УДК 004.23, 004.72

СОЗДАНИЕ УНИВЕРСАЛЬНОГО ИНТЕРФЕЙСА ВЫСОКОГО УРОВНЯ ДЛЯ «УМНОГО» УЗЛА БЕСПРОВОДНОЙ СЕНСОРНОЙ СЕТИ

Аверкин Алексей Николаевич[1], Лавров Георгий Константинович[2]

[1]*Кандидат физико-математических наук, доцент*
ГБОУ ВО Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: averkin2003@inbox.ru.

[2]*Аспирант*
ГБОУ ВО Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: glavrov@mail.ru.

В статье представлена модель «умного» сенсорного узла и схема функционирования. Сформулирована способ передачи данных и правил на основе языка разметки. Описаны типы XML-парсеров и измененная модель «умного» сенсорного узла. Кроме того, предложен язык разметки для команд управления нечетким контроллером, сделано сравнение объема передаваемых данных при использовании языка XML и бинарного формата данных. Сделан вывод об области применения языка XML и бинарного формата передаваемых данных.

Ключевые слова: беспроводная сенсорная сеть, умный сенсорный узел, язык разметки, нечеткий контроллер.

CREATION OF A UNIVERSAL HIGH-LEVEL INTERFACE FOR A WSN SMART NODE

Averkin Alexey[1], Lavrov Georgy[2]

[1]*Candidate of Science in Physics and Mathematics, professor*
Dubna International University of Nature, Society and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: averkin2003@inbox.ru.

[2]*PhD student*
Dubna International University of Nature, Society and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: glavrov@mail.ru.

The article presents a model of the smart sensor node and the scheme of its functioning. The markup language – based way of data and rules transmission is formulated. Some types of XML-parsers and the modified model of the smart sensor node are described. Furthermore, the markup language for commands running a fuzzy controller is proposed and the amounts of transmitted data using the XML and the binary data format are compared. In conclusion, the field of application of XML language and the binary format of transmitted data is determined.

Keywords: wireless sensor network, smart sensor node, markup language, fuzzy controller.

Введение

Беспроводные сенсорные сети – одна их самых перспективных технологий 21 века. Первые сенсоры были созданы в Университете Беркли совместно с корпорацией Интел.

Узел беспроводной сенсорной сети представляет собой программно-аппаратную платформу, состоящую из нескольких специальных сенсоров (например, датчик температуры, освещенности), установленную на автономный беспроводной контроллер. Беспроводная сенсорная сеть состоит из большого числа таких узлов, которые установлены в окружающей среде и функционируют как единое целое. Для возможности самоорганизации беспроводных сетей было разработано специальное программное обеспечение, а также среда разработки приложений. Специальное программное обеспечение реализует возможность коммуникации, маршрутизации и поддержки приложений в БСС.

Основная задача БСС заключается в сборе информации об окружающем мире, некоторой ее обработке и передача пользователю. Например, сеть может измерять температуру и влажность в различных точках здания.

БСС обладает некоторыми характеристиками, отличными от характеристик традиционных сетей и беспроводных сетей узлами которых являются компьютеры. Одна из главных – это энергопотребление, так как узлы работают от батарей. Затраты на взаимодействие между узлами гораздо выше, чем на вычисления, а значит, пересылка больших объемов данных существенно уменьшает время жизни сети. Узлы сети имеют маленький радиус для связи и могут общаться только с ближайшими соседями. Сеть состоит из большого числа узлов, что ведет к ошибкам измерения и передачи данных. Сеть должна продолжать работать правильно при выходе из строя некоторых узлов и при добавлении новых. Затраты энергии на передачу и обработку данных возрастают при увеличении сети.

Одной из основных проблем при разработке систем, основанных на беспроводных сенсорных сетях, является форма представления данных и формат данных, передаваемых между клиентским приложением и сетью и внутри сети.

Эта проблема важна, поскольку доступ к самой сети и к ее отдельным узлам должны получать как можно больше устройств и приложений, например, карманные компьютеры, мобильные телефоны, персональные компьютеры. Кроме того, сеть должна иметь возможность общаться с Интернет. Для решения этой проблемы необходимо создать универсальный язык общения высокого уровня, который мог бы свободно пониматься всеми перечисленными системами. Должна быть возможность преобразовывать этот язык из одной формы в другую для интеграции разнородных систем, работающих на различных платформах.

Таким образом, описанная выше проблема говорит о необходимости создания для беспроводной сенсорной сети универсального интерфейса высокого уровня. Такой интерфейс позволит создавать сети, обладающие гибкой системой взаимодействия с внешними приложениями различных типов и способные встраиваться в уже существующие архитектуры. Появится возможность изменять задачу для такой сети без перепрограммирования ее узлов.

1. Нечеткая продукционная система на узле беспроводной сенсорной сети

Для реализации универсального интерфейса высокого уровня установим на узле беспроводной сенсорной сети системы нечеткого логического вывода. Узел сети, в которой введена нечеткость, назовем «умным сенсорным узлом» [1]. Главной частью такого узла является система нечеткого вывода, которая состоит из базы знаний (набора нечетких правил вывода, основанных на лингвистических переменных) модуля фаззификации и дефаззификации, которые переводят фактические значения переменных в термы и обратно. Умный узел может аппроксимировать любую функцию входных параметров. Нечеткие правила могут быть также использованы для агрегации, маршрутизации, кластеризации, слияния данных, что ведет к уменьшению затрат энергии сети. Нечеткие правила для сети могут быть составлены экспертом или с помощью нечетких нейронных сетей.

Данная система должна быть универсальной, адаптивной, иметь маленький размер кода, должен быть специальный язык для правил.

1.1 «Умный» сенсорный узел

Общая схема «умного» сенсорного узла приведена на рисунке 1.

СРЕДА – объект мониторинга (реальная среда, или ее симулятор).

КЛИЕНТ – пользовательский интерфейс.

БСС – беспроводная сенсорная сеть.

Внутренние ФС – внутренние физические сенсоры. Измерительные устройства, установленные на «умном» сенсорном узле.

Внутренние ФА БД – база данных внутренних физических атрибутов. Внутренние физические атрибуты – числовые значения, полученные при измерении сенсорами физических параметров среды (температура, влажность, освещенность).

Внешние ФА – внешние физические атрибуты, числовые значения, посылаемые «умному» узлу клиентом или БСС.

Внешние ВА БД – база данных внешних виртуальных атрибутов. Внешние виртуальные атрибуты, числовые значения, посылаемые БСС или клиентом.

Внутренние ВА БД – база данных внутренних виртуальных атрибутов. Внутренние виртуальные атрибуты – числовые значения, которые вычисляются на основе физических атрибутов.

