

# Using NViz tool for Environmental Sensor Networks

Anh-Vu Dinh-Duc<sup>1</sup>, Non-member

## ABSTRACT

Visualization tools help users to observe the status of the Wireless Sensor Networks (WSNs). Although various visualization tools have been created for certain projects so far, these tools can only be used in certain scenarios, due to their hard-coded packet formats and network's properties. To speed up development process, a visualization tool which can adapt to any kind of WSN is essentially necessary. A general-purpose visualization tool - NViz, which can represent and visualize data for all WSN applications, is proposed. NViz allows user to set their network's properties and packet formats through XML files. Based on properties defined, user can choose the meaning of them and let NViz represents the data respectively. Furthermore, a better Replay mechanism, which lets researchers and developers debug their WSN easily, is also integrated in this tool. NViz is designed based on a layered architecture which allows for clear and well-defined interrelationships and interfaces between each components. As a demonstration, NViz is used for designing an environmental sensor network.

**Keywords:** Visualization tool, Environmental Sensor Networks

## 1. INTRODUCTION

Recently, there has been increasing interest in WSNs that are constructed by sensor nodes. Many WSN applications in the fields of environmental monitoring, security, industrial control, and object tracking have been introduced and deployed in practice. Despite the rapid development of sensor networks, there are still many challenges developers face in deployment of WSN applications:

1) Information visualization: Visualization tools are used to gather information about network properties such as: node status (e.g., battery level or communication power), network topology, wireless bandwidth, or link state and present them in graphs or other forms on screen. However, WSNs are always application-specific which require different network properties to be monitored. Besides, different packet formats are also needed to be designed for different

application scenarios. Therefore, most existing visualization tools today are developed under specialized applications which are almost useless when applications changed.

2) Unit conversion: Raw sensor data collected from sensor nodes are typically provided as direct digital readings from an ADC (Analog to Digital Converter). Voltage of the sensor is sampled and converted to a number relative to some reference voltages. To make the data useful, visualization tools should provide the ability of converting raw sensor data to engineering units based on convert function defined by user. Most existing visualization tools are lack of that ability.

3) Simulation: The rapidly changing state of sensor network due to fast incoming packets makes it hard for users to debug their WSNs. Researchers and developers always want to observe the detail variations inside their network to verify their algorithms or to validate their designs. However, most existing visualization tools only provide a basic replay function which helps users record their incoming packet in log files and playback them later. Users cannot determine when to start playing back or run forward/backward through the log files. A new "record and play back" mechanism to enable user to monitor their WSNs like videos is therefore needed.

In this paper, we present NViz, a general purpose visualization tool which is designed to take these challenges into account. Users now can set up their network properties and packet formats through two XML files. Moreover, convert function for each network properties can also be defined in XML file. An environmental sensor network application is designed to demonstrate the robustness of the tool.

## 2. RELATED WORKS

In recent years, there are many visualization tools have been introduced such as: SpyGlass[3], MoteView[4], NetTopo[5], Octopus[6] . . . However, most of them are developed for specialized applications and cannot verify with the change of application scenarios[1, 2]. Users have to modify and re-compile program or create new plug-in module for these tools if they want to use it in their applications.

NetViewer[7], a universal visualization tool for WSN, has been introduced at GLOBECOM 2010. It allows users define packet format in XML file and translates packets according to that. In that way, the development of WSNs and visualization tool could be

separated. The Netviewer can be used for visualization no matter what the node programs are or which the protocols the base station node uses to communicate with the host. It also provides replay function for debugging purpose and on-line data dissemination for data sharing.

However, NetViewer accepts only one packet format, limiting it to be used as a general purpose visualization tool. In practice, WSNs can use many packet formats for different purposes such as association notification, event update or data update. Moreover, replay function of NetViewer is also useless in a large-scale WSN because it only supports users to play, step by step, from beginning through the data log file. In another side, the lack of unit convert function makes NetViewer's data become almost meaningless to human beings.

