cinco componentes fundamentais, os *kiosks*, os *ferries*, os *gateways*, um

proxy e por fim *legacy servers*.

O primeiro elemento, que dá nome à rede, é o *Kiosk*, e é constituído por um *single-board computer* que funciona como um servidor ao qual dois tipos de clientes se vão ligar. Os clientes mais comuns utilizam computadores reciclados, que fazem *boot* através da rede e executam os binários que o *kiosk* providencia. O segundo tipo de utilizador, tais como moradores com mais posses, pode aceder a um ou mais *kiosks* através dos seus dispositivos, sejam *smartphones*, *PDAs* ou computadores portáteis. Estes utilizadores utilizam o *kiosk* essencialmente como um *hotspot* que oferece ligação à Internet. O segundo elemento da rede são os chamados *ferries* que fazem a ligação dos dados entre os *kiosks* e a Internet. Apesar de os controladores dos *Kiosks* possuírem uma variedade de opções de conexão à Internet (GPRS, VSAT e *dial-up*), o objetivo do projeto é utilizar o método de transporte mecânico. Os *ferries* são carros, autocarros, comboios ou até mesmo motas, nos quais existe um *single-board computer* alimentado pela bateria do veículo. Estes computadores têm um espaço de armazenamento entre 20 e 40GB e uma interface WiFi, e comunicam com os *kiosks* e com os *Internet gateways* no caminho. Esta comunicação dura entre 20 segundos a 5 minutos, durante os quais se espera uma transferência de 10-150MB em cada direcção. Estes dados são guardados e encaminhados na forma de *self-identifying bundles*, que são enviados e recebidos de e para os *internet gateways* [13].

Os já referidos *gateways* consistem num computador com uma interface WiFi, armazenamento e conexão à Internet constante. Estes nós estão presentes principalmente em cidades com conexão DSL ou acesso de banda larga à Internet através de cabo. O *gateway* recolhe os dados aquando conectado a um *ferry*, guarda-os localmente, e só depois faz o *upload* para a Internet através de um *proxy*.

A comunicação entre o *kiosk* e a Internet deve garantir os serviços utilizados mais regularmente, tais como o email, transações bancárias e acesso a plataformas governamentais utilizadas pelos cidadãos. Existem diversos servidores que providenciam este tipo de serviço, mas não estão preparados para lidar com comunicações com grande atraso e desconexão frequente. Para tal, é necessário um *proxy* que consiga gerir a questão do atraso e da desconexão, e que faça a ligação aos *legacy servers*.

O último componente da arquitetura, os *legacy servers*, são tipicamente acedidos utilizando TCP/IP e um protocolo da camada de aplicação (POP, SMTP, HTTP). Estes servidores não necessitam de qualquer adaptação para funcionar com a *KioskNet* [13].

A comunicação entre os diversos elementos da rede é TCP/IP, embora a maioria dos elementos também utilize um *Delay Tolerant Networking overlay*, o DTN 2.0 da DTNR, com uma modificação que permite utilizar *flooding-based routing* para cada zona. Embora o DTN 2.0 providencie tolerância a desconexão, falha em serviços importantes, tais como a escolha por parte de uma aplicação a que dispositivo se ligar e a mudança entre quiosques. Para além destes, o DTN 2.0 não suporta totalmente a conexão com os *legacy servers*. Para resolver estas falhas, é usado o *Opportunistic Connection Management Protocol* (OCMP). O OCMP é visto como uma *session layer* tolerante à desconexão que corre

em cima do DTN e de outras conexões disponíveis. Cada tipo de comunicação utilizado é modelado como um objeto de conexão (CO) em OCMP, como por exemplo, o *backhaul* mecânico do DTN é encapsulado como DTN CO. Tal como este, COs semelhantes podem ser criados para as conexões TCP dos diversos tipos [14].

O maior desafio deste projeto é a criação/manutenção dos quiosques com custos reduzidos - custo inicial de $100-$700 por quiosque, e $70 de manutenção por quiosque por mês, que inclui o custo dos técnicos no local. Esta solução é cerca de 4-10 vezes mais barata que as restantes soluções existentes [14].

2.3.3 TrainNet

O projeto *TrainNet* consiste numa rede veicular que usa comboios para transportar dados tolerantes a atraso. Este projeto acrescenta valor à rede de comboios, equipando as estações e os próprios comboios com dispositivos de armazenamento, tais como discos rígidos. A rede criada tem duas aplicações: em primeiro lugar, proporciona uma ligação de elevada largura de banda que pode ser utilizada para entregar informação que não seja em tempo real. Em particular, os operadores de rede podem utilizar a *TrainNet* para distribuição de conteúdos como *Video On Demand*, que requerem uma elevada largura de banda, ainda tendo como vantagem a escalabilidade das ligações e do armazenamento, que facilmente pode ser incrementado. Isto é possível, pois cada estação possui um ponto de acesso/ligação. Em segundo lugar, mas não menos relevante, este projeto oferece uma alternativa economicamente viável de distribuição de conteúdos em zonas mais distantes que possuam ligação via carris. Desta forma, populações próximas a estações de comboios poderão ter acesso a conteúdos digitais tais como música, vídeo, televisão, programas e filmes com baixo custo [15].

A rede é composta por três tipos de nós, nomeadamente *source station*, a estação de onde o comboio parte, *destination station*, a estação destino do comboio, e por fim os próprios comboios (*trains*). Para a troca de dados entre a estação e o comboio, podem ser utilizados três métodos distintos. Antes da chegada do comboio, a estação passa todos os dados a ser enviados para um ou mais discos rígidos, e o mesmo acontece no comboio. Aquando a chegada do comboio à estação, pode ser usado qualquer um dos métodos descritos abaixo.

- Um funcionário troca os discos da estação com os do comboio.
- É utilizado um cabo de fibra ótica para ligar os discos da estação aos do comboio, e transmitir os dados. Nesta situação também é necessária a existência de um funcionário que ligue o cabo entre os discos, e que o desligue antes da partida do comboio.
- Pode ainda utilizar as *free space optical links*, que têm capacidade para transmitir a uma taxa de 1.25Gbps de dados. Esta situação não requer troca de discos entre a estação e o comboio, e não

Tabela 2.1: Aplicações de DTNs

Nome	Área de Aplicação	Protocolo
SeNDT	Poluição aquática e monitorização de ruído	Bundle Protocol, Licklider Transmission Protocol
KioskNet	Comunicação digital em zonas rurais	DTN2, OCMP
TrainNet	Transporte de grandes quantidades de dados para grandes áreas geográficas	N/A

requer um funcionário dedicado a essa tarefa.

Devido ao espaço de armazenamento limitado em cada um dos nós e às restrições aplicadas ao modelo explicado acima, o projeto está em evolução propondo como trabalho futuro a comunicação entre comboios, considerar múltiplos carris, e as suas intersecções e a aplicação do melhor algoritmo de alocação de espaço para situações em que o espaço livre nos discos não seja suficiente.