БЗ – база знаний (база правил).

Тонкие стрелки обозначают трафик данных. Толстые – трафик знаний (правил).

База данных атрибутов хранит имена атрибутов и их значения.

«Умный» сенсор – многофункциональное устройство, обрабатывающее запросы пользователей, способное настраивать и управлять БСС. Умный сенсор может измерять физические параметры среды и на их основе вычислять более сложные параметры. Например, по температуре и влажности вычислять комфортность. Этот параметр является виртуальным атрибутом (его нельзя измерить датчиками).

Основные функции умного сенсорного узла:

- узел измеряет с помощью датчиков физические параметры окружающей среды (влажность, температура) и сохраняет их в базе данных внутренних физических атрибутов;

- узел принимает физические параметры окружающей среды из БСС или от клиентского приложения и сохраняет их в базе данных внешних физических атрибутов;

- узел отправляет содержимое базы данных внутренних физических параметров среды в БСС или пользователю по запросу;

- узел принимает внешние виртуальные атрибуты из БСС или от клиента и сохраняет их в базе данных внешних виртуальных атрибутов;

- с помощью системы нечеткого вывода, базы правил, базы данных внешних и внутренних физических атрибутов, базы данных внешних виртуальных атрибутов узел вычисляет внутренние виртуальные атрибуты и сохраняет их в базе данных внутренних виртуальных атрибутов;

- узел отправляет внутренние виртуальные атрибуты в БСС и клиенту по запросу;

- узел дополняет базу данных внутренних виртуальных атрибутов и базу правил на основе информации, полученной из БСС и от клиента по запросу клиента;

- узел удаляет информацию о внутренних виртуальных атрибутах и правилах из базы данных по запросу клиента.

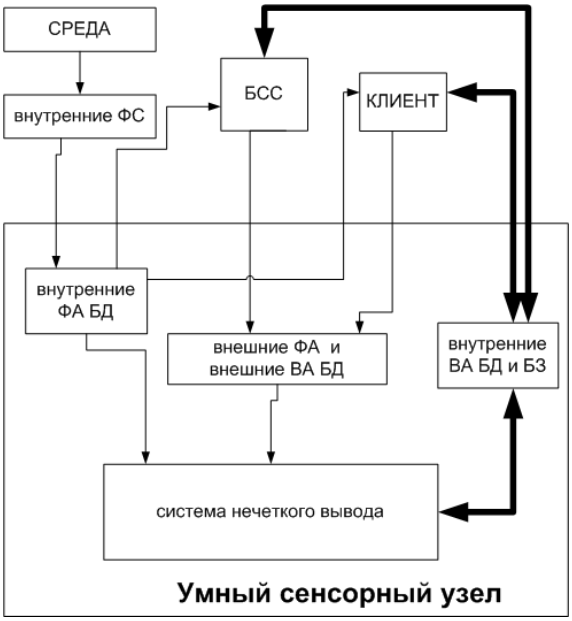

Рис. 1. Общая схема умного сенсорного узла

Первоначально базы данных внутренних и внешних виртуальных и физических атрибутов пуста. Пуста база правил. Работа узла начинается с посылки клиентов нечетких правил. Если условия или результаты правил содержат внешние атрибуты, то должны быть указаны узлы-источники, из которых их можно получить. Далее происходит следующее:

- нечеткие правила добавляются в систему нечеткого вывода;
- датчики измеряют физические атрибуты, и заполняется база данных внутренних физических атрибутов;
- узел получает данные из узлов-источников и заполняется базы данных внешних физических и виртуальных атрибутов;
- если есть данные в базах данных внешних и внутренних физических и виртуальных атрибутов, запускается нечеткий вывод, и вычисляются внутренние виртуальные атрибуты;
- содержимое баз данных внутренних физических и виртуальных атрибутов посылается в БСС или клиенту по запросу клиента;
- база правил посылается клиенту по запросу клиента;
- по запросу клиента удаляется часть содержимого или все содержимое баз данных умного узла.

По аналогии с экспертными системами можно определить метаправила, эти правила на «умном» узле будут определять, какая группа правил для системы нечеткого вывода будет использована в данный момент.

1.2 Возможное применение узлов с нечетким контроллером

При использовании баз правил в беспроводных сенсорных сетях эта сеть приобретает свойства обычной активной сети. Схожесть активной и беспроводной сети в этом случае заключается в том, что, отправляя «умному» узлу запрос и правила для его выполнения, этот узел становится своеобразным сервером. Однако различия остаются, в БСС присутствует трафик данных и трафик знаний, а в активной сети – только трафик данных.

Используя сенсор с нечетким контроллером, можно реализовать:

- эффективные алгоритмы маршрутизации;
- сокращение энергопотребления сети;
- управление трафиком данных.

Рассмотрим подробнее последний пункт.

Если весь поток данных измеряемых сетью будет передаваться клиенту для дальнейшей обработки, то из-за огромного объема данных и помех при измерении и передаче будет возникать большое число коллизий, а также существенно увеличится потребление энергии узлами, что быстро приведет к отказу сети. Что бы сократить передачу ненужных данных по сети, промежуточные узлы

или группы соседних узлов должны проводить предварительную обработку данных (например, вычисление среднего значения) и передавать другим узлам посчитанное среднее значение, а не весь поток данных.

Рассмотрим пример. В здании установлена БСС с нечеткими узлами, которая измеряет температуру, влажность и освещенность. Необходимо рассчитать какая из комнат является наиболее комфортной. Вместо передачи клиенту всего потока данных можно передать в сеть набор правил для нечетких контроллеров на вычисление параметра *COMFORTABILITY* (комфортность) по заданным параметрам *TEMPERATURE* (температура), *HUMIDITY* (влажность) и *ILLUMINATION* (освещенность):

$R^{(1)}$: ***IF*** *(TEMPERATURE* ***IS*** *HIGH)* ***AND*** *(HUMIDITY* ***IS*** *MIDDLE)* ***AND*** *(ILLUMINATION* ***IS*** *HIGH)* ***THEN*** *(COMFORT* ***IS*** *VERY HIGH).*

$R^{(2)}$: ***IF*** *(TEMPERATURE* ***IS*** *LOW)* ***AND*** *(HUMIDITY* ***IS*** *HIGH)* ***AND*** *(ILLUMINATION* ***IS*** *LOW),* ***THEN*** *(COMFORT* ***IS*** *LOW).*

Ниже на рисунке 2 показан процесс передачи данных и знаний при обработке запроса «какая комната является наиболее комфортной».