### 3. DESIGN OF ZIGBEE NETWORK LAYER

#### 3.1 Overview of ZigBee

There are 3 device types in a ZigBee network:

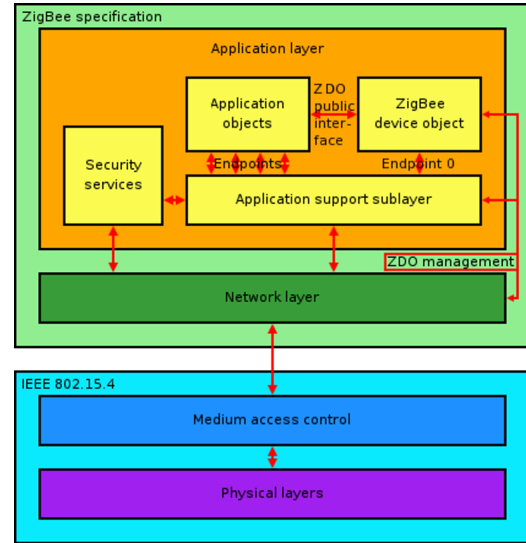
- 1) ZigBee Coordinator (ZC) initiates and configures the network formation. It can act as an IEEE 802.15.4 PAN Coordinator or as a ZigBee Router once the network is formed. There is only one ZigBee Coordinator in each ZigBee network.

- 2) ZigBee Router (ZR) acts as an IEEE 802.15.4 PAN Coordinator and participates in multi-hop routing of packet after associating with the ZC or with a associated ZR.

- 3) ZigBee End Device (ZED) does not participate in routing or allow other devices to associate with it.

The ZigBee network layer provides mechanisms for these devices to join or leave a network, and to route frames to their intended destinations. In addition, the discovery and maintenance of routes between devices devolve to the network layer. Also, the discovery of one-hop neighbors and the storing of pertinent neighbor information are done at the network layer.

ZigBee is best described by referring to the 7-layer OSI model for layered communication systems. The Alliance specifies the bottom three layers (physical, data link, and network) as well as an Application Support Sub-layer (APS) that allows end developers to design custom applications which use the services provided by the lower layers. Fig. 1 represents the layered architecture of the ZigBee/IEEE 802.15.4 protocol stack. It should be noted that the ZigBee Alliance use an existing data link and physical layers that were specified in the IEEE 802.15.4 standard. Low-rate, low-power consumption and low-cost wireless networking are the key features of the IEEE 802.15.4 protocol, which typically fit the requirements of WSNs. ZigBee specification relies on the IEEE 802.15.4 data link and physical layers, building up the network and application layer, thus defining a full protocol stack for Low-Rate Wireless Private Area



**Fig.1:** The ZigBee/IEEE 802.15.4 protocol stack.

Networks (LR-WPAN). In this paper, the architecture for Network layer in that protocol stack is focused.

#### 3.2 ZigBee Network Layer

To interface with the application layer, the network layer conceptually includes two service entities:

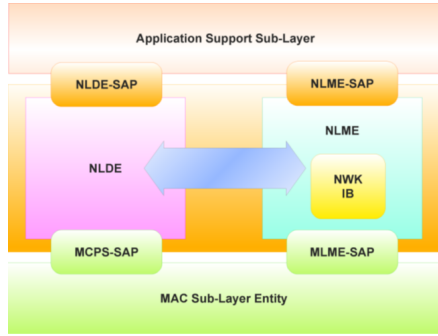
- 1) Network Layer Data Entity (NLDE): provides the data transmission service for upper layer via its associated Service Access Point - NLDE-SAP. It takes responsibility of generating Network level Protocol Data Unit (NPDU), topology-specific routing and security. More specifically, NLDE receives an Application Protocol Data Unit (APDU) from the application layer and create a new corresponding NPDU based on the implemented routing and security algorithm. After that, it sends the NPDU to MAC layer through MAC Common Part Sub-Layer SAP (MCPS-SAP).

- 2) Network Layer Management Entity (NLME): provides the management service for upper layer via its Service Access Point - NLME-SAP. Moreover, the NLME also utilizes the NLDE to achieve some of its management tasks. It also maintains a database of managed objects known as the Network Information Base - NIB (Fig. 2).

In next part, we will propose our architecture for implementing this Network Layer.

#### 3.3 Novel ZigBee Network Layer Architecture

Because ZigBee network layer has to provide many types of service, we decide to conceptually split it to several independent components. Each of them shall take responsibility of a specific network service and can be modified without any further influence on other component.