2.3.4 Síntese e discussão

Conforme se pode verificar pela análise da tabela 2.1, que resume as principais características das aplicações anteriormente apresentadas [12], todos os projetos descritos são vantajosos na sua aplicação, e as desvantagens apresentadas não são limitativas. Tendo em conta que o projeto *TrainNet* ainda não utiliza as DTNs para conexão e transferência de dados entre os elementos da rede, não é possível comparar a eficiência dos protocolos utilizados com os restantes projetos, mas é vantajoso na medida em que é percetível o problema de armazenamento neste tipo de redes. O projeto *SeNDT* é o mais semelhante ao tema a ser abordado neste documento, visto que consiste numa arquitetura composta por sensores e nós móveis, cujo objetivo é a recolha de dados, transmissão, e posterior processamento. Uma desvantagem deste projeto está relacionado com o protocolo usado que apenas contempla a situação em que apenas existem o nó fonte e o nó destino, não podendo ser enviado para outro nó intermediário mesmo que seja para um minimizar o atraso na entrega dos dados. O protocolo utilizado neste projeto é vantajoso na medida em que suporta interrupções na conexão, contudo, não contempla controlo de colisões. O projeto *KioskNet*, que utiliza uma adaptação do protocolo DTN 2.0 da DTNRG tem a grande vantagem de utilizar recursos já disponíveis, como os servidores, e apenas adaptar a comunicação a redes com tolerância a atraso. No que diz respeito à transmissão de dados, esta está sujeita ao movimento das pessoas com *smartphones* ou a transportes públicos, o que pode trazer um grande atraso na sua entrega, e assim tornar a acessibilidade à rede mais reduzida. Em termos de protocolos, o OCMP é compatível com os protocolos de encaminhamento já usados, pelo que não é necessário fazer uma implementação específica, embora, ao adicionar mais informação às

mensagens já trocadas, torna-as maiores e causa uma maior carga na rede.

2.4 Tecnologia

2.4.1 Sensores

Um sensor é um dispositivo capaz de converter uma quantidade física a ser medida, num sinal que possa ser lido, mostrado, guardado e processado. Existem diversos tipos de sensores com características bastante distintas, dependendo da utilização pretendida.

Os sensores mais comuns dependem de uma outra unidade, tal como um micro-controlador, que receba, processe e envie os dados adquiridos. Existem sensores para diversas aplicações, tais como os de temperatura, humidade, movimento, entre outros. Como exemplo deste tipo de sensores temos um dos sensores de temperatura mais populares, o LM35, que é um circuito integrado de 3 pinos (+5V, GND e saída). Este sensor em específico deteta temperaturas entre -55ºC e 150ºC, com um erro aproximado de 0.5ºC [16].

Outro tipo de sensores, os chamados sensores autónomos, só dependem de si mesmos. Possuem alimentação própria, capacidade de processamento dos dados, e até módulos de comunicação. Um exemplo deste tipo de sensores, é o *Wireless Temperature Sensor* da *ConnectSense*, que é alimentado por uma bateria de 5V, um módulo de comunicação Wi-Fi, um gestor de dados que envia a informação para a *cloud*, e portátil. Este sensor deteta temperaturas entre 0 e 80ºC, e tem uma precisão de +-0.5ºC. Estes, apesar de não necessitarem de mais nenhum dispositivo são desvantajosos na medida em que o seu preço é bastante superior aos ditos sensores comuns. [17]

2.4.2 Micro-controladores

Os micro-controladores são necessários para fazer toda a recolha e processamento de informação. As principais características que fazem destes dispositivos a melhor opção em detrimento, por exemplo, de um dispositivo móvel, estão relacionados com o seu custo, o baixo consumo energético, o peso e a flexibilidade no uso de sistemas operativos e linguagens de programação. Para além disso, permitem a adição de módulos para as mais diversas situações, tais como módulos de comunicação, módulos GPS, entre outros.

Arduíno UNO - O Arduíno é um micro-controlador *open-source* baseada em *hardware* e *software* de simples uso. As placas Arduíno têm a capacidade de ler dados de *input*, como dados provenientes de sensores, e transformá-los num *output*, em acções tais como ativar um motos, ligar um LED ou até mesmo enviar a informação para outro dispositivo. Para além das funcionalidades básicas, a este

dispositivo podem ser adicionados módulos que permitem funções mais avançadas, que são os chamados *shields*. É um micro-controlador de baixo custo, com um ambiente de programação e configuração simples, com hardware e software expansíveis. Em termos de alimentação, utiliza a ligação micro-usb, podendo ser ligado facilmente a uma fonte de energia [18].

Raspberry Pi - O Rasperry Pi é um micro-computador, que corre um sistema operativo baseado em LINUX. Para além de funcionar como micro-controlador, tem algum poder de processamento, pelo que é elegível para tarefas mais complexas de processamento de dados. A sua memória é expansível através da utilização de um cartão de memória com maior capacidade, ou até mesmo com um dispositivo de armazenamento USB [19].

armStone - O armStone é um *single-board computer* de baixo custo. Para além das diversas interfaces que possui, traz um sistema operativo pré-instalado, que pode ser alterado. É compatível com Windows, LINUX e futuramente com Android e IOS. Á semelhança do Raspberry Pi, descrito anteriormente, também pode ser usado como micro-controlador [20].

2.4.3 Drones

Para o projeto proposto, o drone terá como função a recolha de dados fornecidos pelos sensores. Para tal, é necessário que possua autonomia para percorrer a rede de sensores. Para além disso, de forma a evitar colisões, será vantajosa a presença de uma câmara e tendo em conta que a rota terá que ser programável, a sua API deverá estar disponível. Para que a recolha dos dados seja possível, o drone deverá ainda suportar peso extra, para que o micro-controlador possa ser agregado a este. Atualmente, o mercado deste tipo de dispositivos oferece uma grande variedade de produtos. Os drones apresentados abaixo têm a característica comum de a sua API estar disponível, sendo assim possível a sua configuração e adaptação ao projeto.

Parrot - Os drones Parrot são uma solução comercial e pronta a utilizar para fins lúdicos, embora tenham uma interface de configuração que permite uma utilização mais avançada. Estes *drones* possuem uma c—amara HD integrada, uma estrutura robusta, WiFi, um processador de 32 bits com Linux instalado e ainda sensores de pressão e ultrassons para o controlo de voo. Têm uma autonomia de 9 a 25 minutos, dependente da carga adicional ao peso do próprio *drone* [21].

DJI - Os *drones* DJI, populares pelo modelo mais comercializado *Phantom*, possuem características semelhantes às dos *drones Parrot*. É também disponibilizado um *Software Delevolpment Kit* que permite a obtenção de imagens do percurso, e ainda definir a rota que o *drone* deverá seguir. Para além

dos modelos *Phantom*, existe o *Matrice 100*, que é um *drone* completamente programável, adequado para desenvolvimento. Este modelo está preparado para que lhe sejam adicionados dispositivos externos (sensores, controladores, câmaras, entre outros). A sua autonomia depende do peso adicionado ao peso base do *drone* [22].

2.4.4 Síntese e discussão

Na secção anterior foram apresentados dois tipos de sensores: os sensores dependentes de um micro-controlador e os sensores autónomos. Qualquer um dos tipos apresenta características vantajosas, tais como as mencionadas acima. Os critérios utilizados na escolha de sensores estão relacionados com o tipo de utilização prevista, mas características como a precisão, o alcance, a calibração, a autonomia e o custo são fatores bastante importantes na escolha de um bom sensor. Os sensores autónomos são alimentados por baterias próprias, enquanto os sensores dependentes de micro-controladores, para além de consumirem menos energia devido à sua simplicidade, são alimentados pelo próprio controlador. Uma vez que o mercado de sensores autónomos ainda está em crescimento, a utilização de sensores dependentes de outros dispositivos oferece uma maior flexibilidade, escalabilidade, para além de terem um custo inferior. Desta forma, no contexto da presente tese será dada preferência à utilização de sensores deste tipo, isto é, sensores que se integrem facilmente com micro-controladores, sempre que se encontrem disponíveis sensores com as características pretendidas.

Relativamente aos micro-controladores foram apresentados três tipos de dispositivos. Os micro-controladores *Rasberry Pi* e *armStone* diferem no número de interfaces *ethernet*, uma e duas respetivamente. O peso, outra característica muito importante, no caso do *armStone* é de aproximadamente 55g, apenas mais 10g que o *Raspberry Pi.* Em termos de utilização, tendo em conta o seu preço de mercado e as características especificadas acima, o Raspberry Pi apresenta vantagem relativamente ao preço de mercado e à flexibilidade, uma vez que suporta o sistema operativo LINUX, que é o sistema operativo pretendido. Para o controlo dos diversos sensores, a utilização do Arduíno é vantajosa em termos de facilidade de recolha de informação, e, recorrendo a um *shield* WiFi, é possível transmitir os dados para a unidade de processamento. Para a aplicação pretendida, o *Rasberry Pi* tem preferência devido à sua versatilidade.

