1. Subsystems Integration - Introduction

- The fundamental is to achieve an effective means to operate the business, such as an “effective building”.
- Successful integration will reduce the life cycle cost of the facilities.
- Proper integration can also improve the performance of people.
- Integration can produce outcome with new implication.
- Traditional integration may include subsystems in automation and control, such as HVAC, lighting, Fire, security, and other building’s subsystems.
- Integration of building systems to data system open another dimension.
  - Systems include BAS, business information system, voice system, video system, etc. each reside on a common, integrated information infrastructure.
  - Can correlate a particular incident with its related history of influential data.

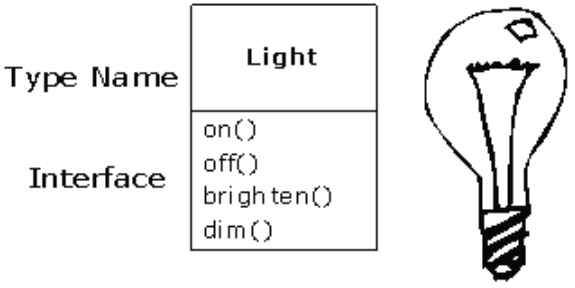
2. Field Controller Levels Integration – "What" to integrate

- The evolution of license free or royalty free protocols accelerate the integration practices.
  - Example: Modbus, OptoMUX, OPC.
- Popularity of Object Oriented approach, gave birth Object Oriented Protocols
  - Examples: BACnet's Objects & Services, LonMark Functional Profiles, EIB Object Interface.
- In each of these OO-Protocols,
  - Related pieces of information are stored together as object.
  - Multiple objects can be grouped together into another object in a hierarchical fashion.
  - Objects can be viewed as "service providers".
  - To make a request of an object, you "send a message" to that object.
  - Message is a request to call a "method" that belongs to a particular object.
  - Object has "states" represented by its "properties".

Object manipulation example (General)

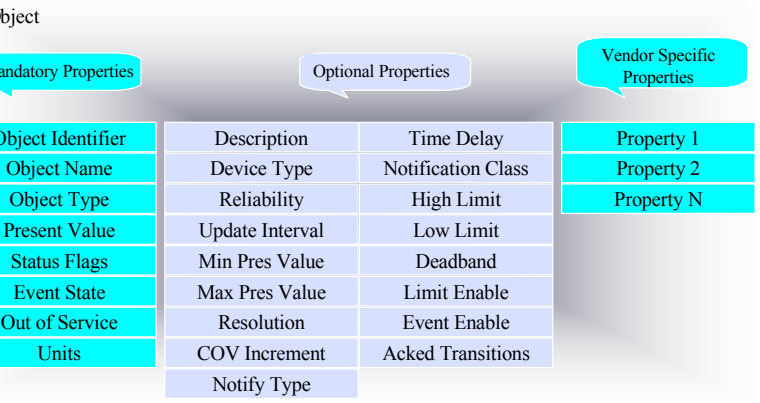
- The name of the type/class is Light.
- the name of this particular Light object is lt.
- The requests that you can make of a Light object are to “turn it on”, “turn it off”, make it “brighter”, or make it “dimmer”.
- You create a Light object by defining a “reference” (lt) for that object and calling new to request a new object of that type.
- To send a message to the object, you state the name of the object and connect it to the message request with a period (dot).

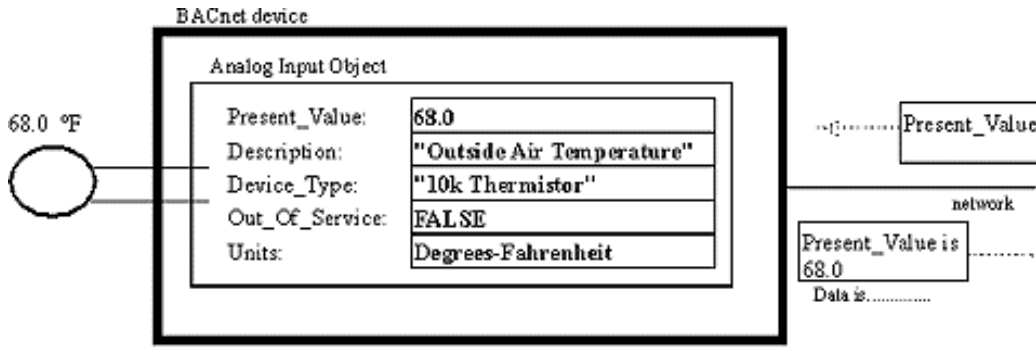
```
Light lt = new Light();
lt.on();
```