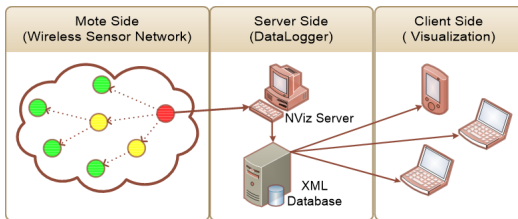
**Fig.2:** The ZigBee network layer.

There are a lot of internal as well as external interfaces in network layer defined by ZigBee specification. In this paper, only functionalities, design and interfaces of each component that are not specified in ZigBee specification 2008, are presented. Other interfaces can be found in ZigBee specification 2008.

Each device type in a ZigBee network (ZC, ZR and ZED) needs network layer to provide different services. Therefore, to limit the memory usage, we try to separate implementation code between coordinator, router and end device by using dummy component and preprocessor directives in configuration to change components' connections.

#### 4. VISUALIZATION TOOL NVIZ

NViz is designed as client-server model (Fig.4) where the server part handles packet access and all information storage. It also represents network behaviors in real time. The client part allows user to remote access to database and review historical network behaviors like a video thank to checkpoint mechanism in server-side. In the following part, we will describe NViz architectural model as well as explain the responsibility of each layer and the components that we have implemented in each layer.



**Fig.4:** Client-Server model of NViz.

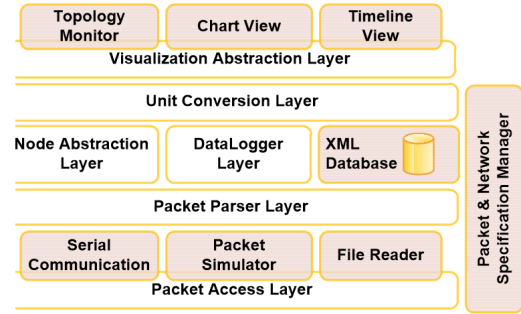
##### 4.1 Server Architecture of NViz

In server-side, NViz is designed and divided into five independent layers and a cross-layer component (Fig.5) which can be developed and tested separately.

*Packet and Network Specification Manager* is a component involved in four lower layers. By loading two XML specification files, NViz allows users to

configure this component and lets it provide information about Network properties and Packet formats to other layers (see Section 5).

Furthermore, each layer in the architecture takes responsibility for handling one part of visualization and can be modified without any further influence on other layers.



**Fig.5:** Server architecture layers.

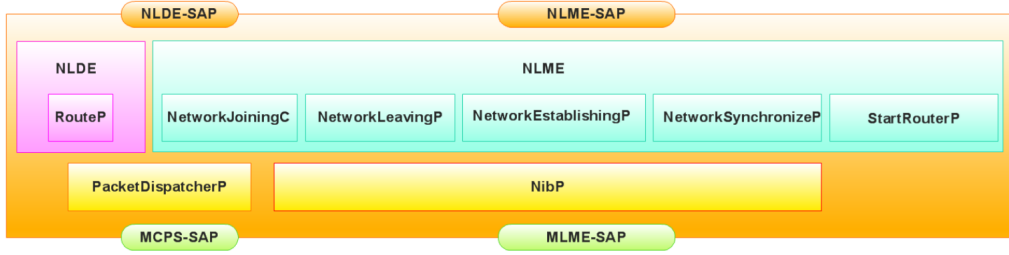
1) *Packet Access Layer*: This layer is responsible for generating packets. Currently, we have developed three approaches for packet access. Packets can be received from a sink node that connected to host, generated by user through Packet Simulator component or read from an input file. NViz currently supports TinyOS serial communication library. Moreover, with file access approach, NViz can also be used to support other simulator programs like ns-2 or OM-NeT++.

2) *Packet Parser Layer*: this layer is responsible for parsing valid packets received and discarding the bad packets. Packets will be parsed due to network properties and packet formats information contained in two XML files. These files are defined and loaded to Packet & Network Specification Manager component by user at start up time.

3) *Node Abstraction and Datalogger Layer*: By using network properties information provided by Packet & Network Specification Manager component, Node Abstraction component will abstracts virtual nodes, links and environment. Information received from Packet Parser layer will be cached in this layer and can be accessed easily by upper layer as node property, link property and environment property. Datalogger component in this layer is used to generate checkpoints and logs corresponding to current network state presented by Node Abstraction component (see Section 5).

4) *Unit Conversion Layer*: In this layer, all property values from Node Abstraction component will be converted by corresponding convert function (see Section 6).