Por último, em relação aos drones foram comparados três dispositivos. Os *Parrot*, o *DJI Phantom* e o *DJI Matrice 100*. Os dispositivos apresentados diferem principalmente no custo, sendo que nesse aspeto os *Parrot* são mais vantajosos, e ainda, tendo em conta que são um modelo mais conhecido, existe mais *software open-source* disponível. No caso do *Phantom*, a utilização de outros dispositivos eletrónicos em cima do aparelho é possível, mas não é aconselhado pela marca pois pode afetar a

capacidade de voo do *drone*.

3

Arquitectura

Conteúdo

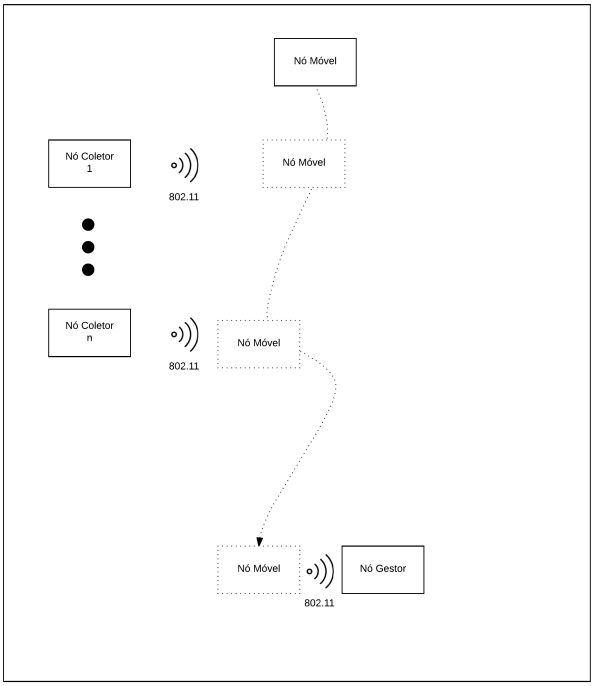

Figura 3.1: Diagrama da Rede

Nesta secção é apresentada a arquitectura da solução, que consiste num sistema de recolha de dados de sensores com recurso a tecnologias de baixo custo e *open-source*.

3.1 Visão geral da rede dsensor

A solução encontrada para implementar a arquitetura passa pela utilização de um nó móvel que passa pelos sensores para recolher a informação e termina o seu percurso no nó que gere a informação. A arquitetura proposta será assim baseada numa rede do tipo DTN, na qual são considerados três tipos de nós: nós coletores, nó móvel e nó gestor.

Os nós coletores são constituídos por um conjunto de sensores e pelo micro-controlador que faz a gestão da informação dada pelos sensores, e envia através de um módulo WiFi. O nó móvel consiste num *drone* com um micro-controlador agregado e alimentado por uma bateria, e tem como função a recolha dos dados junto dos nós coletores. Este nó, como o próprio nome indica, vai estar em movimento por forma a ficar ao alcance de cada um dos nós coletores. Por fim, o nó gestor é composto por um computador que faz o tratamento da informação enviada pelo nó móvel e permite a sua visualização. A figura 3.1 apresenta um diagrama geral da rede e da comunicação entre os diferentes nós.

Os nós coletores são distribuídos pelo local de implementação do projeto, tendo uma posição fixa conhecida pelo controlador do nó móvel. Estes nós vão recolher periodicamente informações acerca

de temperatura, luminosidade, pressão atmosférica, humidade e ruído do local, e guardar a informação até que o nó móvel faça a recolha desta informação. Os nós coletores, embora inseridos na mesma rede, não comunicam entre si. Cada um deles apenas envia dados para o nó móvel aquando conectados. Posteriormente à recolha dos dados e recorrendo ao nó gestor, o utilizador conseguirá aceder à informação.

O nó móvel recolhe a informação periodicamente e guarda-a localmente. Enquanto isso, verifica se o nó móvel está ao alcance para que possa enviar a informação recolhida. Da mesma forma, e com uma periodicidade menor, o nó móvel verifica se algum nó coletor está ao seu alcance para que seja estabelecida conexão.

Os dados recolhidos são guardados num ficheiro de texto, e a cada aquisição nova é acrescentada uma nova linha. Estes dados poderiam ser guardados em formato binário, o que reduziria o espaço ocupado pelos mesmos, no entanto, tendo em conta a maior dificuldade na manipulação deste tipo de dados e a quantidade de informação a recolher e transmitir, não justifica o processamento necessário para os converter em texto após receção no nó destino. Ao serem transmitidos, os ficheiros são divididos em blocos pelo que, para que sejam enviados de uma só vez, o ficheiro não deverá exceder o tamanho definido para um bloco. Os dados recolhidos consistem numa linha de um ficheiro de texto que contém a identificação do nó que os recolhe e os valores lidos pelos sensores desse nó. A periodicidade de recolha dos dados por parte dos nós coletores depende da preferência do utilizador e do tipo de dados recolhidos. No caso em que os sensores utilizados monitorizam as condições do solo e da atmosfera, não é necessário efetuar recolhas em curtos períodos de tempo, visto as alterações não serem muito frequentes. De acordo com os valores definidos por defeito na implementação do protocolo escolhido, referido e explicado em detalhe na secção 4.1.2, cada bloco tem 4096 bytes. Este valor é referido pelos autores do documento que apresenta este protocolo como sendo o valor com melhores resultados para ligações sem fios [23]. Tendo em conta que cada amostra consiste numa linha de um ficheiro de texto, pelas diversas amostras recolhidas, cada uma ocupa em média 51 bytes. Para que os ficheiros não excedam o tamanho de um bloco, como referido anteriormente, cada ficheiro poderá conter cerca de

$$4096 \div 51 \simeq 80$$

amostras. Para esta situação, se o nó móvel se conectar aos nós coletores duas vezes por dia, ou seja de 12 em 12 horas, a recolha de amostras poderá ter uma periodicidade de aproximadamente 10 minutos.

$$\frac{80}{12 \times 60} \simeq 0.11$$

Na situação em que o buffer do nó coletor atinge o máximo da sua capacidade, serão perdidos os dados mais antigos, tal como acontece com o nó móvel. Ao recolher os dados de todos os nós

coletores, o primeiro ficheiro recebido será o primeiro a ser eliminado em detrimento da receção de um novo.

3.2 Arquitetura de Software

3.2.1 Nó coletor

O nó coletor, como referido anteriormente, é o responsável pela recolha dos dados obtidos através de sensores em contacto com o exterior. A figura 3.2 apresenta a arquitetura de software deste tipo de nó. É composto por um conjunto de sensores ligados ao dispositivo de controlo, que contém um módulo de gestão de comunicação, um de transmissão de dados, processamento de dados, leitura de sensores e uma interface de comunicação. O módulo de gestão de comunicação em conjunto com a interface de comunicação detetam a presença do nó móvel e efetuam a ligação e transmissão de dados para o mesmo. O módulo de processamento dos dados processa e transforma os dados recolhidos pelo módulo de leitura dos sensores, dados esses que são enviados para o módulo de transmissão de dados que por sua vez os envia para o nó móvel através da interface de comunicação.

