Armazenamento e disponibilização de dados oriundos de redes de sensores na nuvem

Geymerson S. Ramos, André L. L. Aquino

[1]Grupo de Pesquisa SensorNet-UFAL
Instituto de Computação – Universidade Federal de Alagoas
Maceió – AL – Brasil

geymerson.r@gmail.com, alla.lins@gmail.com

Resumo. *Este artigo discorre sobre os estudos em andamento que visam a integração de redes de sensores sem fio e computação em nuvem. Ao longo do texto, explicita-se o atual estado da pesquisa, ferramentas utilizadas tais como sensores, a programação aplicada, sobre o atual banco de dados, sobre a estrutura de nuvem que pretendemos utilizar e, ao final, resultados obtidos até então e expectativas futuras da pesquisa.*

1. Introdução

Internet das coisas (IoT - Internet of Things) [Atzoria et al. 2010], Big Data [Emani et al. 2015] e, Redes de Sensores Sem Fio (RSSF) [Akyildiz et al. 2002] são algumas das atuais vertentes do campo computacional. Impulsionadas pelo rápido desenvolvimento tecnológico e maiores necessidades de conectividade, as supracitadas alavancam o meio corporativo, científico e medicinal. O entendimento e expertise nessas áreas passaram a ser de grande demanda e oferecem vastas oportunidades àqueles que gostam de aventurar-se em meios ainda timidamente explorados, e que há não muito tempo, eram considerados quiméricos e intangíveis.

Uma RSSF é constituída por um conjunto de nós sensores com capacidade de armazenamento, processamento, recebimento e transmissão de dados e que possuem por finalidade monitorar eventos, tais como: temperatura, pressão, umidade, posição de objetos etc. Estas redes possuem diversas aplicações nas áreas ambiental, médica, militar, transporte inteligente entre outras.

Neste artigo, discutimos os estudos da pesquisa em andamento que visa posteriormente migrar do banco de dados não relacional em uso para uma estrutura mais complexa em nuvem, que será a princípio mantida em um cluster gerenciado via OpenStack [OpenStack 2015], no Laboratório de Computação Científica e Visualização (LCCV), situado na universidade, isso suprirá as demandas de Infrastructure as a Service (IaaS). Necessidades relacionadas a Software as a Service (SaaS) e Platform as a Service (PaaS) serão mencionadas futuramente conforme as melhores soluções sejam estudadas e demonstrem-se necessárias. Esclarecimentos sobre IaaS, SaaS e PaaS em [Mell and Grance 2010].

Os sensores utilizados no nosso estudo são os motes da plataforma MicaZ [MEMSIC 2015], que possui o sistema operacional TinyOS que é programável pela linguagem NesC [Davis and Gay 2009]. Essa linguagem é uma derivação da linguagem C, portanto, conhecimentos desta podem ser exportados para NesC. Para realizar a coleta e disseminação dos dados, utilizamos o componente `Collection` do TinyOS.

`Collection` é um protocolo de disseminação baseado em árvore onde um ou mais motes servem como raizes, que são os destinos finais do dado, que a partir daí pode ser consumido por outros serviços, como um computador local. Os demais motes fazem retransmissão de informação de forma que o fluxo dos dados é dos motes para uma raiz.

Outro recurso utilizado, na fase de configuração da rede, foi o protocolo `Deluge` [Hagedorn et al. 2008], que é um recurso do TinyOS para oferecer *Over The Air Programming* (OTAP) para a reprogramação *on-line* dos sensores. O `Deluge` é um protocolo de reconfiguração das aplicações binárias presentes nos sensores, onde as aplicações são disseminadas pela rede para cada um dos nós e assim reconfiguradas.