Сенсорам, расположенным в первой комнате, посылается набор правил и операция агрегации, после вычисления комфортности в первой комнате это значение и правила посылаются узлам сети во второй комнате, после вычисления комфортности во второй комнате, полученное значение сравнивается с комфортностью в первой комнате, и на узлы в третьей комнате посылается максимальное из сравниваемых значений и правила. После получения комфортности в третьей комнате полученное значение сравнивается со значением, которое пришло на узлы вместе с правилами, и максимальное из них передается клиенту. Таким образом, осуществляется агрегация данных при помощи нечеткой беспроводной сенсорной сети.

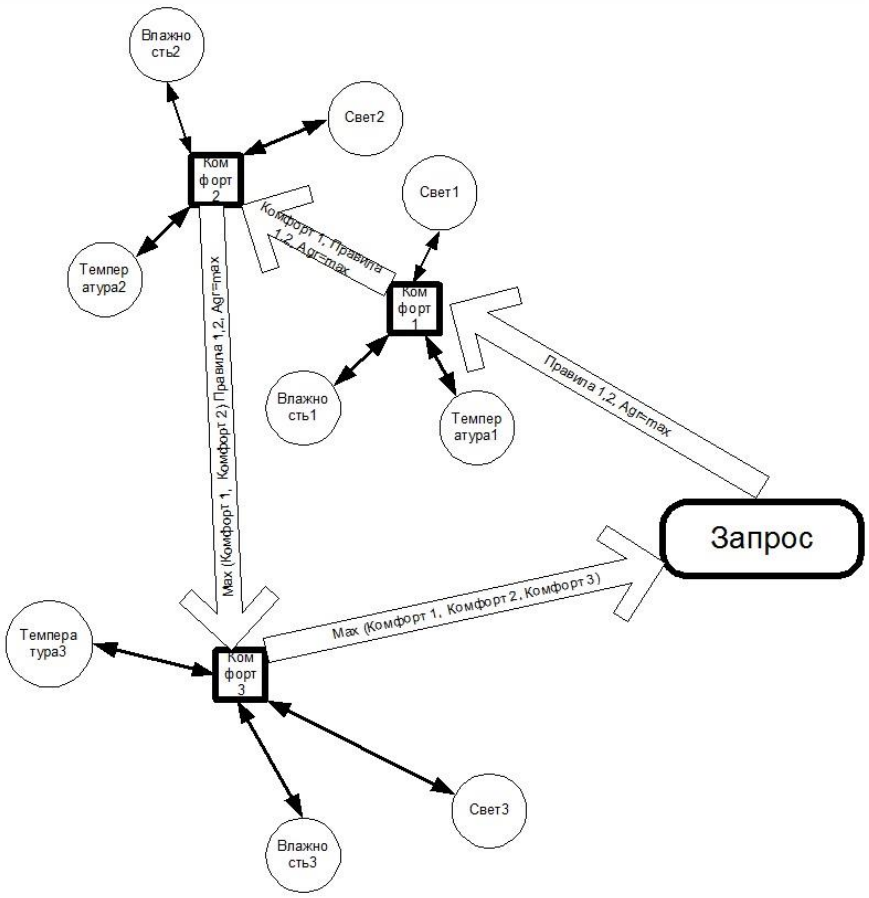

Рис. 2. Процесс передачи данных и знаний при обработке запроса «какая комната является наиболее комфортной»

2. Использование языка разметки и бинарных команд для передачи информации в беспроводных сенсорных сетях

В качестве универсального языка общения высокого уровня для передачи лингвистических переменных, термов и правил удобно использовать язык *XML* (*extensible markup language*). Это позволит передавать данные и правила в формате независимом от платформы, соединять в одну сеть гетерогенные узлы, использовать для взаимодействия с сетью различные устройства, например, карманные персональные компьютеры и мобильные телефоны. Кроме того, такой язык достаточно просто использовать. Обрабатывать правила и данные передаваемые с помощью языка разметки можно в зависимости от содержания самих данных (так как *XML* содержит не только данные, но и их описание). Язык можно использовать не только как средство передачи данных, но как стандарт описания правил и переменных [2-5].

Для реализации такой возможности на узлы сети должен быть установлен парсер. Главным требованием для парсера является маленький объем (ресурсы узла ограничены). Сам язык разметки также должен быть достаточно простым, набор тэгов и их названия должны быть минимальным, так как при передаче данных в форме *XML* возрастает трафик.

Рассмотрим этот подход более подробно.

2.1 Модель передачи лингвистических переменных и базы правил с помощью языка разметки

В беспроводной сенсорной сети присутствует «умный» сенсорный узел. Он получает информацию от других узлов сети, от клиента, от сенсоров, установленных на узле и с помощью встроенного нечеткого контроллера и базы знаний может эффективно вычислять различные величины. Получение информации и базы знаний здесь является важным этапом, так как именно способ передачи информации определяет возможность связи узла с различными устройствами и с узлами других типов.

Таким образом, одной из главных задач при создании универсального интерфейса является разработка программного комплекса по передаче знаний и лингвистических переменных между клиентским приложением и сенсорной сетью и между «умными» узлами сенсорной сети.

2.1.1 Модель передачи знаний

Общая модель передачи знаний изображена на рисунке 3.

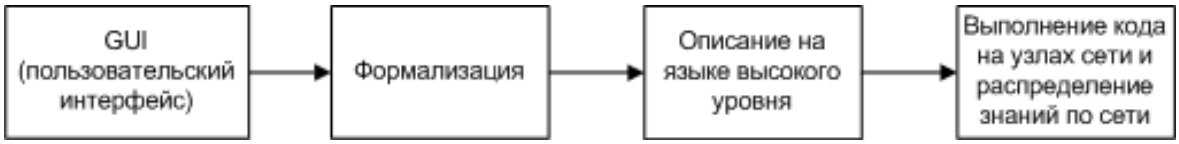

Рис. 3. Общая модель передачи знаний

Она состоит из следующих этапов:

- с помощью пользовательского интерфейса определяется предметная задача, ее основные объекты, условия и цели;

- задача переводится на язык нечеткой сенсорной сети, определяются лингвистические переменные, их термы и функции принадлежности;

- проблема в терминах сенсорной сети транслируется в описание на языке высокого уровня. В определенных условиях он может быть написан вручную разработчиком для экономии вычислительных ресурсов;

- выполнение участков кода на узлах реализует передачу знаний, лингвистических переменных, термов, функций принадлежности согласно заданному алгоритму.

В результате лингвистические переменные и знания распределяются по сети состоящей из «умных» сенсорных узлов.

