

A distributed sensor network

By Sebastian Dransfeld

Abstract

Sensor networks in manufacturing business are growing rapidly. More and more information is gathered for processing, and decision-making. To be able to keep up, sensors have to become more advanced and be easier to deploy on the network as independent instances. This can be made possible by using the ideas and technologies behind web-services, and new powerful cheap micro-controllers. In addition, a web interface eliminates the need for dedicated control software.

Introduction

To gather complete and detailed information has become a necessity in modern production. Production processes are highly automated, with little human intervention. Since automated machinery has a high production rate, and the margins for errors are small, the information from such systems must come from sensors mounted on or near the process, and the information has to be handled by computers. No human can register all necessary information with the speed and accuracy needed.

To solve the problems with information gathering, processing and storage, it is necessary to build up a highly advanced network with many sensor nodes. Today the most common type of sensor network is a simple master/slave network, where one master computer polls many slave sensors. Direct connection between sensors isn't feasible. What we need is a more advanced network where each sensor (or group of sensors), is a separate unit. This unit should work independently and provide useful information on request, not just raw data. Since the advanced sensor should work independently, it also shouldn't be necessary for an operator to use a local program to check the status. It should rather be a human accessible interface implemented into the sensor.

This kind of network with nodes that work separately and which has a uniform connection interface, both for human and computer interaction, exist today. We call it the Internet. The technologies behind the Internet are old and have proven themselves to be very useful. In addition there are many people capable of doing programming with these technologies, and the hardware has become very inexpensive.

So the solution to our information need in production facilities, might be to create an intranet with advanced sensors; a distributed sensor network.

Smart sensors

A sensor is a small device, which can register many different properties, like temperature, distance, presence etc. The information is usually translated to a voltage, or a current, which is transmitted onto a connected cable. The signal is pushed continuously, so it is difficult to have several sensors connected to one cable. To build up a network with such analogue sensors is quite difficult and demanding. So these dumb sensors have evolved into digital sensors.

A digital sensor reads the analogue signal and translates it into a digital with an A/D converter. The signal can be either high or low (for example present or not), which can be represented by one bit, or a continuous (for example. a temperature), which may require many bits. These digital sensors usually don't push the information onto their connection, but they can wait for an activation signal before they transmit.

Even though digital sensors are much more advanced and capable for networking than analogue sensors, they still aren't good enough for our demands. They only transmit raw data, and they can't have advanced activation functions. So we need to develop the advanced sensor, or a "smart sensor". This means that we connect the digital sensor to a micro-controller capable of doing more advanced calculations. The micro-controller can also be connected to input devices (keyboard, pointer) and to output devices (screen, network). This smart sensor should be able to gather information from several sources, process the information and make decisions based on the information. In addition it has to be connected to an I/O system so that humans and computers easily can read and write information to it.

The information processing is split into parts, the first is processing and understanding, the second is transmission. As stated earlier, digital sensors return a number. But this number can represent many things, like “the temperature is XX degrees” or “the transportation belt is running”. With a normal digital sensor we have to decode this signal upon arrival, but with a smart sensor we demand that this decoding happens in the sensor. After such decoding the information has usually grown in size, and it has a more complicated structure. To transmit a complicated data structure over a network demands much more than sending a simple number. You have to be sure that both the receiver and the transmitter have the same understanding of the data. In addition it should be a general transmission format, since we desire to gather different data, which requires different representation structures on our distributed sensor network.

The requirements for a “smart sensor” demand that it has sufficient processing power and enough memory. In addition it has to be able to do advanced network communication. Today it is possible to purchase advanced 8-bit micro-controllers with several megabytes of memory for a few dollars. Since the input might be a few bytes from one or more digital sensors, it should be sufficient. In the following sections I will look at the possible technologies for networking and communication and see if such a micro-controller has the power to run the necessary services.

Network

Now we have defined a "smart sensor" which should be able to do network communication. But what kind of network? At the shop floor in a production plant field-buses are very common. These field-buses are often designed with a digital sensor in mind. This means that they are often implemented as a serial-bus, with no possibilities of advanced network structures. In one end the sensor, on the other end a computer polling the sensor. If the field-bus allows more advanced networks, they often only allow one master at a time, which can poll each digital sensor one at a time, or transmit information to one sensor (or all). In addition these field-buses are proprietary, which makes it more complicated to build a general network structure with a general network protocol.

We want to create a large network system with many interconnected nodes, where all nodes should be able to communicate with each other. As mentioned in the introduction, the Internet is such a network. The hardware layer in the backbone of the Internet is the Ethernet. Ethernet has proven itself fast and reliable, much faster than any field-bus; up to 1 gigabit/second vs. speed just above 1 megabit/second. Ethernet was designed with the purpose of having many interconnected computers. In the early days this benefit was a bit problematic, because of the medium access strategy called Carrier Sense Multiple Access with Collision Detection. This means that all nodes listen to the network, and if there's no traffic, they can start submitting. If two nodes sent information at the same time, the data would collide, and all submitting nodes would have to resend their data. Nowadays this problem has almost disappeared, partly because of the high-speed implementations, but mostly because of intelligent switches that divide the path between two nodes into small steps.