- In this example, if Light has property “color”,
  - Lt.on.color = red; //may be used to change Light color properties

BACnet - as an example



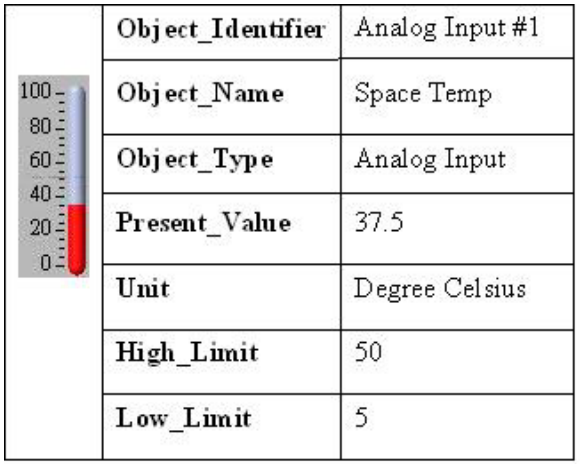


- For BACnet equipment to work together, components must be able to exchange and understand BACnet messages.
- BACnet defines a standard set of "*Objects*", each of which has a standard set of "*Properties*".
- Properties describe the Object and its current status to other devices on the BACnet internetwork.
- It is through these properties that the Object may be controlled by other BACnet devices.

BACnet object example - Analog Input Object

- One of the standard BACnet objects is the Analog Input Object
- Above diagram represents an analog sensor input such as a thermistor as it might be seen over the network through five of its properties.
  - Present_Value and Out_Of_Service, provide status about the sensor input represented by the Analog Input Object.
  - Some of the other properties-such as Description, Device_Type and Units-are set during installation.
  - Yet others (an Analog Input Object can have up to 25 Properties) may be set by the equipment manufacturer.
  - All may be read; in this example, a query about the Present_Value Property of this Analog Input Object would get the reply "68.0".
- Each element of a complete building control system is represented by one or more of the Objects, whether “Analog Input Object” for a sensor, a “Schedule Object” for scheduling, or “Notification Class Object” for distributing alarms.
- The choice of which Objects are present in a BACnet device is determined by the device's function and capabilities.
- Every BACnet device must have a “Device Object”, the Properties of which fully describe the BACnet device to the network.

BACnet Object Identification



- The Object_Identifier is a 32-bit code that identifies the type of Object (also identified by the Object_Type Property) and its "*Instance*" number, which together uniquely identify the Object within its BACnet device.
  - Example: Object_Identifier - Analog Input #1
- BACnet requires one *Device Object* to be present in every BACnet device.
  - The Device Object makes information about the device and its capabilities available to other devices on the networks.
  - Unlike other Objects, the Device Object's Instance number must be unique across the entire BACnet internetwork because it is used to uniquely identify the BACnet devices.

BACnet Services

- *Services* are the means by which one BACnet device acquires information from another device, commands another device to perform some actions, or announces to one or more devices that some event has taken place.
- An "*application program*" running on the BACnet device issues service requests and processes them upon receipt.
- BACnet defines 32 Services and classifies them into five categories.
  - These Service categories are the Alarm and Event, File Access, Object Access, Remote Device Management and Virtual Terminal Services.
  - For example, the Object Access Services provide the means to read, modify and write Properties, and to add and delete Objects.