Para o armazenamento dos dados, utilizamos o MongoDB [MongoDB 2015], uma vez que se apoia fortemente nas técnicas de MapReduce [Lee et al. 2011]. O projeto aqui apresentado é de caráter operacional, todavia, é interessante que estudos e análises possam ser feitos em cima dos dados coletados e, portanto, o caráter analítico é também essencial. O MongoDB é de natureza operacional, no entanto, dá suporte a ferramenta Hadoop [Tan et al. 2011], que tem por finalidade realizar operações de caráter analítico em conjuntos de dados, e consequentemente, cria uma aplicação mais funcional.

Dentre as diversas contribuições do nosso trabalho, podemos destacar: montagem de uma RSSF para o monitoramento de prédios inteligentes; criação de um banco de dados dedicado ao armazenamento de informações advindas dos sensores distribuídos no prédio; e disponibilização dos dados para as mais diversas aplicações e/ou pesquisa.

2. Aplicação desenvolvida

A aplicação consiste na utilização de uma RSSF com 12 sensores distribuídos em um prédio para monitorar a temperatura. Os sensores utilizados são os da plataforma MicaZ com o TinyOS 2.1.2. Como a nossa rede suporta OTAP a configuração e instalação dos nós é feita pela rede. Os sensores coletam os dados e os enviam, de forma *ad-hoc*, para estação base por intermédio do protocolo disseminação `Collection`. Esse protocolo é necessário, pois nem todos os sensores possuem conexão direta com a estação base, logo é preciso rotear os dados. A implementação dos nós sensores contemplam tanto os nós folha como a estação base. Para os nós folha temos que implementar os seguintes códigos NesC:

- `Sense.h`: Onde define-se a estrutura que representa a unidade de dados que é enviada pela rede através dos sensores, os `packets`. Nos `packets`, são inseridos os campos de dados desejados e possíveis de extrair dos sensores, além de outros valores importantes durante a execução da aplicação.
- `SenseC.nc`: Onde são declaradas as interfaces utilizadas pelo componente e em seguida é feita a implementação, sendo que, as interfaces são inseridas no bloco de código `module`, que caracteriza a assinatura do componente e tem o mesmo nome deste. Os eventos devem obrigatoriamente ser implementados e tipicamente informam que um comandos foi finalizado com sucesso, ou não. O bloco de código `implementation` é onde é feita a implementação.
- `SenseAppC.nc`: Onde é feita a etapa de ligação dos componentes da aplicação. Uma vez que um componente é criado em NesC, é possível utilizar este em outros componentes já existentes. Então, na implementação de `SenseAppC.nc`, primeiramente são declarados os componentes utilizados e,

subsequentemente, são feitas as conexões entre os mesmos. Tudo é realizado no bloco `implementation`.

Para a estação base temos que implementar os códigos NesC: `SenseBaseC.nc` e `SenseBaseAppC.nc` que seguem a mesmo lógica e função dos códigos `SenseC.nc` e `SenseAppC.nc`, respectivamente.

Os nós foram posicionados em 12 diferentes pontos do prédio com diferentes salas que variam, entre 5 e 25 m^2, as salas menores ficaram com 1 sensor e as maiores, 3 salas, com 2 sensores. O sensor mais distante da estação base ficou a aproximadamente 30 m de distância e, devido a atenuação do sinal nas paredes, foi necessário fazer o roteamento *ad-hoc*. Os sensores realizam o monitoramento de forma contínua e enviam os dados a cada segundo.

A estação base, conectada a um computador via USB, envia os dados coletados para a aplicação de integração que faz a interface e armazena os dados no MongoDB. O MongoDB 3.0 foi instalado numa máquina com 4 GB de memória, processador Intel Core 2 Quad CPU Q6600 @ 2.40GHz x 4 e disco rídigo de 100 GB, com sistema operacional Linux Ubuntu 14.04 LTS 64 bits.

A aplicação de integração permite a integração das RSSFs com o MongoDB. Desenvolvemos uma interface que gerenciará os dados dos sensores no computador. A partir desta é possível visualizar, armazenar ou realizar qualquer outra ação apropriada com conteúdo dos `packets` gerados na RSSF. Com isso, os dados são armazenados no MongoDB para que possam ser consultados posteriormente sempre que necessário.