To really exploit the benefit of Ethernet, the sensor network must also use the most common protocols that are used. The lowest level is the Internet Protocol (IP), and on top of it the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). To get a functional Intranet other protocols must be implemented as well (like ICMP and ARP), but these are used for controlling the network and aren't interesting for the application programmer. TCP/IP is a stream-based protocol that keeps track of the data sent between two hosts to guarantee delivery in the correct order, whilst UDP/IP just sends packets without tracking. UDP is faster since it doesn't require much checking, but TCP is better when sending large amounts of data.

The different Internet protocols aren't designed with small, fast, real-time networks in mind. They have a quite large overhead (the ratio between the total amount of data sent and the amount of useful data), but studies show that because of the higher speed of Ethernet, Ethernet can compete well with more specialized networks.

Web services

The "smart sensor" is now connected to an Ethernet, and using the Internet protocols. In the introduction I wrote that the advanced sensor should be able to provide useful information on request, and that the provided information should be understandable

by both human and computer. To pass information on a network, binary protocols are the most efficient. Standards like Java RMI and CORBA provide the framework for advanced network interaction, but there are drawbacks. RMI is dependant on Java, and CORBA is very complex. And the biggest problem is that binary data is only suitable for computers, not for the human eye. Since the network is fast, and the micro-controllers have quite a lot of resources, a text-based interface should be used.

One of the most hyped buzzwords in Internet is “Web services”. Large software companies like IBM, SUN and Microsoft see web services as the future, and they provide server software like Microsoft’s .NET, SUN’s ONE and IBM’s WebSphere. These products are large software packages requiring large computers with fast processor and loads of memory. For our system with 8-bit micro-controllers with 1 megabyte of memory this isn’t a viable solution. But the principles are. The technologies behind web services give us a standard communication interface for deploying several small servers on the net with a text-based interface understandable for both humans and computers. Equipment manufacturers can easily develop user interfaces on their smart sensors. Then a buyer can purchase a sensor, install it in the manufacturing process, connect it to the office LAN and then read the status with a standard web browser. No need for specialised software that only understands one vendor’s equipment.

The basis of web-services is HTTP, or the HyperText Transport Protocol and XML. HTTP is the basis of the world-wide-web and was designed to send simple formatted text documents between a server and a client. At the client a user would sit and get the document up in a web-browser. With web-services HTTP gets exploited to its limit. HTTP will be used to send **all** information between nodes, not just plain HTML, by using the possible to direct requests (by using the path) and to define the content (with Content-Type). The advantages of HTTP are many. It is well known, and has been used for a long time, and it has a lot of well-defined responses, which makes it easy to debug errors. XML is eXtensible Markup Language, which means that it uses tags to wrap up data. A start-tag is enclosed by <>, and the end-tag is enclosed by </>. The real benefit of this tag-structure is that it is easy to create a nested structure, and thereby making it possible to send almost all information in a logic and practical structure. The backside is that the tags require much space.

Smart sensor implications

It seems like Ethernet, Internet protocols and a HTTP server would be a good solution for the smart sensor interface. The smart sensor is a device that must be small, cheap and use low amounts of power. In a medium sized factory with a high level of automation, hundreds of smart sensors could be required.

On the market today there exist a wide variety of 8-bit micro-controllers with maybe 1 MB of memory. Mostly they are used to run tight assembly coded loops for real-time operations. On a small device like that it isn't possible to install a large operating system and web-server. But these small micro-controllers have become very fast, therefore they can handle the overhead of using text instead of binary data.

To implement a TCP stack and a HTTP server on such a micro-controller seems like a task to large to be useful. Most probably is the information gathered from the attached sensors just a few bits. But an implementation of a TCP stack and a HTTP server is a general interface. All different kinds of sensors could use the same interface, so the work must only be done once, which was one of the ideas behind using Ethernet and Internet protocols in the first place.

So instead of using resources to create a wide variety of interfaces and sensors which do their own thing, we use the resources to create a communication interface which should be sufficient for a wide variety of products. The result is that we can create a human interface against the smart sensor, and it is easy for central servers to communicate with it to.

Web services communication

There are two ways of communicating with a HTTP server. The first is GET/POST, which is implemented in the HTTP protocol, and the other is to send data in the message body, which the receiving HTTP server has to decode.