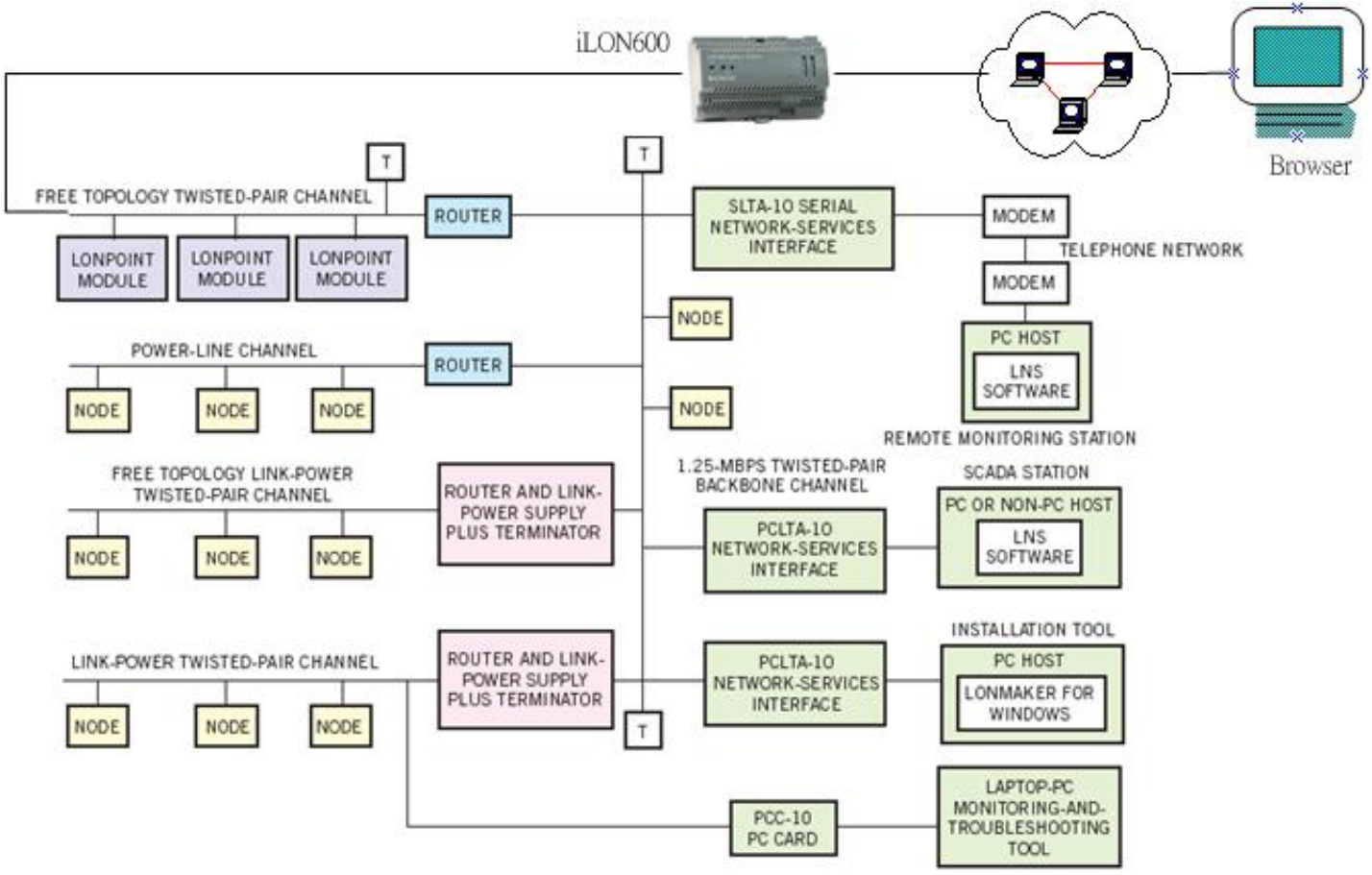
Refer to Lecture notes pg.4 for BACnet coding example

LonWorks – as an example

	OSI Layer	Purpose	Services
7	***Application***	Application Program	Standard objects & types, config props, file xfer, network services
6	***Presentation***	Data Interpretation	Network variables, application messages, foreign frames
5	***Session***	Remote Actions	Dialog, remote procedure calls, connection recovery
4	***Transport***	End-to-End Reliability	End-to-End acks, service type, pkt sequencing, duplicate detect
3	***Network***	Destination Addressing	Unicast & multicast, destination addressing, packet routing
2	***Data Link***	Media Access & Framing	Framing, data encoding, CRC, media access, collision detect
1	***Physical***	Electrical Interconnect	Media specific details, xceiver type, physical connect

- **The main elements in interoperable integration are**
  - SNVT, SCPT, LonMark Objects & Functional Profiles

LonWorks - Standard Network Variable Types (SNVT)



SNVTs are transferring across these links.