5) *Visualization Abstraction Layer*: This layer provides interface for user to interact with the data collected. Many visualization services can be easily implemented in this layer by using data services provided in lower layer. Currently, this layer consists

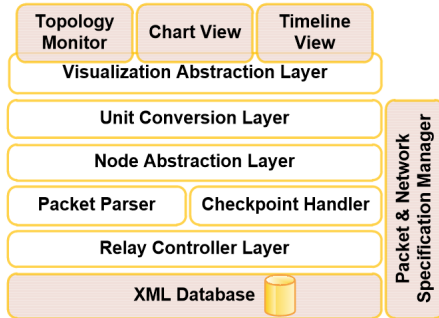


**Fig.3:** The novel ZigBee network layer architecture.

of three components: Topology Monitor, Chart View and Timeline View component.

#### 4.2 Client Architecture of NViz

Components within NViz client are conceptually split into one of five layers and a cross-layer component. Among them, three upper layers take the same responsibility as in server division. In addition, Packet & Network Specification Manager component will be initialized by loading two XML files stored in server's XML database.



**Fig.6:** Client architecture layers.

Other two layers' responsibility is described below:

1) *Relay Controller Layer*: This is the main layer in NViz-client. It uses a new mechanism to allow user review their historical networks activities easily and efficiently. Network can be simulated like a video. User can start to play at any time in the past as well as choose the speed of playing. Depending on user's requests, this layer starts a system timer to generate packets from checkpoints and logs stored in XML database. Section 6 represents more information about the replay mechanism.

2) *Checkpoint Handler Layer*: Packet Parser component takes the same responsibility as in NViz-server. Checkpoint Handler component is added to this layer to provided ability of rebuilding Node Abstraction component when checkpoints are loaded by Replay Controller Layer.

### 5. NETWORK AND PACKET SPECIFICATION

One of the most prominent characteristics of NViz is flexibility. To achieve this, NViz allows users to characterize their mote platform as well as packet formats through two XML files: Network Specification and Packet Specification.

#### 5.1 Network Specification

Nowadays, there are many Mote platforms available on the market. Each platform is designed and optimized for a particular application. Therefore, each node has some metadata that is constant for the entire deployment lifetime such as: set of sensors, configuration, convert functions Moreover, each application scenario also needs to consider specific attributes of link or environment. In NViz, all these information are specified by a network specification file and fixed during network operation. Fig.7 shows how a WSN can be specified in NViz.

```
<NetworkSpecification LogPerCheckpoint="100">
  <NodeProperties>
    <Property ID="0" convert="x" length="2"
      max="65535" min="0">Address</Property>
    <Property ID="1" convert="x" length="1"
      max="4" min="1">Function</Property>
    <Property ID="2" convert="x*122.3/[Vref]" length="2"
      max="1023" min="0">Temperature</Property>
    <Property ID="3" convert="1.223*1024/x" length="2"
      max="1023" min="0">Vref</Property>
  </NodeProperties>
  <LinkProperties>
    <Property ID="1" convert="x" length="1"
      max="255" min="1">Strength</Property>
  </LinkProperties>
  <EnvrProperties>
    <Property ID="1" convert="x" length="1"
      max="23" min="0">Channel</Property>
    <Property ID="2" convert="x" length="2"
      max="65535" min="0">PANID</Property>
  </EnvrProperties>
</NetworkSpecification>
```

**Fig.7:** An example of Network Specification XML file.

Network in NViz is specified by three types of property consisting of node, link and environment property. Each property possesses the following attributes:

- **ID**: Each property has a unique ID number different from all other properties of the same type. This ID will be used in Packet Specification file to determine what property is associated with particular packet's field.

- *Convert*: Convert function of property.
- *Length*: Number of bytes of property. This information is also used to determine length of packet's field associated with this property.
- *Max, Min*: Maximum and minimum values of property (before being converted). These attributes will be used to validate incoming packets.
- *Name*: Name of property.