Desenvolvemos uma classe Java, `Sense.java` que faz a inserção dos dados dos sensores no MongoDB. Esta classe é executada sempre que um novo `packet` é recebido. Os dados coletados são armazenados de acordo com a seguinte organização:

```
"nodeID": "nodeNumber",
"sensirion_temp": "Sensirion sensor temperature  data",
"sensirion_hum": "Sensirion sensor humidity data",
"intersema_temp": "Intersema sensor temperature data",
"intersema_press": "Intersema sensor pressure data",
"infrared_light": "infrared light data",
"light": "visible light data",
"accel_x": "x axis acceleration data",
"accel_y": "y axix acceleration data",
"voltage": "voltage data",
"country": "my country",
"estate": "my estate",
"city": "my city",
"latitude": "measurement's latitude",
"longitude": "measurement's longitude",
"date": "current date"
```

Esta estrutura se adéqua ao formato de documento que MongoDB utiliza para armazenar os dados. Este tipo de arquivo é chamado de *BSON*, que são representações binárias de documentos *Javascript Object Notation* (JSON). Os Arquivos *BSONs* são mais abrangentes em quantidade de tipos de dados com relação aos *JSONs*.

De posse desses dados efetuamos o armazenamento. Para isso, ao receber uma mensagem, verificamos se é compatível com uma instância da classe `SenseMsg.java`, que é compatível com nossa mensagem em NesC. Os dados do `packet` são resgatados e processados. Após isso, uma conexão com o banco de dados na máquina local é realizado.

Realizada a conexão, é necessário escolher a base de dados e em seguida selecionar em qual coleção de dados encontram-se os dados a serem acessados ou salvos.

O documento a ser inserido no banco de dados é criado com o campo `nodeID`, que contém o número do nó emissor. Os demais campos são adicionados e, de forma implícita, o MongoDB criará um outro campo `_id` no documento, este último representa um identificador único de cada documento inserido. MongoDB permite que várias bases de dados sejam criadas em uma mesma máquina, cada base de dados pode conter mais de uma coleção, nas coleções são armazenados os dados (*Documents*). Este tipo de estrutura permite boas práticas de organização para tipos de sensores diversificados.

Considerando que a nossa RSSF está operacional e alimentando o MongoDB, podemos realizar diferentes consultas a esses dados. Para exemplificar a utilização de nossa aplicação considere o seguinte trecho de código Java:

```
1  ...
2  Date fromDate = new Date(1444867200000L);
3  Date toDate = new Date(1444903199000L);
4  BasicDBObject query = new BasicDBObject();
5  query.put("nodeID", 7);
6  query.put("date", BasicDBObjectBuilder.start("$gte",fromDate).add("$lte",toDate).get());
7  DBCursor cursor = coll.find(query);
8  ...
9  while(cursor.hasNext()) {
10     DBObject document = cursor.next();
11     myList.add(document.get("sensirion_temp").toString());
12 }
13 ...
```