Os sensores recolhem os dados do meio ambiente, e guardam-nos localmente num ficheiro. Este ficheiro contém a data e hora da aquisição, o identificador do nó, e os dados recolhidos dos diversos sensores. Periodicamente, e de acordo com o tipo de dados a recolher, é efetuada uma recolha e adicionada uma linha ao ficheiro. Estas recolhas terão que ser efetuadas tendo em consideração o tamanho máximo que o ficheiro pode atingir para que a sua transmissão integral seja efetuada no período de tempo para o qual os dados são relevantes. Por exemplo, num sistema agrícola que monitoriza a condição do solo e que utiliza os dados recolhidos de 12 em 12h para verificar a necessidade de rega nesse momento ou não, se o ficheiro de dados exceder o tamanho máximo para que seja transmitido numa só recolha, quando o ficheiro for finalmente transmitido esses dados já não serão relevantes para o utilizador. Desta forma, para que não sejam perdidos dados relevantes a uma ação, a periodicidade da recolha por parte do nó coletor deverá ser em função do tamanho máximo de ficheiro e da conexão com o nó móvel.

A listagem 3.1 é um exemplo de ficheiro de dados recolhidos pelo nó coletor 1, no momento apenas com um sensor de temperatura ligado. Tendo em conta que os relógios dos diversos nós coletores não estão sincronizados entre si, a hora das recolhas não é precisa, podendo diferir em alguns segundos, o que não é relevante para o tipo de aquisição efetuada.

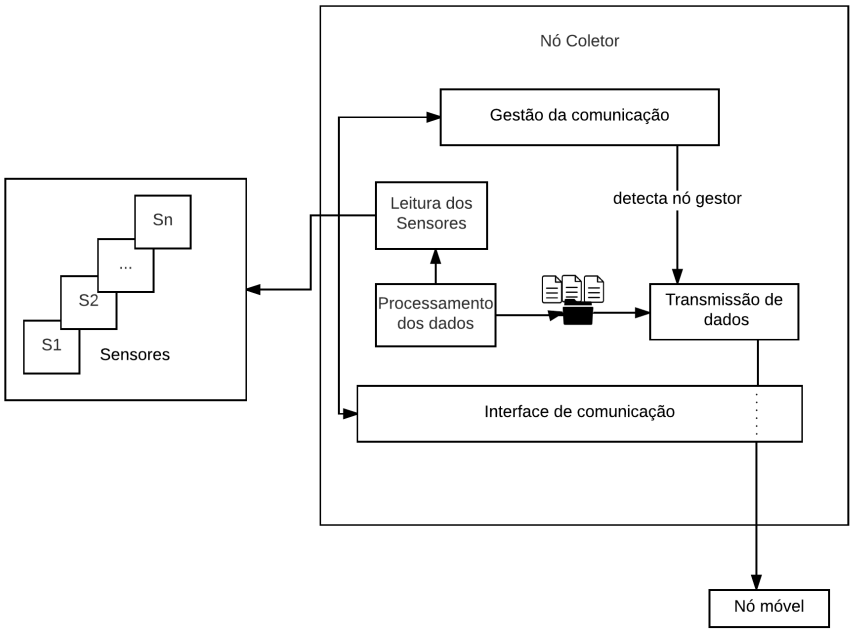

Figura 3.2: Diagrama de software do nó coletor

Listagem 3.1: Ficheiro de Dados.

```
1                                    | Sensores
2 Data       | Hora     |ID no    |S1 | S2          | Sn
3 2017-4-18  | 16:10:10 | Node1   | - | 26.625000   | -
4 2017-4-18  | 16:20:10 | Node1   | - | 26.630000   | -
```

3.2.2 Nó móvel

O nó móvel faz a recepção de dados dos nós coletores e o envio para o nó gestor. A figura 3.3 apresenta em detalhe a arquitetura de software deste nó. É composto por um módulo de gestão de comunicação e por uma interface de comunicação, à semelhança do nó coletor, e por módulos de leitura e escrita de dados. A sua função é apenas a de receber os ficheiros gerados pelos nós coletores e entregá-los no nó gestor após existir comunicação entre os mesmos, como ilustrado na figura 3.3. Sendo o único nó que se conecta com ambos os tipos de nós, é através do módulo de gestão da comunicação que o nó móvel decide se vai enviar ou receber dados, tendo em conta se a conexão é feita com o nó coletor ou com o nó gestor. Para tal ser possível, este nó tem que conhecer os endereços correspondentes a cada um. Quando se conecta a um nó coletor, está preparado para receber os ficheiros que este possui e no caso em que encontra o nó gestor, envia todos os ficheiros recebidos desde a última conexão. Para tal, o nó móvel tem uma diretoria conhecida na qual são colocados os

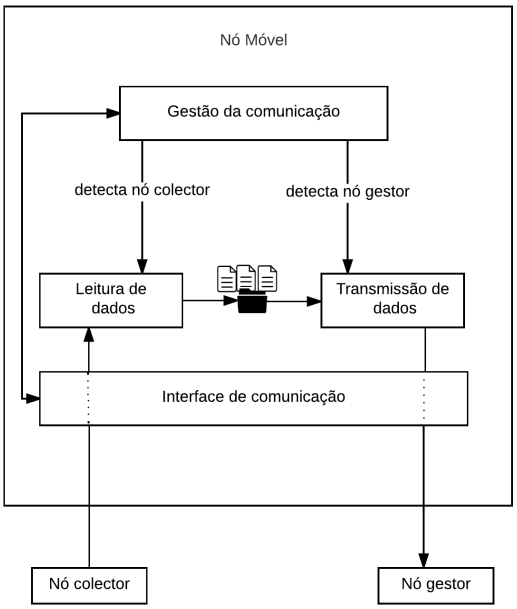

Figura 3.3: Diagrama de software do nó móvel

ficheiros. Quando a conexão é entre o nó coletor e o nó móvel, a pasta é considerada *Inbox*, e no caso de conexão com o nó gestor, é vista como *Outbox*. Na situação em que a pasta é considerada *Inbox*, os ficheiros provenientes do nó coletor são transferidos para a mesma, enquanto na situação da pasta *Outbox*, todos os ficheiros presentes na mesma são enviados para o nó correspondente. Desta forma, o nó sabe que comportamento deve ter em cada uma das situações.

3.2.3 Nó gestor

O nó gestor é o responsável pelo tratamento dos dados constantes no ficheiro gerado pelos nós coletores. A figura 3.4 ilustra a arquitetura de software deste nó, composta pelos módulo de gestão da comunicação e pela interface de comunicação, como nos restantes nós, bem como por um módulo de leitura de dados. Para além destes, contém um módulo de processamento de informação e uma base de dados. Este nó recebe os ficheiros do nó móvel e insere-os na base de dados, que pode ser acedida neste mesmo nó através de uma página local. Da mesma forma que o nó móvel, este recebe os ficheiros numa pasta. Através de um *cron job*, um serviço que executa comandos agendados, é executado um *script* que processa os dados e faz a sua inserção na base de dados local periodicamente. Quando os dados são inseridos na base de dados ficam disponíveis para visualização através da página, cujo endereço é conhecido pelo utilizador. Os dados terão que estar no formato definido à priori nos nós coletores, como mostrado na figura 3.1, para que sejam processados de forma cor-

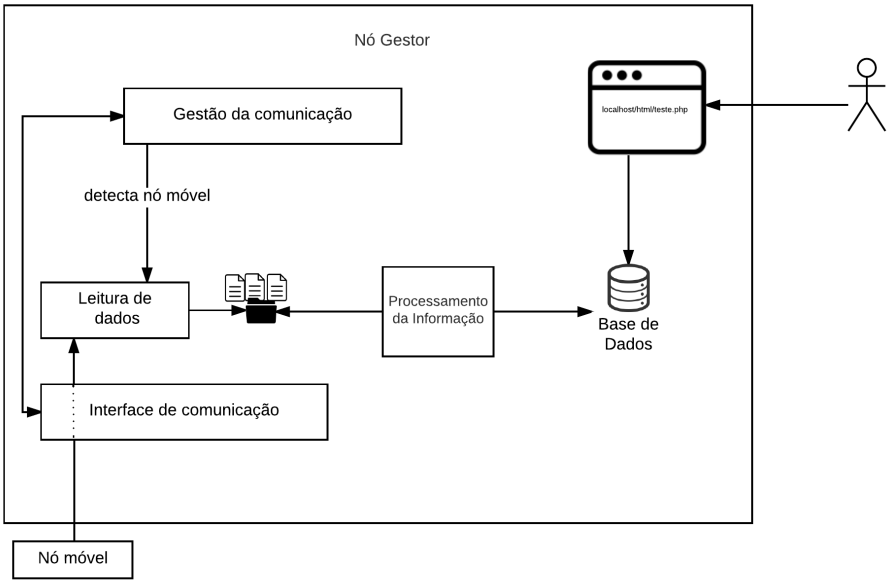

Figura 3.4: Diagrama de software do nó gestor

reta. Caso a formatação não esteja correta é inserida uma linha apenas com o nome do nó, mas sem quaisquer valores, sendo o erro da responsabilidade do nó coletor. Na mesma base de dados são inseridos os dados correspondentes a todos os nós coletores, identificados pelo nome do próprio nó. Estes dados são inseridos na base de dados pela ordem em que são recebidos no nó gestor.