Для реализации модели передачи знаний в качестве языка высокого уровня удобно использовать язык разметки на базе *XML*. Он может быть полезен не только для передачи знаний и лингвистических переменных, но и для комплексного управления сетью и передачи обычных данных. Основными преимуществами такого подхода является платформонезависимость, простота, расширяемость. Кроме того, поскольку языки разметки чрезвычайно широко используются в данный момент, то с узлами сети и со всей сетью в целом смогут общаться различные устройства, например, мобильные телефоны и карманные персональные компьютеры. Доступ к БСС, при таком подходе, можно получать и через Интернет. Язык разметки может быть использован не только как средство коммуникации, но и как способ описания объектов.

На рисунке 4 представлена схема взаимодействия узлов между собой на основе языка разметки.

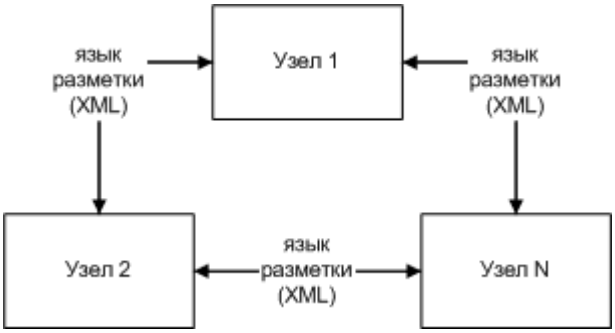

Рис. 4. Схема взаимодействия узлов между собой на основе языка разметки

На рисунке 5 изображена схема взаимодействия беспроводных сенсорных сетей на основе языка разметки.

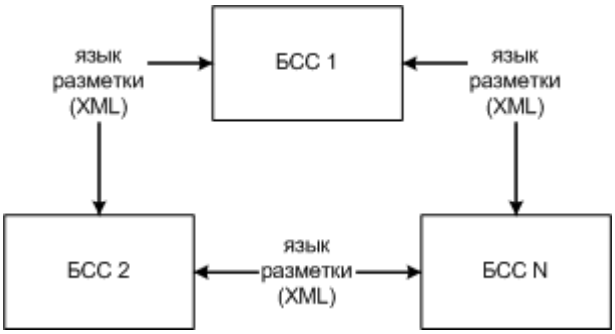

Рис. 5. Схема взаимодействия беспроводных сенсорных сетей на основе языка разметки

2.1.2 Парсеры

Для взаимодействия с помощью языка разметки помимо самого языка нужна *XML*-схема (*XMLSchema*), а также *DOM* (*document object model*)- или *SAX* (*simple API for XML*)-парсер. Они нужны для преобразования передаваемых текстовых данных во внутренние форматы приложений (узлов). Обработка происходит следующим образом (рис. 16):

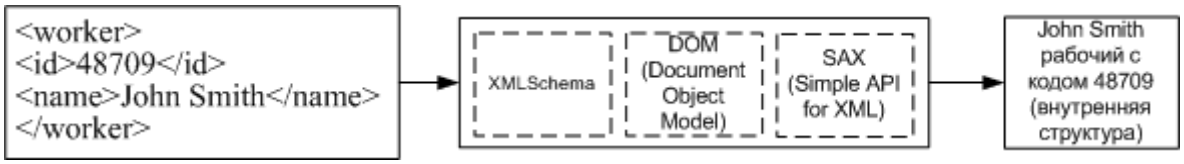

Рис. 6. Схема преобразования текстовых данных во внутренние форматы

DOM-парсер – это набор интерфейсов для создания внутреннего объектного представления документа. Объектная модель строится исходя из *XMLSchema* на основе древовидной структуры любого *XML*-документа. При этом *DOM* содержит только определение интерфейсов, никоим образом не регулируя внутреннюю реализацию самой модели документа. Представление документа требует большого объема памяти, сам парсер также является достаточно сложным.

SAX-парсер обрабатывает по очереди открывающие тэги, данные внутри тэгов, атрибуты и закрывающие тэги, причем у него имеются данные только о названиях тэгах и данных в текстовом формате. А для преобразования документа во внутреннюю структуру необходимо задавать соответствующие обработчики, которые будут вызываться из парсера. Таким образом, обработка идет в виде реакции на события. Это значит, что для простого *SAX*-парсера не нужна *XMLSchema*, он не требователен к ресурсам, является достаточно простым.

Для использования в беспроводных сенсорных сетях подходит *SAX*-парсер, так как ресурсы узла сильно ограничены.

Ниже (рис. 7) представлена общая схема взаимодействия с помощью языка разметки между узлами беспроводной сенсорной сети, между интерфейсом пользователя и всей БСС, а также между сетью и Интернетом. На рисунке также показана возможность подключения дополнительных устройств. В основе взаимодействия лежит простой *SAX*-парсер. Следует также отметить, что сети могут быть гетерогенными.

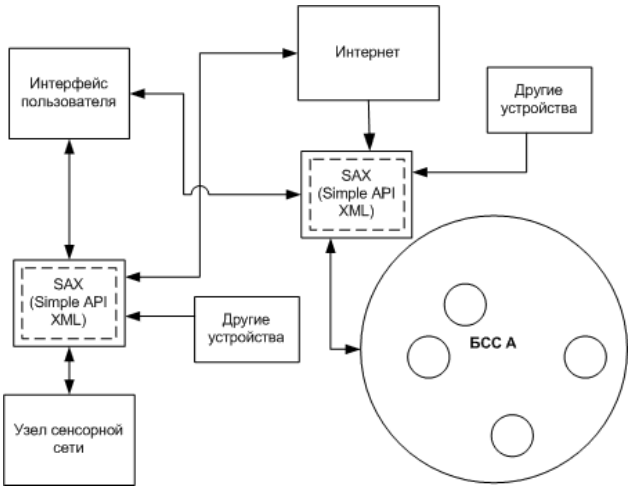

Рис. 7. Общая схема взаимодействия с помощью языка разметки

Из всего вышеперечисленного можно сделать вывод о том, что модель взаимодействия с помощью языка разметки является достаточно простой и гибкой. Она легко может быть встроена в модель «умного» сенсорного узла, из которых строится интеллектуальная беспроводная сенсорная сеть. Измененный «умный» сенсорный узел показан на рисунке 8.