## 5.2 Packet Specification

WSNs can be developed and deployed for many different application scenarios such as: environmental sensing, healthcare, object tracking. In each scenario, WSN provides different kind of information and needs a different set of packet formats. To be able to work with any kind of application scenario, NViz allows user to define their packet formats in a packet specification file. Fig.8 represents an example of packet specification file.

```
<PacketSpecification PacketIDLength="2">
  <Packet ID="1" description="Associate">
    <Field ID="0" type="Node">SourceAddress</Field>
    <Field ID="0" type="Node">DestinationAddress</Field>
    <Field ID="1" type="Link">LinkStrength</Field>
    <Field ID="1" type="Node">NodeFunction</Field>
  </Packet>
  <Packet ID="2" description="UpdateTemperature">
    <Field ID="0" type="Node">NodeAddress</Field>
    <Field ID="3" type="Node">VRef</Field>
    <Field ID="2" type="Node">Temperature</Field>
  </Packet>
  <Packet ID="3" description="Update Function">
    <Field ID="0" type="Node">NodeAddress</Field>
    <Field ID="1" type="Node">Function</Field>
  </Packet>
  <Packet ID="4" description="Update Environment">
    <Field ID="1" type="Envr">Channel</Field>
    <Field ID="2" type="Envr">PANID</Field>
  </Packet>
</PacketSpecification>
```

**Fig.8:** An example of Packet Specification XML file.

In NViz, every packet starts with packet ID field which determines its packet type. Length of packet ID number is set through attribute PacketIDLength of PacketSpecification element. Moreover, each packet type also has description attribute which defines its name. Each packet consists of a number of fields. Each field is specified by its associated property and its name. Associated property is determined by its ID and type. For example, in Fig.5, packet Associate has four fields:

- *SourceAddress* and *DestinationAddress*: are associated with Address property of node.
- *LinkStrength*: are associated with Strength property of link.
- *NodeFunction*: is associated with Function property of node.

NViz also provides wizards to help user define their Network Specification and Packet Specification files easily.

## 6. UNIT CONVERSION

Unit conversion layer takes responsibility of combining raw sensor data and calibration coefficients to

return final data in engineering units for Visualization Abstraction layer. Moreover, convert functions are defined in Network Specification file and can be changed easily for particular application.

Every property in NViz has a unit convert function which has a general form:

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{c0}, \dots, \mathbf{cn}, \mathbf{d0}, \dots, \mathbf{dn})$$

Where  $\mathbf{x}$  is the raw sensing data of current property,  $\mathbf{ci}$  is the set of calibration coefficients and  $\mathbf{di}$  is the set of dependant variables. Dependant variables are other properties which has the same type of current property.

An example of using convert functions in WSNs which uses Micaz platform is showed Fig.4. All sensing data in Micaz are depended on reference voltage which depends on battery level. Because ADC in Micaz is 10-bit, we have following equation:

$$V_{input} = \frac{ADC_{temperature} * V_{ref}}{1024}$$

Moreover,  $V_{ref}$  depends on battery voltage and has following convert function ( $V_{BG} = 1.223V$ ):

$$V_{ref} = V_{batt} = \frac{V_{BG} \times 1024}{ADC_{battery}} (V)$$

As an example, assume that we use a linear temperature sensor (like LM35DZ) whose output voltage is linearly proportional to the Celsius temperature (e.g. 10mV/°C) and  $V_{ref} = 122.3mV$ . We will have the final convert function for temperature property:

$$Temperature = \frac{ADC_{temperature} * 122.3}{ADC_{battery}} (^\circ C)$$

Therefore, in Fig.7,  $V_{ref}$  property has the convert function: "1.223\*1024/x" and Temperature property has the convert function: "x\*122.3/[Vref]".

```
public class $className implements Convert {
  public Double convert(Long x, Object ref)
  {
    return ($expression);
  }
}
```

**Fig.9:** A template file for unit conversion.

To implement this layer, NViz uses javax.tools library to compile convert functions in run-time while loading Network Specification file [9]. Each property in NViz has its own convert class responsible for convert its value. Every convert class in NViz takes the same template (Fig.9). In fact, a convert class is produced in following steps:

- When a property is loaded from Network Specification file, a string containing the contents of the template file will be created. This string is the convert class for this property and will be compiled at the final step.



- **\$className** in convert string will be replaced by name of current property and **\$expression** will be replaced by its convert function.

- All variables located between square brackets in conversion function will be replaced using pattern: **((Type) ref.Name)**. Type is type of property and Name is name of variable. For example, in conversion function of temperature property, **[Vref]** will be replaced by **((Node) ref.Vref)**

- Finally, convert string will be compiled to create convert class. This class will be used to convert data whenever Visualization Abstraction layer requests data from lower layer.