4

Implementação

Conteúdo

Tendo como base a arquitetura apresentada no capítulo anterior, foi implementado um sistema de teste para comprovar a viabilidade da solução. Neste capítulo serão apresentados os detalhes da implementação, as decisões tomadas e os resultados obtidos.

4.1 Rede

4.1.1 Rede sem fios

Como tecnologia de acesso foi escolhido *WiFi*, definido pelo standard IEEE 802.11. O *WiFi* é uma tecnologia de rede sem fios que permite a comunicação de dispositivos através de um sinal propagado pelo ar. Dentro das possibilidades para redes *WiFi* comuns, existem as mais variadas aplicações, tais como o acesso à Internet em locais públicos, bastando para tal ligar-se a um ponto de acesso disponível. Para que os nós comuniquem entre si sem ser necessário um ponto de acesso, estes estão configurados em modo ad-hoc, que é um tipo de rede local construído pelos próprios dispositivos espontaneamente. Os próprios nós coordenam o envio e receção das mensagens entre si.

A rede ad-hoc criada tem o nome ”dsensor”, e os nós utilizam IP's estáticos para que apenas comuniquem entre si, e não com o exterior, reduzindo os riscos de segurança na rede. Na rede ”dsensor”foi escolhida a gama 10.0.0.0/24, sendo que se trata de uma rede privada e estes correspondem aos primeiros endereços definidos para redes privadas. Embora os endereços 10.0.0.0 sejam endereços classe A, tipicamente utilizados com máscara /8, nos ambientes de teste existiam diversos dispositivos com endereços classe C (gama 192.168.1.0/24) e classe B (172.16.0.0/12), e por uma questão de simplificação foram utilizados os endereços da gama 10.0.0.0/24. Esta gama de endereços permite a ligação de até 254 nós. Contudo, caso seja necessário adicionar mais nós à rede, deve-se redimensionar a máscara de rede de acordo com o necessário.

Na interface wireless de cada um dos nós intervenientes na rede corre o protocolo IBR-DTN.

4.1.2 Protocolo IBR-DTN

A par da rede ad-hoc configurada em todos os intervenientes da rede, estes correm o protocolo DTN desenvolvido pelo IBR, e assim denominado IBR-DTN. Este protocolo é implementado recorrendo a um *daemon*, um programa que corre como processo de *background* em cada nó, não necessitando de intervenção do utilizador para além da configuração inicial de acordo com as características pretendidas para a rede. A listagem apresentada no apêndice A é exemplo do ficheiro de configuração do nó móvel da rede. Na configuração deste protocolo é definido o nome do nó correspondente, que tem a forma dtn://nodeName.dtn, através do qual pode ser encontrado pelos restantes nós da rede. O protocolo base utilizado em DTN, o BP, necessita de um *Convergence-Layer Adapter*, como referido na

secção 2.2.3. Neste caso específico é utilizado o TCP Convergence-Layer (TCP-CL), que utiliza TCP para enviar e receber os bundles. É estabelecida uma conexão TCP entre os nós e posteriormente é trocado um *header* que define os parâmetros da ligação TCP-CL. É ainda enviado o identificador do *endpoint*, que neste caso será o identificador do nó, que é o próprio nome. Após esta configuração os nós conseguem trocar bundles entre si [8]. Tendo em conta que um dos nós está em constante movimento,o nó móvel, são enviados *discovery beacons* de 5 em 5 segundos para que cada nó se anuncie a novos possíveis vizinhos, e a conexão tem um *keep-alive timeout* de 10 segundos. Entre as diversas estratégias de encaminhamento possíveis, referidas na secção 2.2, foi escolhida a opção *default routing*, que efetua a entrega dos bundles aos vizinhos e aos nós estáticos. Desta forma é possível definir quais os nós que podem conectar-se à rede, e receber bundles. A definição dos nós estáticos é a presente nas linhas 70 a 92 da listagem A.1. A primeira linha de cada conexão estática refere o endereço IP do nó, seguida do porto que recebe os bundles, no qual o valor 4556 refere o porto por defeito para comunicação DTN , o nome do nó, o protocolo utilizado e por fim as *flags* que permitem conexão imediata ao nó sem ser necessária ligação à Internet.

4.2 Arquitetura de Hardware

Para testar a solução proposta foram utilizados dois nós coletores, um nó móvel e um nó gestor. Os nós colectores foram simulados recorrendo a micro-controladores Raspberry Pi 3 com o sistema operativo Raspbian Jessie, e com sensores de temperatura e humidade ligados através da interface GPIO do dispositivo. Para simular os nós móvel e gestor foram utilizados computadores portáteis com o sistema operativo Ubuntu 15.10. Para que a rede funcione de forma correta, todos os nós necessitam de um módulo WiFi configurável em modo ad-hoc. O nó gestor tem uma base de dados mySQL, um servidor local Apache e PHP instalados para que seja possível apresentar os dados de forma correta. Por questões de compatibilidade de processador, foram utilizados Raspberry Pi 3, visto terem um processador ARMv7 que não apresenta qualquer problema com a utilização do protocolo IBR-DTN, o que não acontece com o modelo 1B.

4.2.1 Nó colector

O nó coletor, sendo o responsável pela recolha dos dados do meio ambiente, tem uma posição fixa conhecida pelo utilizador. Foram utilizados dois nós coletores com posições distintas, e com sensores diferentes. O nó coletor 1, com identificador *node1*, recolhe a temperatura do meio. A figura 4.1 apresenta a montagem do sensor de temperatura de forma a efetuar a recolha. Foi usado um sensor de temperatura à prova de água, o DS18B20, que liga ao micro-controlador através dos pins GPIO do dispositivo [24]. Os fios representados a preto e vermelho correspondem ao *GROUND* e aos 5V, res-

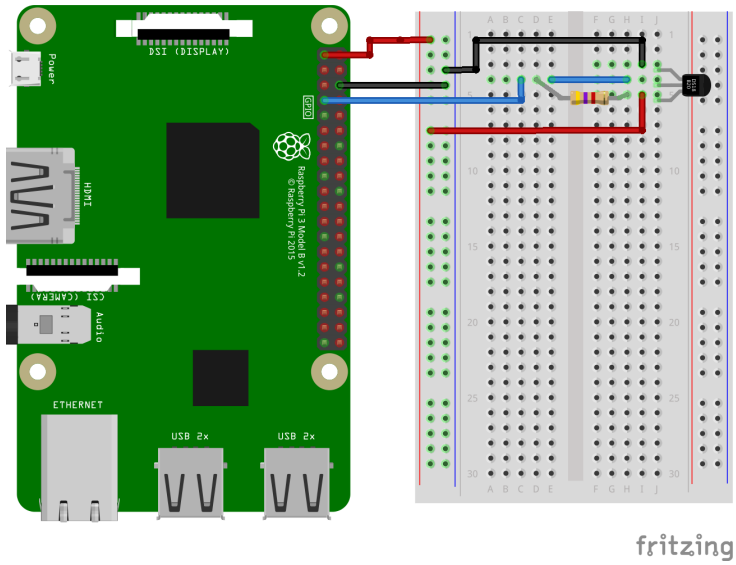

Figura 4.1: Hardware do nó coletor *node1*

petivamente, enquanto o fio representado a azul faz a ligação a uma porta de dados, através da qual é transmitido o valor lido pelo sensor.