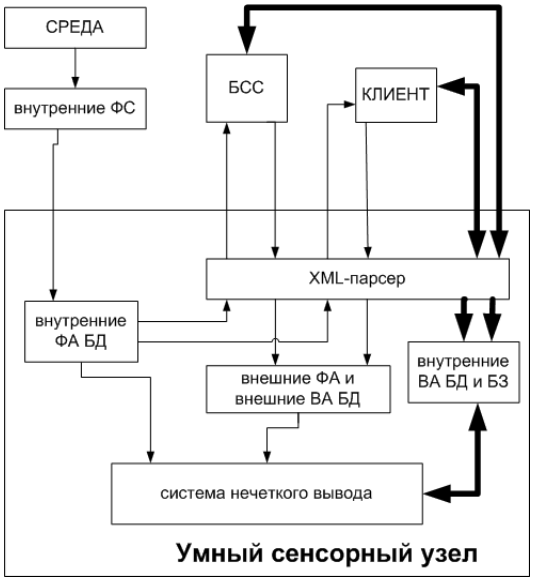

Рис. 8. Схема измененного «умного» сенсорного узла

2.2 Язык разметки на базе XML для управления нечетким контроллером

Для работы с «умным» сенсорным узлом необходимо управлять параметрами нечеткого контроллера, добавлять, изменять, удалять лингвистические переменные, правила и термы, получать и устанавливать значения. Для этого разработана система команд и записана в форме языка разметки.

2.2.1 Язык разметки

Ниже представлена схема предложенного языка разметки.

```
<?xml version="1.0" encoding="windows-1251"?>
<!ELEMENT al ( l, t,  o,  a,   e, li )>
<!ELEMENT at (lt, l,  n, x+,  y+     )>
<!ELEMENT ar ( r, l, lt,  n, li+, ti+)>
<!ELEMENT rl ( l)>
<!ELEMENT rt (lt)>
<!ELEMENT rr ( r)>
<!ELEMENT sl ( l, t,  o,  a,   e, li )>
<!ELEMENT st (lt, l,  n, x+,  y+     )>
<!ELEMENT sr ( r, l, lt,  n, li+, ti+)>
<!ELEMENT gl ( l)>
<!ELEMENT gt (lt)>
<!ELEMENT gr ( r)>
<!ELEMENT ls EMPTY>
<!ELEMENT ts EMPTY>
<!ELEMENT rs EMPTY>
<!ELEMENT ss ( l, s)>
<!ELEMENT cs EMPTY>
<!ELEMENT vg ( l)>
<!ELEMENT vs ( l, x)>
<!ELEMENT l  (#PCDATA)>
<!ELEMENT t  (#PCDATA)>
<!ELEMENT o  (#PCDATA)>
<!ELEMENT a  (#PCDATA)>
<!ELEMENT li (#PCDATA)>
<!ELEMENT e  (#PCDATA)>
<!ELEMENT lt (#PCDATA)>
```

```
<!ELEMENT n  (#PCDATA)>
<!ELEMENT x  (#PCDATA)>
<!ELEMENT y  (#PCDATA)>
<!ELEMENT ti (#PCDATA)>
<!ELEMENT r  (#PCDATA)>
<!ELEMENT s  (#PCDATA)>
<!ELEMENT er (#PCDATA)>
```

2.2.2 Обозначения

* – элемент может встречаться 0 или более раз.

? – элемент может встречаться один раз или не встречаться вообще.

\+ – элемент может встречаться 1 или более раз.

EMPTY – элемент является пустым.

#PCDATA (*parsed character data*) – анализируемые символьные данные, т. е. данные, которые могут обрабатываться парсером.

Если после элемента не стоит никакого знака, значит, элемент должен встречаться ровно один раз.

2.2.3 Описание элементов

Элемент *al* задает команду по добавлению лингвистической переменной в нечеткий контроллер. Команда по добавлению переменной содержит идентификатор лингвистической переменной (элемент *l*); тип лингвистической переменной (элемент *t*) – входная или выходная; операцию, которую выполняет нечеткий контроллер при выводе значения выходной лингвистической переменной (элемент *o*); сетевой адрес связанного значения лингвистической переменной (элемент *a*) – адрес узла на котором установлен датчик, endpoint связанного значения для лингвистической переменной (элемент *e*) – идентификатор компонента (например, датчика освещенности); идентификатор связанного значения для лингвистической переменной (элемент *li*).

At – элемент задает команду по добавлению терма в лингвистическую переменную нечеткого контроллера. Команда по добавлению терма содержит идентификатор терма (элемент *lt*); идентификатор лингвистической переменной, которой принадлежит терм (элемент *l*); количество точек в терме (элемент n); набор *x*-координат точек терма (элемент *x*); набор *y*-координат (мощностей) точек терма (элемент *x*).

Ar – элемент задает команду по добавлению правила в нечеткий контроллер. Команда по добавлению правила содержит идентификатор добавляемого правила (элемент *r*); идентификатор выходной переменной (элемент *l*); идентификатор выходного терма (элемент *lt*); количество входных лингвистических переменных (элемент n); список идентификаторов входных лингвистических переменных (элемент *li*); список идентификаторов входных термов (элемент *ti*).

Rl – элемент задает команду по удалению лингвистической переменной из нечеткого контроллера. Команда по удалению переменной содержит идентификатор удаляемой лингвистической переменной (элемент *l*).

Rt – элемент задает команду по удалению терма лингвистической переменной из нечеткого контроллера. Команда по удалению терма содержит идентификатор удаляемого терма (элемент *lt*).

Rr – элемент задает команду по удалению правила из нечеткого контроллера. Команда по удалению правила содержит идентификатор удаляемого правила (элемент *r*).

Sl – элемент задает команду по установке новых параметров для лингвистической переменной. Команда по установке параметров переменной содержит идентификатор лингвистической переменной (элемент *l*); тип лингвистической переменной (элемент *t*) – входная или выходная; операцию, которую выполняет нечеткий контроллер при выводе значения выходной лингвистической переменной (элемент *o*); сетевой адрес связанного значения лингвистической переменной (элемент *a*) – адрес узла

на котором установлен датчик, endpoint связанного значения для лингвистической переменной (элемент *e*) – идентификатор компонента (например, датчика освещенности), идентификатор связанного значения для лингвистической переменной (элемент *li*).

St – элемент задает команду по установке новых параметров для терма лингвистической переменной. Команда по установке параметров терма содержит идентификатор терма (элемент *lt*); идентификатор лингвистической переменной, которой принадлежит терм (элемент *l*); количество точек в терме (элемент n); набор *x*-координат точек терма (элемент *x*); набор *y*-координат (мощностей) точек терма (элемент *x*).

Sr – элемент задает команду по установке новых параметров для правила. Команда по по установке параметров правила содержит идентификатор правила (элемент *r*); идентификатор выходной переменной (элемент *l*); идентификатор выходного терма (элемент *lt*); количество входных лингвистических переменных (элемент *n*); список идентификаторов входных лингвистических переменных (элемент *li*); список идентификаторов входных термов (элемент *ti*).

Ql – элемент содержит команду по получению параметров лингвистической переменной. Команда содержит идентификатор лингвистической переменной, параметры которой необходимо получить (элемент *l*).