## 7. REPLAY MECHANISM

NViz implements a new replay mechanism to better support users to observe or debug their networks. When the packets arrive at host, NViz not only displays them on screen but also keeps a record on them. Record in NViz consists of checkpoints. Each checkpoint contains network state information, and a list of logs.

```
<Checkpoint t="1328163686181">
  <Node addr="0" att1="1.0" att2="102.0" att3="378.0">
    <Link att1="123.0" dest="1"/>
    <Link att1="213.0" dest="2"/>
  </Node>
  <Node addr="1" att1="2.0" att2="103.0" att3="356.0">
    <Link att1="158.0" dest="3"/>
    <Link att1="153.0" dest="4"/>
  </Node>
  <Node addr="2" att1="2.0">
    <Link att1="154.0" dest="6"/>
  </Node>
  <Node addr="3" att1="2.0">
    <Link att1="143.0" dest="5"/>
  </Node>
  <Node addr="4" att1="3.0"/>
  <Node addr="5" att1="3.0"/>
  <Node addr="6" att1="3.0"/>
  <Envr att1="11.0" att2="1.0"/>
  <L p="0|2|0|3|1|6F|0|7B|" t="1328163457311"/>
  <L p="0|2|0|4|1|65|0|70|" t="1328163469215"/>
  <L p="0|2|0|5|1|92|0|84|" t="1328163488303"/>
  <L p="0|2|0|6|1|86|0|79|" t="1328163509031"/>
  <L p="0|1|0|2|0|7|91|3|" t="1328163529150"/>
  <L p="0|2|0|7|1|9C|0|80|" t="1328163551462"/>
  <L p="0|2|0|0|1|77|0|68|" t="1328163580910"/>
  <L p="0|2|0|1|1|62|0|66|" t="1328163603094"/>
  <L p="0|2|0|3|1|6D|0|79|" t="1328163625542"/>
  <L p="0|2|0|4|1|63|0|71|" t="1328163646006"/>
</Checkpoint>
```

**Fig.10:** Check-point file.

Fig. 10 represents an example of checkpoint in practice. Every checkpoint in NViz has “t” attribute which determines when it was created. Network state information is represented in checkpoint by a list of Node, Link and Envr elements. All of this information allow Checkpoint Handler component in NViz-client to reconstruct the network completely. In addition, checkpoint also contains a number of logs which are stored in “L” elements. Number of logs per checkpoint is set by “LogPerCheckpoint” in Network Specification file (Fig. 7). Logs contain information about incoming packets, including receiving time and packets data.

Replay Controller Layer in NViz-client uses the system timer to simulate player time. It allows user to

play back at any time in the past as well as choose the speed of playing. NViz-client has Seek and Control Bar like any classical media player. Whenever user chooses time in Seek Bar, NViz will load the most recent checkpoint before that and reconstruct network accordingly. After that, logs in that checkpoint will be loaded until reach desired time. Depends on speed chosen by user, a system timer will be started. By calculating the time interval between packets, that timer can replay the historical network activities almost the same as actual.

Fig.11 represents a snapshot of NViz tool for above environmental sensor network application.