O segundo nó coletor, *node2*, apresenta uma montagem semelhante, mas para um sensor que mede a humidade do meio ambiente. Para obter dados próximos de uma situação real, os nós coletores foram colocados a uma de distância aproximada de 135m, como se pode verificar pela figura 4.2. Embora estes nós, representados a vermelho, não comuniquem entre si, esta distância garante que existe desconexão entre ambos os coletores e o nó móvel durante o seu movimento.

Durante a implementação destes nós surgiram diversos problemas relativos à comunicação entre os mesmos e o nó móvel relacionados com as definições por defeito do sistema operativo. Ocorreram diversas falhas de conexão entre os nós por estes entrarem em modo suspensão, problema esse que foi resolvido alterando as definições do sistema operativo. Para simplificar todo o processo de configuração e teste, os micro-controladores funcionaram ligados à corrente e com monitor, teclado e rato. Num ambiente de utilização real funcionaria apenas o micro-controlador alimentado por uma bateria devido à falta de infraestrutura e à ausência de necessidade de efetuar alterações e monitorizações constantes como foi o caso.

4.2.2 Nó móvel

O nó móvel, com identificador *node253* e representado a verde na figura 4.2, foi simulado por um computador portátil por questões de simplificação em termos de visualização das ligações em tempo real. Poderia ter sido utilizado um Raspberry Pi semelhante ao dos nós coletores, mas para tal seria necessária uma bateria para o alimentar. Este nó faz a rota entre os nós coletores pelo trajeto assina-

Figura 4.2: Localização dos nós

lado na figura 4.2. O seu ponto de partida foi junto ao *node2*, seguiu para o *node1*, e fez o percurso inverso para se conectar ao nó gestor, localizado no ponto inicial. Para que o ambiente de teste fosse o mais próximo do real, embora tenha sido utilizado um computador portátil, foi ligado um *dongle* WiFi com características semelhantes ao módulo WiFi do Raspberry Pi para que não houvesse diferença significativa em relação ao alcance do nó face à situação real.

4.2.3 Nó gestor

O nó gestor, *node3* a azul na figura 4.2, é também representado por um computador portátil, localizado no mesmo sítio que o nó coletor *node2*.Tal como os nós coletores, também este tem uma localização fixa e conhecida, visto que é o nó utilizado pelo utilizador para consultar os dados recolhidos. Foi utilizado um computador portátil com o sistema operativo Ubuntu 15.04, no qual foi instalado um servidor Apache para permitir a execução de código PHP para apresentar os dados na página local. Foi criado um script PHP que faz o processamento dos dados recebidos e a sua inserção na base de dados MySQL armazenada no próprio nó, recorrendo a um *cronjob* que o executa a cada segundo. Este script lê os ficheiros por ordem crescente, começando no nó com o identificador mais baixo, sendo que os dados são assim inseridos na base de dados de acordo com essa ordem.

Tabela 4.1: Exemplo dos dados recolhidos

Data	Hora	Nó	Temperatura	Humidade
2017-04-18	18:09:12	Node2	29.184	10.03
2017-04-18	18:10:12	Node2	29.234	10.01
2017-04-18	18:11:12	Node2	29.312	10.10
2017-04-18	18:12:12	Node2	29.417	10.07
2017-04-18	18:09:14	Node1	30.187	09.27
2017-04-18	18:10:14	Node1	30.375	09.81
2017-04-18	18:11:14	Node1	30.312	09.45

4.3 Configuração dos nós

Os ficheiros de configuração dos nós a nível do protocolo IBR-DTN, referidos na secção 4.1.2 e apresentados no anexo B são semelhantes para os todos os nós da rede, mudando apenas o identificador do nó a ser configurado e as ligações aos restantes. Os nós coletores e gestor apenas se conectam ao nó móvel *node253*, e este último conecta-se a todos visto que é o que faz a receção e entrega dos dados entre ambos os tipos de nó. As listagens B.1, B.2 e B.3 apresentam as alterações aos ficheiros de cada nó, relativamente ao ficheiro de configuração do nó móvel referido anteriormente.

A aquisição dos dados dos sensores ligados aos nós coletores foi efetuada com recurso à linguagem de programação C, utilizando bibliotecas específicas para acesso e controlo dos pins GPIO do Raspberry Pi. A biblioteca WiringPi permite programar o Raspberry Pi da mesma forma que ao Arduíno, simplificando o acesso e permitindo a ligação e controlo dos pins GPIO. Isto permite a leitura dos dados recolhidos pelo sensor, que são escritos no momento numa nova linha do ficheiro que será enviado aquando conexão com o nó móvel.

O nó gestor, ao receber os ficheiros de cada um dos nós coletores lê o seu conteúdo e insere as linhas numa base de dados *mysql* local. Esta base de dados contém os mesmos campos do ficheiro criado pelos nós coletores, e através de uma página PHP local os dados são disponibilizados ao utilizador, como mostrado na tabela 4.1. Estes dados foram adquiridos com intervalos de um minuto, e no momento podemos verificar que a humidade detetada pelo *node2* foi 0%, e no *node1* a temperatura próxima de 30ºC. Tendo em conta que a chegada dos ficheiros ao nó gestor não acontece de forma periódica, visto estar dependente da rota do nó móvel, os ficheiros já presentes no nó gestor são lidos a cada minuto e caso exista alguma alteração, as novas linhas são adicionadas à base de dados.

4.4 Testes

Os testes da implementação tiveram lugar no Instituto Superior Técnico, *campus* do Taguspark. A figura 4.2 mostra a localização de cada um dos nós no interior do edifício. Para além da distância percetível pela imagem, o nó *node1* encontra-se no piso 0 do edifício, enquanto o nó *node2* está no

Tabela 4.2: Tamanho e tempo de envio de um ficheiro

Tamanho (kB)	Tempo (s)
[0-100]	<1
[100-500]	1<t<2
[500-1000]	2<t<3
[1000-3000]	3<t<4
[3000-5000]	4<t<5
[5000-7000]	5<t<7
>7000	t>7

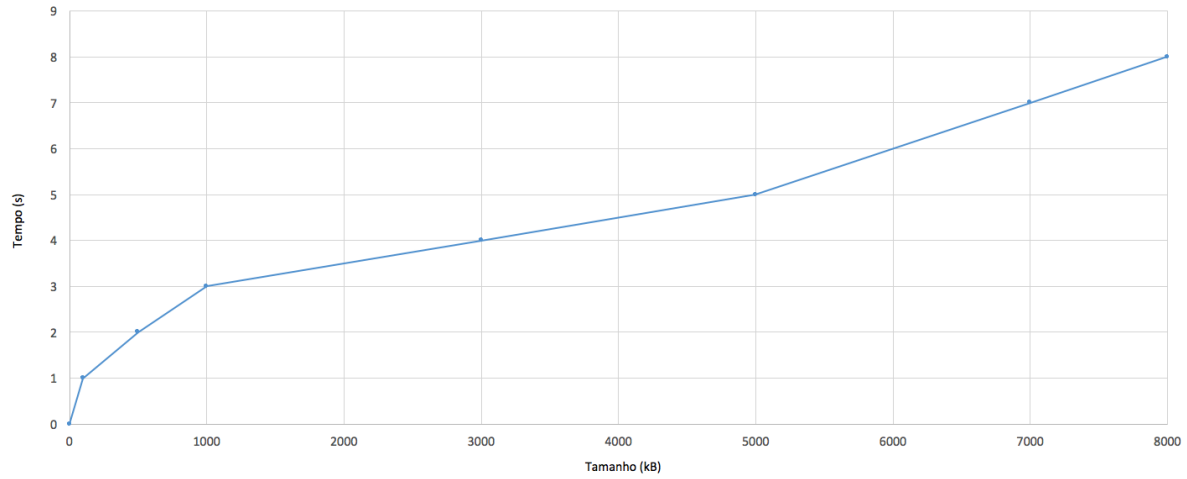

Figura 4.3: Relação tempo/tamanho do ficheiro

piso 2.