Qt – элемент содержит команду по получению параметров терма лингвистической переменной. Команда содержит идентификатор терма, параметры которого необходимо получить (элемент *lt*).

Qr – элемент содержит команду по получению параметров правила. Команда содержит идентификатор правила, параметры которого необходимо получить (элемент *r*).

Ls – элемент содержит команду по получению списка идентификаторов лингвистических переменных, которые содержатся в нечетком контроллере.

Ts – элемент содержит команду по получению списка идентификаторов термов, которые содержатся в нечетком контроллере.

Rs – элемент содержит команду по получению списка идентификаторов правил, которые содержатся в нечетком контроллере.

Ss – элемент содержит команду по установке статуса лингвистической переменной – включена или выключена. Команда содержит идентификатор переменной (элемент l) и статус, который будет установлен (элемент *s*).

Cs – элемент содержит команду по удалению всех данных из нечеткого контроллера.

Vg – элемент содержит команду по получению значения лингвистической переменной. Команда содержит идентификатор переменной, значение которой нужно получить (элемент l).

Vs – элемент содержит команду по установке значения лингвистической переменной. Команда содержит идентификатор переменной, значение которой нужно установить (элемент l); значение, которое будет установлено (элемент *x*).

L – элемент определяет идентификатор лингвистической переменной.

T – элемент определяет тип лингвистической переменной – входная выходная.

O – элемент определяет операцию, которую выполняет нечеткий контроллер при выводе значения выходной лингвистической переменной.

A – элемент задает сетевой адрес связанного значения лингвистической переменной – адрес узла на котором установлен датчик.

Li – тэг задает элемент списка, содержащего идентификаторы лингвистических переменных.

E – элемент задает endpoint связанного значения для лингвистической переменной – идентификатор компонента (например, датчика освещенности).

Lt – элемент задает идентификатор терма лингвистической переменной.

N – элемент задает размер списка элементов (например, списка идентификаторов переменных).

X – элемент задает *x*-координату точки терма лингвистической переменной или значение лингвистической переменной.

Y – элемент задает *y*-координату точки терма лингвистической переменной.

Ti – тэг задает элемент списка, содержащего идентификаторы термов лингвистических переменных.

R – элемент задает идентификатор правила.

S – элемент задает статус лингвистической переменной – включена или выключена.

Er – элемент задает код ошибки выполнения команды.

Предложенный XML-язык является достаточно простым и кратким. Он позволяет задать и получить все параметры нечеткого контроллера. Рассмотрим примеры команд по управлению нечетким контроллером.

2.2.4 Примеры XML-запросов для нечеткого контроллера

```
 <al>            - добавление лингвистической переменной.
   <l>4</l>
   <t>1</t>
   <o>1</o>
   <a>0</a>
   <e>2</e>
   <li>0</li>
</al>

<al>            – ответ.
   <er>0</er>
</al>

<at>            – добавление терма.
    <lt>4</lt>
    <l>4</l>
   <n>3</n>
   <x>0</x>
   <x>500</x>
   <x>800</x>
   <y>0</y>
   <y>0.5</y>
   <y>1.0</y>
</at>

<at>            – ответ.
  <er>0</er>
 </at>

 <ar>            – добавление правила.
   <r>3</r>
   <l>4</l>
   <lt>4</lt>
   <n>1</n>
   <li>3</li>
   <ti>3</ti>
 </ar>

<ar>            – ответ.
  <er>0</er>
 </ar>

 <gt>            – получение терма.
```

```
  <lt>4</lt>
</gt>
```

```
<gt>            – ответ.
  <er>0</er>
  <lt>4</lt>
  <l>4</l>
   <n>3</n>
   <x>0</x>
   <x>500</x>
   <x>800</x>
   <y>0</y>
   <y>0.5</y>
  <y>1.0</y>
</gt>
```

```
<ls></ls>       – получение списка идентификаторов лингвистических переменных.
```

```
<ls>            – ответ.
  <er>0</er>
  <n>1</n>
  <l>4</l>
</ls>
```

```
<rl>          – удаление лингвистической переменной.
  <l>4</l>
</rl>
```

```
<rl>          – ответ.
  <er>0</er>
</rl>
```

```
<cs></cs>     – очистка нечеткого контроллера.
```

```
<cs>          – ответ.
  <er>0</er>
</cs>
```

```
<vg>          – получение значения лингвистической переменной.
  <l>4</l>
</vg>
```

```
<vg>          – ответ.
  <er>0</er>
  <x>400.328</x>
</vg>
```

```
<vs>          – установка значения лингвистической переменной.
  <l>2</l>
  <x>234.567</x>
</vs>
```

```
<vs>          – ответ.
  <er>0</er>
</vs>
```

2.3 Бинарные команды для нечеткого контроллера

Представленную систему команд для нечеткого контроллера можно записать не только в формате *XML*, но и в обычном бинарном (массив байт) формате. Такое представление команд может потребоваться для взаимодействия внутри «умного» сенсорного узла. Рассмотрим бинарный формат команд. Первый байт во всех командах определяет тип команды, второй байт во всех командах определяет идентификатор команды.

2.3.1 Добавление лингвистической переменной

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*2Б*	*1Б*	*1Б*
0x04	0x00	*Lingvo ID*	*Lingvo Type*	*Operation*	*Link addrress*	*Link endpoint*	*Link ID*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x00	*Error ID*

Lingvo ID – идентификатор лингвистической переменной.

Lingvo Type – тип лингвистической переменной (входная, выходная).

Operation – операция, выполняемая нечетким контроллером при вычислении значения выходной переменной.

Link address – сетевой адрес связанного значения лингвистической переменной.

Link endpoint – идентификатор приложения для связанного значения лингвистической переменной.

Link ID – идентификатор связанного значения переменной.

2.3.2 Добавление терма

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*4Б * Count*	*4Б * Count*
0x04	0x01	*Term ID*	*Lingvo ID*	*Count*	*X coordinate*	*Y coordinate*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x01	*Error ID*

Term ID – идентификатор терма.

Lingvo ID – идентификатор лингвистической переменной, к которой будет принадлежать терм.

Count – количество точек в терме.

X coordinate – *x*-координата терма.

Y coordinate – y-координата терма.

2.3.3 Добавление правила

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*1Б*	*1Б * Count*	*1Б * Count*
0x04	0x02	*Rule ID*	*Out Lingvo ID*	*Out Term ID*	*Count*	*In Lingvo IDs*	*In Term IDs*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x02	*Error ID*

Rule ID – идентификатор правила.

Out Lingvo ID – идентификатор выходной лингвистической переменной (переменной в правой части правила).

Out Term ID – идентификатор выходного терма (терма в правой части правила).