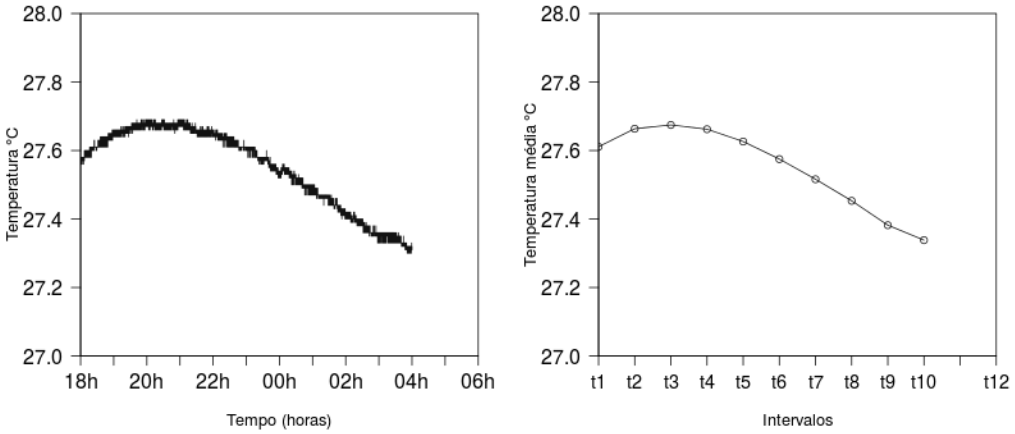
Os valores 1444867200000L e 1444903199000L representam, respectivamente, 21:00:00 do dia 14-10-2015 e 06:59:59 do dia 15-10-2015 em milisegundos (Linhas 2 e 3) no padrão UTC, isso implica três horas adicionais de direfença do horário local. Ou seja, estamos pegando todas as informações armazenadas no MongoDB, nesse intervalo de tempo (linha 6), pelo sensor nodeID = 7 (Linha 5), escolhido arbitrariamente. Para pegar especificamente os dados da temperatura (rotulado como "sensirion_temp") precisamos iterar (Linhas 9-12) sobre os dados obtidos e armazenar seus resultados em uma lista (Linha 11).

A Figura 1 apresenta os gráficos referentes aos dados coletados nesse trecho de código. A Figura 1(a) apresenta todas as temperatura medidas pelo nó 7 ao longo do tempo; e a Figura 1(b) apresenta a média das temperaturas medidas a cada hora.

3. Conclusão e trabalhos futuros

Foi desenvolvida uma aplicação que permite que vários sensores distribuídos pelo edifício enviem dados coletados deste ambiente, e que tais dados sejam armazenados. Estes dados podem ser consultados e utilizados nas mais diversas aplicações. Algumas aplicações em potencial são veículos que possuem computadores on-board, sistemas de segurança e diversos outros dispositivos.

Como trabalhos futuros, pretendemos utilizar uma nuvem pública para armazenar os dados coletados por nossa rede; disponibilizaremos os dados para a comunidade científica; implementaremos uma aplicação que combina os dados dos sensores com informações de redes sociais; e implementaremos um módulo gateway na RSSF que permite a integração de diferentes tipos de sensores.



(a) Leituras dos sensores (b) Média das leituras

Figure 1. Consulta de temperatura

References

[Akyildiz et al. 2002] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine*.

[Atzoria et al. 2010] Atzoria, L., Ierab, A., and Morabitoc, G. (2010). The internet of things: A survey. *Computer Networks - Journal - Elsevier*.

[Davis and Gay 2009] Davis, P. and Gay, D. (2009). *TinyOS Programming*. Cambridge University Press.

[Emani et al. 2015] Emani, C. K., Cullot, N., and Nicolle, C. (2015). Understandable big data: A survey. *Elsevier*.

[Hagedorn et al. 2008] Hagedorn, A., Starobinski, D., and Trachtenberg, A. (2008). Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes. *Information Processing in Sensor Networks*, 22(24):457–466.

[Lee et al. 2011] Lee, K., Lee, Y., Choi, H., Chung, Y. D., and Moon, B. (2011). Parallel data processing with mapreduce: A survey. *SIGMOD Record*.

[Mell and Grance 2010] Mell, P. and Grance, T. (2010). The NIST Definition of Cloud Computing. *Communications of the ACM*, 53(6):50.

[MEMSIC 2015] MEMSIC (2015). Datasheet MICAz: Wireless measurement system. http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf.

[MongoDB 2015] MongoDB (2015). MongoDB inc: For giant ideas. https://www.mongodb.com/.

[OpenStack 2015] OpenStack (2015). OpenStack: Open source software for creating private and public clouds. https://www.openstack.org/.

[Tan et al. 2011] Tan, Y. S., Tan, J., Chng, E. S., Lee, B., Li, J., Date, S., Chak, H. P., Xiao, X., and Narishige, A. (2011). Hadoop framework: impact of data organization on performance. *Software, practice and experience*.