GET and POST

GET and POST are defined in the HTTP standard. GET is a normal request for data on the web-server, but it is possible to attach information to the URL. The request would be like:

```
GET /info.html?foo=bar&info=tull HTTP/1.0
```

The transferred information can be divided into two parts, '/info.html' and 'foo=bar&info=tull'. The first part tells us which document is requested and where it resides, and the other part defines two variables, foo and info, and assign values to them. Variables are divided by '&' and values are assigned by '='.

With POST data is sent in the message body. This makes parsing the request more difficult because all of the sent message must be parsed. With GET only the first line of the request has to be parsed. Standard POST defines that a message string with name/value pairs separated by & should be sent. So the result is the same as with GET.

HTTP POST example:

```
POST /info.html HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 17

foo=bar&info=tull
```

As we see, we have to define what kind of information we are sending, and how much.

Advanced methods

With GET and POST it is hard to send complex data. All information has to be in (name, value) pairs. To solve this problem, two protocols have been built upon HTTP, XML-RPC and SOAP. Both use XML to wrap up the data, the difference lies in their complexity. SOAP is designed to be able to do anything, whilst XML-RPC is designed to be easy.

A standard XML-RPC request is defined by a name and data. The data can be composed by simple datatypes as strings, integers, reals, etc., or complex types as arrays and structs.

XML-RPC request:

```
POST /RPC2 HTTP/1.0
```

```
Content-Type: text/xml
```

```
Content-length: 181
```

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value>
        <i4>41</i4>
      </value>
    </param>
  </params>
</methodCall>
```

XML-RPC response:

```
HTTP/1.1 200 OK
```

```
Connection: close
```

```
Content-Length: 158
```

```
Content-Type: text/xml
```

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <string>South Dakota</string>
      </value>
    </param>
  </params>
```

```
</methodResponse>
```

XML-RPC errormessage:

```
HTTP/1.1 200 OK  
Connection: close  
Content-Length: 426  
Content-Type: text/xml
```

```
<?xml version="1.0"?>  
<methodResponse>  
  <fault>  
    <value>  
      <struct>  
        <member>  
          <name>faultCode</name>  
          <value>  
            <int>4</int>  
          </value>  
        </member>  
        <member>  
          <name>faultString</name>  
          <value>  
            <string>Too many parameters.</string>  
          </value>  
        </member>  
      </struct>  
    </value>  
  </fault>  
</methodResponse>
```

Further evolution

Now I have described the basis to create “smart sensors” with a web interface, and I have short described possible communication interfaces (XML-RPC, GET/POST). This makes the “smart sensors” accessible on an intranet by all computers. The next step is that all nodes on the network interact with each other, but not through a central server. If a node wants to know if the previous step in a production line has completed correctly, it should ask the previous node, not a central computer. This reduces the

amount of information on the network, and the information will come directly from the source.

To make it easier to administrate the different nodes on the network, the information about what a node can do should be implemented in the node. If a new node is connected to the network, the central computer should be able to ask the node about its information and store it in a database. If then another node requests some information that it needs, the central server can do a lookup in the database and redirect it to the right source. These ideas have their implementation under the web services umbrella.

The description document about what a node can do is a XML document conforming to the Web Services Description Language (WSDL). In this document all relevant information is stored. To make the lookup on all the nodes, and to store the information about them, the standard is called Universal Description, Discovery and Integration (UDDI). The UDDI service is probably very complicated, but it just has to run on one central computer in the network, and most modern PC's can handle the workload. The WSDL documents just reside on the smart sensors, and they return them on request.

Conclusions

Modern manufacturing plants have a desperate need for large amounts of sensor information, to be able to refine and improve their production. (Which is necessary to keep up with the competition). As the amount of sensors grow, the sensor network grows to. To be able to create large advanced sensor networks, technologies like Ethernet and Internet protocols should be used because they have proved that they work with large complicated networks.

As the network grows, the sensors must mature to. They can't be small simple devices, which pours their information out on the network in an uncontrolled manner. They have to be "smart" and do their I/O in an ordered fashion. Also, they should be directly accessible by a human interface.

Since it also should be easy and quick to develop such devices, they have to use a standardised interface. To look at such a smart sensor as a web-service seems to be a good solution. It has a large overhead, and requires fast micro-controllers, but the simplicity in developing such devices (after they have their TCP stacks and HTTP servers) makes it worthwhile.

References

- Blake Lloyd, Mark Susnik: *Web embedded Field Devices*
- A. Flammini, P. Ferrari, E.Sisinni, D. Marioli, A. Taroni: *Sensor interfaces: from field-bus to Ethernet and Internet*
- A. Flammini, P. Ferrari, E.Sisinni, D. Marioli, A. Taroni: *Sensor integration in industrial environment: from field-bus to web sensors*
- XML-RPC, <http://www.xmlrpc.com/>
- WSDL, <http://www.w3.org/TR/wsdl>
- UDDI, <http://www.uddi.org/>