Count – количество входных лингвистических переменных (переменных в левой части правила).

In Lingvo ID – идентификатор входной лингвистической переменной (переменной в левой части правила).

In Term ID – идентификатор входного терма (терма в левой части правила).

2.3.4 Установка лингвистической переменной

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*2Б*	*1Б*	*1Б*
0x04	0x0B	*Lingvo ID*	*Lingvo Type*	*Operation*	*Link addrress*	*Link endpoint*	*Link ID*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x0B	*Error ID*

Параметры аналогичны параметрам команды по добавлению лингвистической переменной.

2.3.5 Установка терма

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*4Б * Count*	*4Б * Count*
0x04	0x0C	*Term ID*	*Lingvo ID*	*Count*	*X coordinate*	*Y coordinate*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x0C	*Error ID*

Параметры аналогичны параметрам команды по добавлению терма.

2.3.6 Установка правила

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*1Б*	*1Б * Count*	*1Б * Count*
0x04	0x0D	*Rule ID*	*Out Lingvo ID*	*Out Term ID*	*Count*	*In Lingvo IDs*	*In Term IDs*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x0D	*Error ID*

Параметры аналогичны параметрам команды по добавлению правила.

2.3.7 Удаление лингвистической переменной

1Б	*1Б*	*1Б*
0x04	0x05	*Lingvo ID*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x05	*Error ID*

Lingvo ID – идентификатор удаляемой лингвистической переменной.

2.3.8 Удаление терма

1Б	*1Б*	*1Б*
0x04	0x06	*Term ID*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x06	*Error ID*

Term ID – идентификатор удаляемого терма.

2.3.9 Удаление правила

1Б	*1Б*	*1Б*
0x04	0x07	*Rule ID*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x07	*Error ID*

Rule ID – идентификатор удаляемого правила.

2.3.10 Получение лингвистической переменной

1Б	*1Б*	*1Б*
0x04	0x08	Lingvo ID

Ответ.

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*1Б*	*2Б*	*1Б*	*1Б*
0x04	0x00	*Error ID*	*Lingvo ID*	*Lingvo Type*	*Operation*	*Link addrress*	*Link endpoint*	*Link ID*

Lingvo ID – идентификатор лингвистической переменной, которую требуется получить.

2.3.11 Получение терма

1Б	*1Б*	*1Б*
0x04	0x09	Term ID

Ответ.

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*1Б*	*4Б * Count*	*4Б * Count*
0x04	0x01	*Error ID*	*Term ID*	*Lingvo ID*	*Count*	*X coordinate*	*Y coordinate*

Term ID – идентификатор терма, который требуется получить.

2.3.12 Получение правила

1Б	*1Б*	*1Б*
0x04	0x0A	Rule ID

Ответ.

1Б	*1Б*	*1Б*	*1Б*	*1Б*	*1Б*	*1Б*	*1Б * Count*	*1Б * Count*
0x04	0x02	*Error ID*	*Rule ID*	*Out Lingvo ID*	*Out Term ID*	*Count*	*In Lingvo IDs*	*In Term IDs*

Rule ID – идентификатор правила, которое требуется получить.

2.3.13 Получение списка идентификаторов лингвистических переменных

1Б	*1Б*
0x04	0x0E

Ответ:

1Б	*1Б*	*1Б*	*1Б*	*1Б * Count*
0x04	0x0E	*Error ID*	*Count*	*Lingvo IDs*

Lingvo IDs – список идентификаторов лингвистических переменных.

2.3.14 Получение списка идентификаторов термов

1Б	*1Б*
0x04	0x0F

Ответ.

1Б	*1Б*	*1Б*	*1Б*	*1Б * Count*
0x04	0x0F	*Error ID*	*Count*	*Term IDs*

Term IDs – список идентификаторов термов.

2.3.15 Получение списка идентификаторов правил

1Б	*1Б*
0x04	0x10

Ответ.

1Б	*1Б*	*1Б*	*1Б*	*1Б * Count*
0x04	0x10	*Error ID*	*Count*	*Rule IDs*

Rule IDs – список идентификаторов правил.

2.3.16 Установка статуса лингвистической переменной

1Б	*1Б*	*1Б*	*1Б*
0x04	0x03	*Lingvo ID*	*Status*

Ответ.

1Б	*1Б*	*1Б*
0x04	0x03	*Error ID*

Lingvo ID – идентификатор лингвистической переменной, статус которой требуется установить.

Status – устанавливаемый статус.

2.3.17 Очистка нечеткого контроллера

1Б	*1Б*
0x04	0x04

Ответ.

1Б	*1Б*	*1Б*
0x04	0x04	*Error ID*

2.3.18 Установка значения лингвистической переменной

1Б	*1Б*	*4Б*
0x02	*Lingvo ID*	*Value*

Ответ.

1Б	*1Б*	*1Б*
0x02	*Error ID*	*Lingvo ID*

Lingvo ID – идентификатор лингвистической переменной, значение которой требуется уставновить.

Value – устанавливаемое значение.

2.3.19 Получение значения лингвистических переменной

1Б	*1Б*
0x01	Lingvo ID

Ответ.

1Б	*1Б*	*1Б*	*4Б*
0x01	Error ID	Lingvo ID	Value

Lingvo ID – идентификатор лингвистической переменной, значение которой требуется установить.

Value – полученное значение.

Представленные бинарные команды полностью соответствуют командам в формате *XML*.

2.4 Сравнение команд в формате XML и команд в бинарном формате

Формат *XML* является текстовым форматом данных. Любой символ в *XML*-команде занимает один байт. Это означает, что цифра идентификатора или значения занимает один байт. Такой формат неэкономичен, но очень гибок, и позволяет без специальных инструментов, а просто в текстовом файле, составить команду для нечеткого контроллера.

Бинарный формат является гораздо более экономичным. В один байт команды можно записать идентификатор в диапазоне от 0 до 255. Для значения лингвистических переменных, а также точек термов используются числа с плавающей точкой (тип *float*, 4 байта). Для записи такого числа в бинарном формате требуется ровно 4 байта, для записи того же числа в формате *XML* может понадобиться много символов, а значит размер команды увеличится. Кроме того, в состав *XML*-команды входят тэги.

Для примера, рассмотрим объем команды по добавлению терма в формате *XML* и в бинарном формате.

Формат *XML*:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
<	a	t	>	<	l	t	>	4	<	\	l	t	>	<	l	>	4	<	\	l	>	<	n	>	3	<	\	n	>	<	x	>	0	<	\

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
x	>	<	x	>	5	0	0	<	\	x	>	<	x	>	8	0	0	<	\	x	>	<	y	>	0	<	\	y	>	<	y	>	0	.	5

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
<	/	y	>	<	y	>	1	.	0	<	\	y	>	<	/	a	t	>

- длина команды составляет 91 байт.