Como os nós da rede não estão ligados à Internet, pode existir diferença na hora dos vários intervenientes. Tendo em conta o tipo de dados recolhidos, não é considerado problemático pois as diferenças são de poucos minutos e a temperatura ou humidade, neste caso, não variam significativamente nesse intervalo de tempo.

Para efetuar o teste, o sistema foi simulado na integra. Os nós coletores iniciaram a recolha de dados e o nó móvel começou o seu movimento. Estes testes decorreram durante vários dias, a horas diferentes e com periodicidade de recolhas dos próprios nós, e de comunicação entre o nó móvel e os coletores distintas. Os dados utilizados para calcular o tempo de transmissão dos ficheiros foram recolhidos dos ficheiros *log* dos nós intervenientes, como o excerto apresentado na listagem C.1.

Foram efetuados testes com ficheiros de tamanhos variados para avaliar a taxa de entrega dos pacotes e a velocidade de transmissão dos dados. A transferência de ficheiros do mesmo tamanho foi efetuada 20 vezes, e a média do tempo de transmissão calculada (tabela 4.2).

O alcance WiFi dos nós é de cerca de 25 metros, pelo que, a uma velocidade de 2m/s o nó móvel demora cerca de 12 segundos a perder a conexão. De acordo com as amostras recolhidas, um ficheiro

com cerca de 51 bytes demora menos de 1 segundo do momento em que é colocado na fila para envio, até que chega ao nó destino. O tempo gasto na conexão entre os nós é de 2 segundos, pelo que dispomos de cerca de 10 segundos para transmitir um ficheiro.

A tabela 4.2 e o gráfico da figura 4.3 apresentam a relação entre o tamanho dos ficheiros em kB e o tempo de transmissão do mesmo. Como os *logs* apenas apresentam o tempo no formato HH:MM:SS, não é possível calcular com exatidão o tempo de transferência para valores inferiores a um segundo. Deste modo, os tamanhos dos ficheiros bem como os tempos de transferência foram organizados em intervalos.

Pela análise do gráfico da figura 4.3, podemos concluir que para ficheiros de tamanho superior a 8000kB o tempo de envio fica próximo do limite do tempo de conexão entre os nós. Desta forma, tomando como limite um ficheiro de 8000kB podemos recolher cerca de 82000 amostras de 3 sensores distintos. Isto traduz-se em aproximadamente uma amostra a cada segundo durante 24h.

Para dados cuja alteração não é significativa de segundo para segundo, não é necessário fazer a recolha com esta periodicidade. Amostras recolhidas com uma periodicidade de 10 minutos durante 24h resultam num ficheiro com cerca de 8kB, que, de acordo com os tempos médios obtidos na tabela 4.2 demora menos de um segundo a transmitir. Desta forma, conseguimos garantir uma taxa de entrega de 100%, salvo casos de interrupção da conexão devido a problemas de software ou hardware.

5

Conclusões

Conteúdo

5.1 Conclusão

O objetivo principal deste trabalho visa a aplicação das DTN à recolha de dados de sensores, com utilização de *drones*. A pesquisa apresentada neste documento, bem como a arquitetura escolhida serviram de base à aplicação deste projeto de forma funcional e versátil, com a utilização de dispositivos de baixo-custo e *software open-source*. Ainda que a sua concretização tenha sido feita em ambiente controlado, as suas características permitem a sua aplicação nos diversos contextos, com alteração a nível dos sensores utilizados, caso seja necessário.

Com os testes efetuados à arquitetura desenhada e implementada, comprova-se a viabilidade da solução tanto no cenário testado, como no cenário hipotético que utiliza *drones* para movimentar o nó móvel.

As limitações encontradas nesta solução estão relacionadas com a dimensão dos ficheiros transferidos, uma vez que a velocidade do *drone* pode ser superior à atingida nos testes, o que faz com que os nós coletores estejam ao alcance do nó móvel durante menos tempo. Esta limitação pode ser mitigada com a alteração da periodicidade de recolha de dados por parte dos nós coletores e/ou pela recolha mais frequente por parte do nó móvel.

Em suma, a solução apresentada é uma opção viável e de baixo-custo, aplicável nos mais variados contextos.

5.2 Trabalho Futuro

Futuramente, a este sistema podem ser adicionadas mais funcionalidades, tal como melhoradas algumas características. O controlo do movimento do *drone* pode ser incorporado no nó gestor da rede através da página de consulta dos dados, e ainda ser desenvolvida uma aplicação móvel para acesso a estas funcionalidades. Outro ponto que poderá ser explorado prende-se com a alimentação dos nós coletores. Poderá ser usada alimentação com recurso a energias renováveis, através de painéis solares, e assim minimizar o consumo de energia gasto pelos nós, permitindo que o número de nós e sensores seja aumentado sem que cause um grande impacto no custo da energia.

Bibliografia

[1] H. Ntareme and M. Zennaro, “Delay Tolerant Network on smartphones : Applications for communication challenged areas,” *Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition*, pp. 14:1—-14:6, 2011.

[2] R. J. D’Souza and J. Jose, “Routing Approaches in Delay Tolerant Networks: A Survey,” *International Journal of Computer Applications IJCA*, vol. 1, no. 17, pp. 9–15, 2010. [Online]. Available: http://www.ijcaonline.org/archives/volume1/number17/370-557

[3] J. Shen, S. Moh, and I. Chung, “Routing Protocols in Delay Tolerant Networks : A Comparative Survey,” pp. 1577–1580, 2008.

[4] P. R. Pereira, A. Casaca, J. J. P. C. Rodrigues, V. N. G. J. Soares, J. Triay, and C. Cervello-Pastor, “From Delay-Tolerant Networks to Vehicular Delay-Tolerant Networks,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1166–1182, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6007076

[5] E. Jones and P. Ward, “Routing strategies for delay-tolerant networks,” *... to ACM Computer Communication Review (CCR ...*, 2006. [Online]. Available: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Routing+Strategies+for+Delay-Tolerant+Networks{#}0

[6] S. Ali, J. Qadir, and A. Baig, “Routing protocols in Delay Tolerant Networks - a survey,” *Emerging Technologies ICET 2010 6th International Conference on*, pp. 70–75, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=5638377

[7] K. Scott, “Bundle Protocol Specification,” https://tools.ietf.org/html/rfc5050, [Accessed 28-Maio-2016].

[8] “DTN TCP Convergence-Layer Protocol,” https://tools.ietf.org/html/rfc7242, [Accessed 5-Abril-2017].

[9] S. Farrell, “A Delay and Disruption Tolerant Transport Layer Protocol,” no. September, 2008.

[10] P. Mcdonald, D. Geraghty, I. Humphreys, and S. Farrell, “Sensor Network with Delay Tolerance (SeNDT),” pp. 1333–1338, 2007.

[11] M. Ramadas, “Licklider Transmission Protocol Specification,” https://tools.ietf.org/html/rfc5326, [Accessed 28-Maio-2016].

[12] Z. Feng and K.-W. Chin, “A Survey of Delay Tolerant Networks Routing Protocols,” 2012. [Online]. Available: http://arxiv.org/abs/1210.0965

[13] S. Guo, M. H. Falaki, E. A. Oliver, S. U. Rahman, A. Seth, M. A. Zaharia, U. Ismail, and S. Keshav, “Design and Implementation of the KioskNet System.”