## 8. CONCLUSION

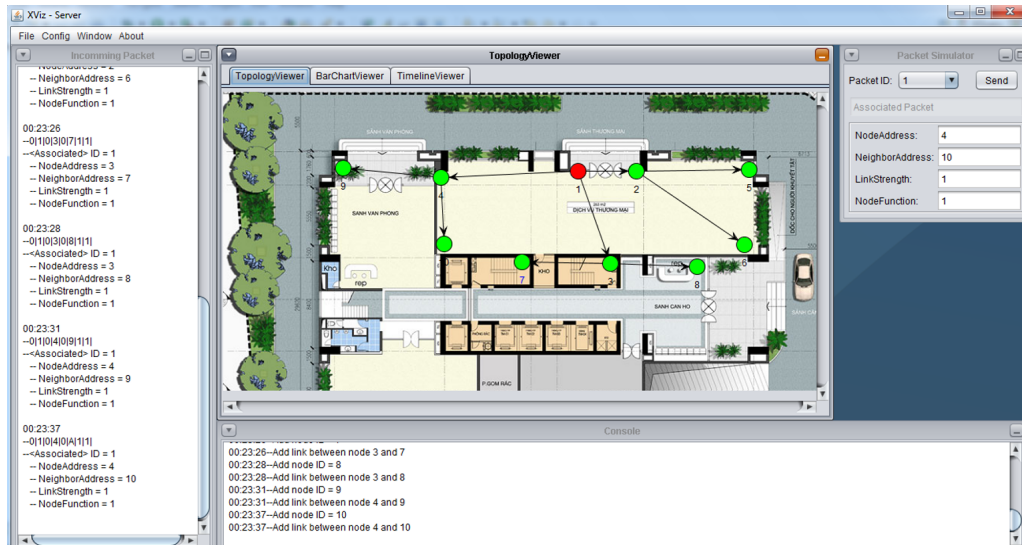
In this paper, a new visualization tool NViz, which can represent and visualize data for any kind of WSN, is proposed. This tool allows users to set their network’s properties and packet formats through XML files. Based on properties defined, user can choose the meaning of them and let NViz represents the data respectively. Replay mechanism, which lets researchers and developers debug their WSN easily, is also integrated in this tool. Furthermore, by defining convert functions in Network Specification file, NViz makes sensor data understandable to human beings.

An environmental sensor network application is designed to demonstrate the robustness of the visualization tool.

NViz is designed based on a layered architecture which allows for clear and well-defined interrelationships and interfaces between each components. More works in the near future will be carried out to prove the universality and efficiency of the proposed architecture.

## References

- [1] B. Parbat, a. K. Dwivedi, and O. P. Vyas, “Data Visualization Tools for WSNs: A Glimpse,” *International Journal of Computer Applications*, vol. 2, no. 1, pp. 14–20, May 2010.
- [2] W. L. Lee, A. Datta, and R. Cardell-Oliver, “Network management in wireless sensor networks,” *Handbook of Mobile Ad Hoc and Pervasive Communications*, pp. 1-20, 2006.
- [3] C. Buschmann, D. Pfisterer, S. Fischer, S. P. Fekete, and A. Kröller, “SpyGlass: a wireless sensor network visualizer,” *Acm Sigbed Review*, vol. 2, no. 1, pp. 1-6, 2005.
- [4] M. Turon, “MOTE-VIEW: A Sensor Network Monitoring and Management Tool,” *The Second IEEE Workshop on Embedded Networked Sensors*, 2005. EmNetS-II., pp. 11-18.
- [5] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang, and M. Hauswirth, “NetTopo: Beyond Simulator and Visualizer for Wireless Sensor Networks,” 2008 Second International Conference



**Fig.11:** An environmental sensor network application .

on Future Generation Communication and Networking, pp. 17-20, Dec. 2008.

- [6] R. Jurdak, A. G. Ruzzelli, A. Barbirato, and S. Boivineau, "Octopus: monitoring, visualization, and control of sensor networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 8, pp. 1073-1091, Aug. 2011.
- [7] L. Ma et al., "NetViewer: A Universal Visualization Tool for Wireless Sensor Networks," in *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, 2010, pp. 1-5.
- [8] B. L. Sa, "Wireless Sensor Network Repairing By Luis Sa," 2008.
- [9] D. J. Biesack, "Create dynamic applications with javax.tools" [Online]. Available: <http://www.ibm.com/developerworks/java/library/j-jcomp/>
- [10] "BaseX, the XML Database" [Online]. Available: <http://www.basex.org/>
- [11] "JUNG:Java Universal Network/Graph Framework" [Online]. Available: <http://www.jung.sourceforge.net/>
- [12] "JFreeChart: a free Java chart library" [Online]. Available: <http://www.jfree.org/jfreechart/>



**Anh-Vu Dinh-Duc** is a professor at the University of Information Technology (UIT), where he leads the Embedded Systems Group. His research interests include design automation of embedded systems, hardware/software verification, VLSI CAD, and reconfigurable architectures. Prof. Anh-Vu Dinh-Duc received the Master and Ph.D. degrees in Microelectronics from the Institut National Polytechnique de Grenoble

(INPG), France, in 1998 and in 2003, respectively. Prof. Anh-Vu Dinh-Duc currently serves as a program/organizing committee member of several ACM and IEEE conferences. He has been a member of the IEEE since 1997.