SNVT

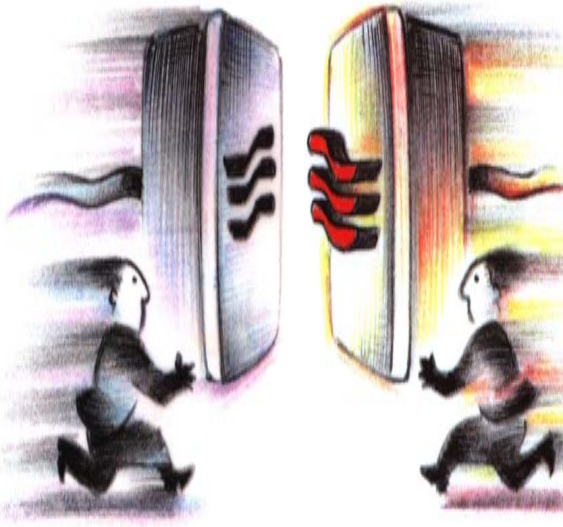
- SNVTs are the are the keys to device interoperability and allow you to logically bind sensors and actuators in a LonWorks network.
- You can declare SNVTs in Neuron C similarly to int, char, or other variable types. *(remember what you had studied in uP applications module?)*

e.g. network output sd_string("@2|1") SNVT_occupancy nvo01Occupancy={OC_UNOCCUPIED};

SNVT_temp	Temperature	Degrees Celsius	-274..6271	0.1 degree
SNVT_angle	Phase/Rotation	Radians	0 - 65	0.001 rads
SNVT_speed	Speed	Meters/Sec	0 - 6553	0.1 m/s
SNVT_elapsed_tm	Elapsed Time	HH:MM:SS	0 - 65535 days	1 msec
SNVT_lev_cont	Continuous Level	Percent	0 - 100%	0.5%
SNVT_ascii	ASCII String	Characters	30 Chars	N/A
SNVT_count	Events	Count	0 - 65535	1 Count

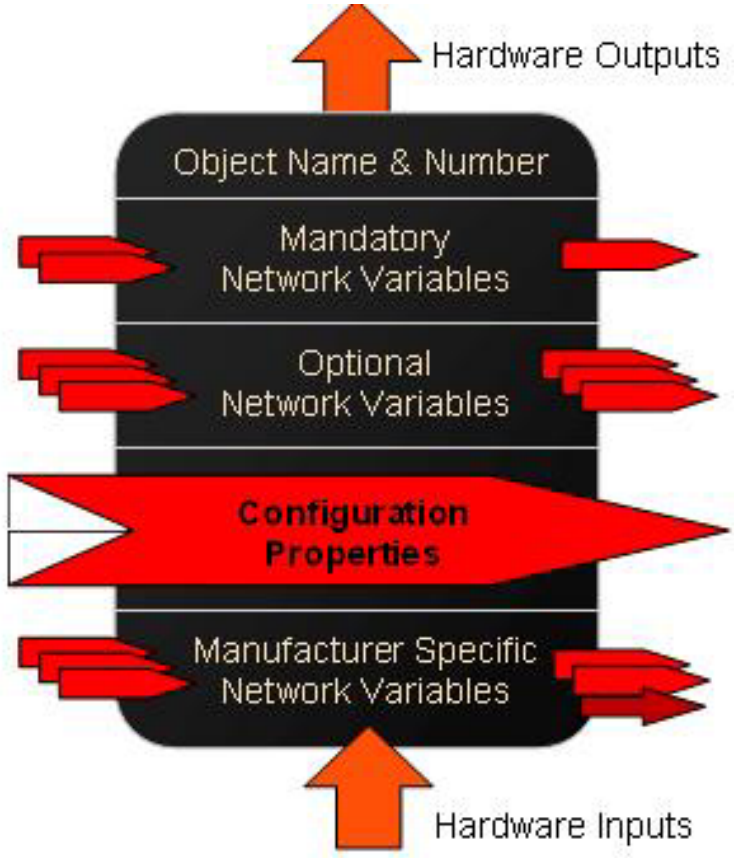
Device Interoperability with LONMARK Objects

- Standard method for device self-documentation
  - Manufacturer, model number, product version
- Specify functional behavior
  - Generic sensor, actuator or controller
  - Industry-specific Functional Profiles
- Allow device plug-n-play among manufacturers