[14] S. Guo, M. H. Falaki, E. a. Oliver, S. Ur Rahman, A. Seth, M. a. Zaharia, and S. Keshav, “Very low-cost internet access using KioskNet,” *SIGCOMM Computer Communication Review*, vol. 37, no. 5, pp. 95–100, 2007. [Online]. Available: http://dx.doi.org/10.1145/1290168.1290181

[15] M. Zarafshan-araki and K.-w. Chin, “TrainNet : A transport system for delivering non real-time data,” vol. 33, pp. 1850–1863, 2010.

[16] T. Instruments, “LM35 Precision Centigrade Temperature Sensors,” http://www.ti.com/lit/ds/symlink/lm35.pdf, [Accessed 31-Maio-2016].

[17] C. Sense, “Wireless Temperature Sensor,” https://www.connectsense.com/wireless-temperature-sensor, [Accessed 31-Maio-2016].

[18] “Arduino,” https://www.arduino.cc, [Accessed 31-Maio-2016].

[19] “Raspberry Pi,” https://www.raspberrypi.org, [Accessed 31-Maio-2016].

[20] “armStone,” https://www.fs-net.de/en/products/armstone/, [Accessed 31-Maio-2016].

[21] “Parrot Drones,” http://www.parrot.com/, [Accessed 31-Maio-2016].

[22] “DJI,” http://www.dji.com/, [Accessed 31-Maio-2016].

[23] S. Schildt, J. Morgenroth, L. Wolf, S. Schildt, J. Morgenroth, and L. Wolf, “Electronic Communications of the EASST Workshops der wissenschaftlichen Konferenz Kommunikation in verteilten Systemen 2011 (WowKiVS 2011) IBR-DTN : A lightweight , modular and highly portable Bundle Protocol implementation IBR-DTN : A lightweight , modular and highly portable Bundle Protocol implementation,” vol. 37, 2011.

[24] “DS18B20 Sensor,” https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf, [Accessed 10-Julho-2017].

A

Anexo

Listagem A.1: Ficheiro de configuração.

```
# ########## NODE 253 ########## #

local_uri = dtn://node253.dtn
logfile = /var/log/ibrdtn/ibrdtn.log
api_interface = any
api_port = 4550

# ########## storage configuration ########## #

# define a folder for temporary storage of bundles
blob_path = /tmp

# define a folder for persistent storage of bundles
storage_path = /var/spool/ibrdtn/bundles

# defines the storage module to use
storage = default

# Defines, whether bundleSets are stored persistently in the storage path
use_persistent_bundlesets = no

# ########## convergence layer configuration ########## #

# Specify how often discovery beacons are sent.
discovery_interval = 5

# specify the discovery mechanism to use; 0 = DTN2 compatible discovery
discovery_version = 0

# To disable discovery announcements, set this option to zero.
discovery_announce = 1

# Enable crosslayer discovery
discovery_crosslayer = yes

# a list of names for convergence layer instances.
net_interfaces = lan0
```

```
38
39 # Try to connect to other nodes each x seconds.
40 net_autoconnect = 10
41
42 # configuration for a convergence layer named lan0
43 net_lan0_type = tcp
44 net_lan0_interface = wlan0
45 net_lan0_port = 4556
46
47 # NODELAY option in TCP disables the nagle algorithm, if set to yes (default).
48 tcp_nodelay = yes
49
50 # The bundles are split into chunks while they are transmitted over TCP.
51 tcp_chunksize = 4096
52
53 # Keep-alive time-out for connections
54 keepalive_timeout = 10
55
56 # ########## routing configuration ########## #
57
58 # routing strategy
59 # values: default | epidemic | flooding | prophet | none
60 routing = default
61
62 # forward bundles to other nodes (yes/no)
63 routing_forwarding = yes
64
65 # forward singleton bundles directly if the destination is a neighbor
66 routing_prefer_direct = yes
67
68 # static connections
69
70 ### node1.dtn ###
71 static1_address = 10.0.0.1
72 static1_port = 4556
73 static1_uri = dtn://node1.dtn
74 static1_proto = tcp
75 static1_immediately = yes
```

```
76 static1_global = no
77
78 ### node2.dtn ###
79 static2_address = 10.0.0.2
80 static2_port = 4556
81 static2_uri = dtn://node2.dtn
82 static2_proto = tcp
83 static2_immediately = no
84 static2_global = no
85
86 ### node3.dtn ###
87 static3_address = 10.0.0.3
88 static3_port = 4556
89 static3_uri = dtn://node3.dtn
90 static3_proto = tcp
91 static3_immediately = no
92 static3_global = no
```

B

Anexo

Listagem B.1: Ficheiro de configuração node 1

```
# ########## NODE 1 ########## #

local_uri = dtn://node1.dtn

.
.
.

# static connections

### node253.dtn ###
static3_address = 10.0.0.253
static3_port = 4556
static3_uri = dtn://node253.dtn
static3_proto = tcp
static3_immediately = no
static3_global = no
```

Listagem B.2: Ficheiro de configuração node 2

```
# ########## NODE 2 ########## #

local_uri = dtn://node2.dtn

.
.
.

# static connections

### node253.dtn ###
static3_address = 10.0.0.253
static3_port = 4556
static3_uri = dtn://node253.dtn
static3_proto = tcp
static3_immediately = no
```

```
static3_global = no
```

Listagem B.3: Ficheiro de configuração node 3

```
# ########## NODE 3 ########## #

local_uri = dtn://node3.dtn

.
.
.

# static connections

### node253.dtn ###
static3_address = 10.0.0.253
static3_port = 4556
static3_uri = dtn://node253.dtn
static3_proto = tcp
static3_immediately = no
static3_global = no
```

C

Anexo

Listagem C.1: Ficheiro *log* node 253

```
Thu May 11 19:37:01 2017 NOTICE QueueBundleEvent: New bundle queued [547843021.1]
   dtn://node253.dtn/outboxSender
Thu May 11 19:37:01 2017 NOTICE BundleEvent: bundle [547843021.1]
   dtn://node253.dtn/outboxSender received

Thu May 11 19:37:05 2017 NOTICE RequeueBundleEvent: Bundle requeued [547843021.1]
   dtn://node253.dtn/outboxSender
Thu May 11 19:37:06 2017 INFO DiscoveryAgent: DTN2 compatibility mode
Thu May 11 19:37:06 2017 INFO DiscoveryAgent: DTN2 compatibility mode
Thu May 11 19:37:06 2017 NOTICE ConnectionEvent: connection setup dtn://node1.dtn
Thu May 11 19:37:06 2017 NOTICE ConnectionEvent: connection setup dtn://node2.dtn
Thu May 11 19:37:06 2017 NOTICE ConnectionEvent: connection setup dtn://node3.dtn
Thu May 11 19:37:06 2017 NOTICE ConnectionEvent: connection up dtn://node2.dtn

Thu May 11 19:37:08 2017 NOTICE ConnectionEvent: connection up dtn://node3.dtn

Thu May 11 19:37:09 2017 WARNING TCPConnection: connection to dtn://node1.dtn failed

Thu May 11 19:37:16 2017 NOTICE RequeueBundleEvent: Bundle requeued [547843021.1]
    dtn://node253.dtn/outboxSender
Thu May 11 19:37:17 2017 NOTICE ConnectionEvent: connection setup dtn://node1.dtn

Thu May 11 19:37:18 2017 NOTICE BundleCore: singleton bundle delivered: [547843021.1]
    dtn://node253.dtn/outboxSender
Thu May 11 19:37:18 2017 NOTICE TransferCompletedEvent: transfer of bundle [547843021.1]
    dtn://node253.dtn/outboxSender to dtn://node3.dtn completed

Thu May 11 19:37:18 2017 NOTICE BundleEvent: bundle [547843021.1]
    dtn://node253.dtn/outboxSender deleted
```