Бинарный формат:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
04	01	04	04	03	0	0	0	0	0	0	fa	43	0	0	48	44	0	0	0	0	0	0	0	3f	0	0	80	3f

- длина команды составляет 29 байт.

Ниже представлена таблица (см. табл. 1) сравнения длин команд для нечеткого контроллера в байтах в формате *XML* и в бинарном формате.

Таблица 1. Сравнение длин команд

Команда	*Размер бинарной команды (в байтах)*		*Размер команды в формате XML (в байтах)*	
	Запрос	*Ответ*	*Запрос*	*Ответ*
Добавление лингвистической переменной.	9	3	от 59	от 19
Добавление терма.	от 13	3	от 51	от 19
Добавление правила.	от 8	3	от 63	от 19
Установка лингвистической переменной.	9	3	от 59	от 19
Установка терма.	от 13	3	от 51	от 19
Установка правила.	от 8	3	от 63	от 19
Удаление лингвистической переменной.	3	3	от 17	от 19
Удаление терма.	3	3	от 19	от 19
Удаление правила.	3	3	от 17	от 19
Получение лингвистической переменой.	3	от 3	от 17	от 19
Получение терма.	3	от 3	от 19	от 19
Получение правила.	3	от 3	от 17	от 19
Получение списка идентификаторов лингвистических переменных.	2	от 4	9	от 27
Получение списка идентификаторов термов.	2	от 4	9	от 27
Получение списка	2	от 4	9	от 27

Команда	*Размер бинарной команды (в байтах)*		*Размер команды в формате XML (в байтах)*	
	Запрос	*Ответ*	*Запрос*	*Ответ*
идентификаторов правил.				
Установка статуса лингвистической переменной.	4	3	от 25	от 19
Очистка нечеткого контроллера.	2	3	9	от 19
Установка значения лингвистической переменной.	5	3	от 25	от 19
Получение значения лингвистической переменной.	2	от 3	от 17	от 27

Из таблицы видно, что длина команды в *XML*-формате может быть больше длины той же команды в бинарном формате в 7-8 раз.

Энергетические ресурсы узла сети существенно ограничены. Причем, основная энергия тратится на передачу и прием данных по радио, следовательно, передача длинных команд значительно сократит время жизни сети. Кроме того, в существующих реализациях сетевых протоколов для беспроводных сенсорных сетей максимальная длина массива байт, который может быть передан по радио, ограничена.

Таким образом, для обеспечения гибкости универсальных интерфейсов для беспроводных сенсорных сетей и обеспечения долгосрочной работы самой сети, необходимо использовать *XML*-формат команд управления нечетким контроллером для взаимодействия клиентского приложения и корневого узла сети (то есть узла сети, связанного с компьютером), а для передачи данных внутри сети использовать бинарный формат команд.

XML-команда будет передаваться корневого узлу по *COM*-порту, разбираться с помощью парсера и переводится в бинарный формат, а затем пересылаться по радио узлу с установленным нечетким контроллером.

С дальнейшим развитием устройств для беспроводных сенсорных сетей и усовершенствованием программного обеспечения эти ограничения могут быть сняты.

Заключение

При реализации приложений для беспроводных сенсорных сетей возникает две проблемы. Первая проблема – это проблема создания универсальной формы представления передаваемых между клиентом и сетью и внутри сети данных. Для решения этой проблемы необходимо создать универсальный язык общения высокого уровня, позволяющий добиться гибкого механизма взаимодействия сенсорных сетей и внешних приложений.

Вторая проблема заключается в возможности переформулирования задачи, выполняемой сенсорной сетью. При стандартном подходе такое изменение задачи требует перепрограммирования всех узлов сети. Решением проблемы является реализация на узле сети системы, способной изменять стратегию функционирования в зависимости от решаемой задачи без изменения алгоритма работы.

Наиболее эффективной моделью такой системы является нечеткая продукционная система, осуществляющая нечеткий логический вывод. Нечеткая продукционная система основана на теории нечетких множеств.

Реализацией нечеткой продукционной системы является нечеткий контроллер.

Сенсорный узел с установленным на нем нечетким контроллером становится «умным» сенсорным узлом. Использование «умного» сенсорного узла и языка высокого уровня для представления данных и знаний для узла позволяет решить сформулированные выше проблемы и создать универсальный интерфейс высокого уровня для беспроводной сенсорной сети.

Для решения проблемы создания универсальной формы представления передаваемых данных и правил удобно использовать язык разметки на базе *XML*. Он позволяет беспроводной сенсорной сети независимым от платформы способом взаимодействовать с внешними приложениями.

Для работы с языком *XML* сформулирована модель передачи данных и знаний и определены требования для *XML*-парсера, используемого для разбора *XML*-сообщений.

Парсер встраивается в модель «умного» сенсорного узла.

Несмотря на удобство использования команд по управлению нечетким контроллером в формате *XML* возникает проблема большого объема передаваемых данных при использовании такого подхода. Это связано с тем, что формат *XML* – это текстовый формат.

Для решения этой проблемы проведено исследование, в ходе которого предложен бинарный формат команд по управлению нечетким контроллером и осуществлено сравнение объема передаваемых данных в формате *XML* и в бинарном формате.

В результате исследования можно сделать вывод, что для взаимодействия клиентского приложения и беспроводной сенсорной сети эффективно использовать *XML* формат для представления данных, а при взаимодействии узлов внутри сети – бинарный формат для представления данных.

Список литературы

1. Averkin A. Technology of Smart Sensor Nodes for intelligent decision-making and information processing support in WSN.
2. Tampalini F., Cassinis R. Fuzzy logic controller based on XML formatted files for behaviour-based mobile robots // Technical Report, DEA-Unibs. – 2005.
3. Atanasova T. XML view of fuzzy system knowledge // International workshop on Personal Computers and Particle Accelerator Controls PCaPAC, Hamburg, Germany. – 2000.
4. Makatchev M., Tso S.K. Human-Robot Interface Using Agents Communication in an XML-Based Markup Language // Robot and Human Interactive Communication. – 2000.
5. Soto A. R., Capdevila C. A., Fernandez E. C. Fuzzy Systems and Neural Networks XML Schemas for Soft Computing // Mathware Soft Comput. – 2003. – №. 2-3. – Pp. 43-56.
6. Поспелов Д.А. Нечеткие множества в моделях управления и искусственного интеллекта. – М.: Наука. Гл. ред. физ.-мат. лит., – 1986. – С. 312.
7. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2001.