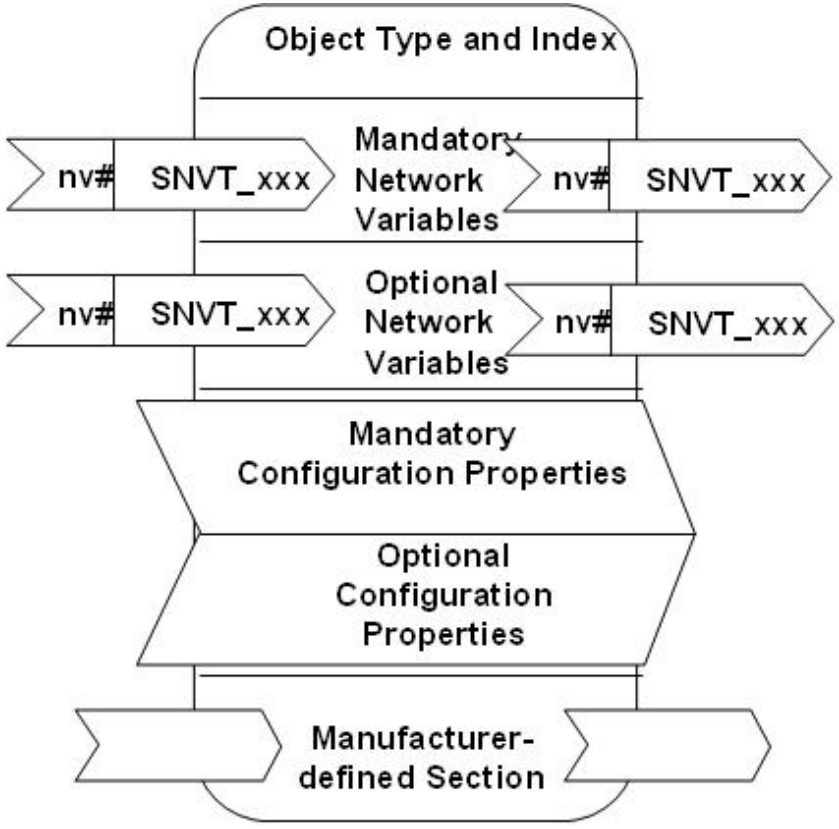
LONMARK Devices & Functional Profiles

- Have a standard “logical” interface
- Group network variables and configuration information into LONMARK Objects
- Use Functional Profiles whenever possible– (application-specific LONMARK Objects)
- Interoperate with other LONMARK Devices
- Ease installation, configuration and maintenance



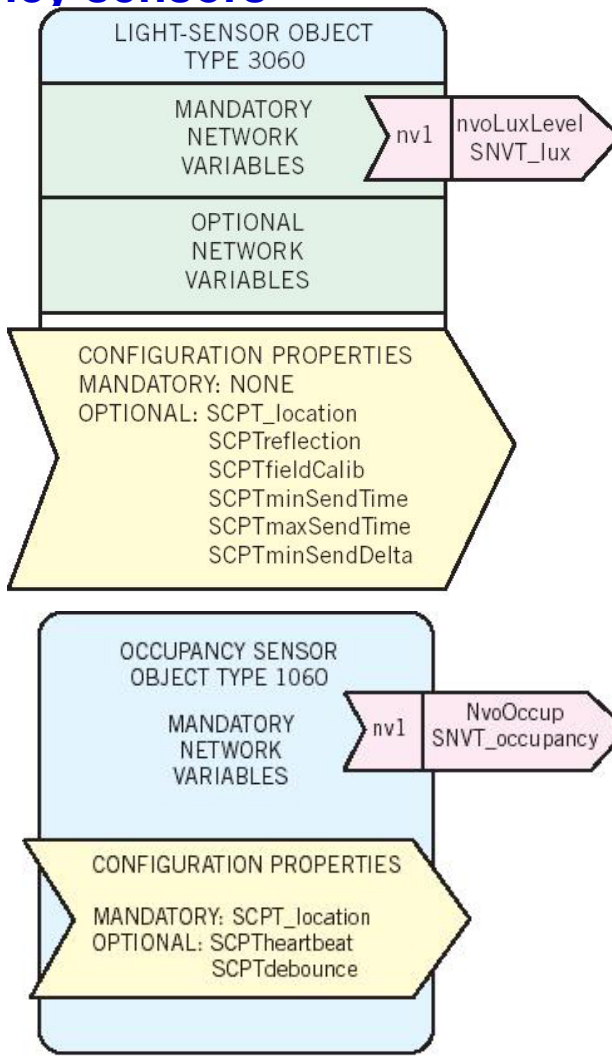
LONMARK Object Features

- **Type of object**
- **Index on device**
- **Mandatory Network Variables**
  - Minimum implementation
  - Use SNVTs
- **Optional Network Variables**
  - Implemented in standardized manner
  - Use SNVTs
- **Configuration Properties**
  - Applies to device, object or network variable
  - Can be implemented by SNVTs
- **Manufacturer-defined section**
  - Manufacturer-defined network variables and types
  - Proprietary, non-interoperable interface



Example design – Light and Occupancy sensors

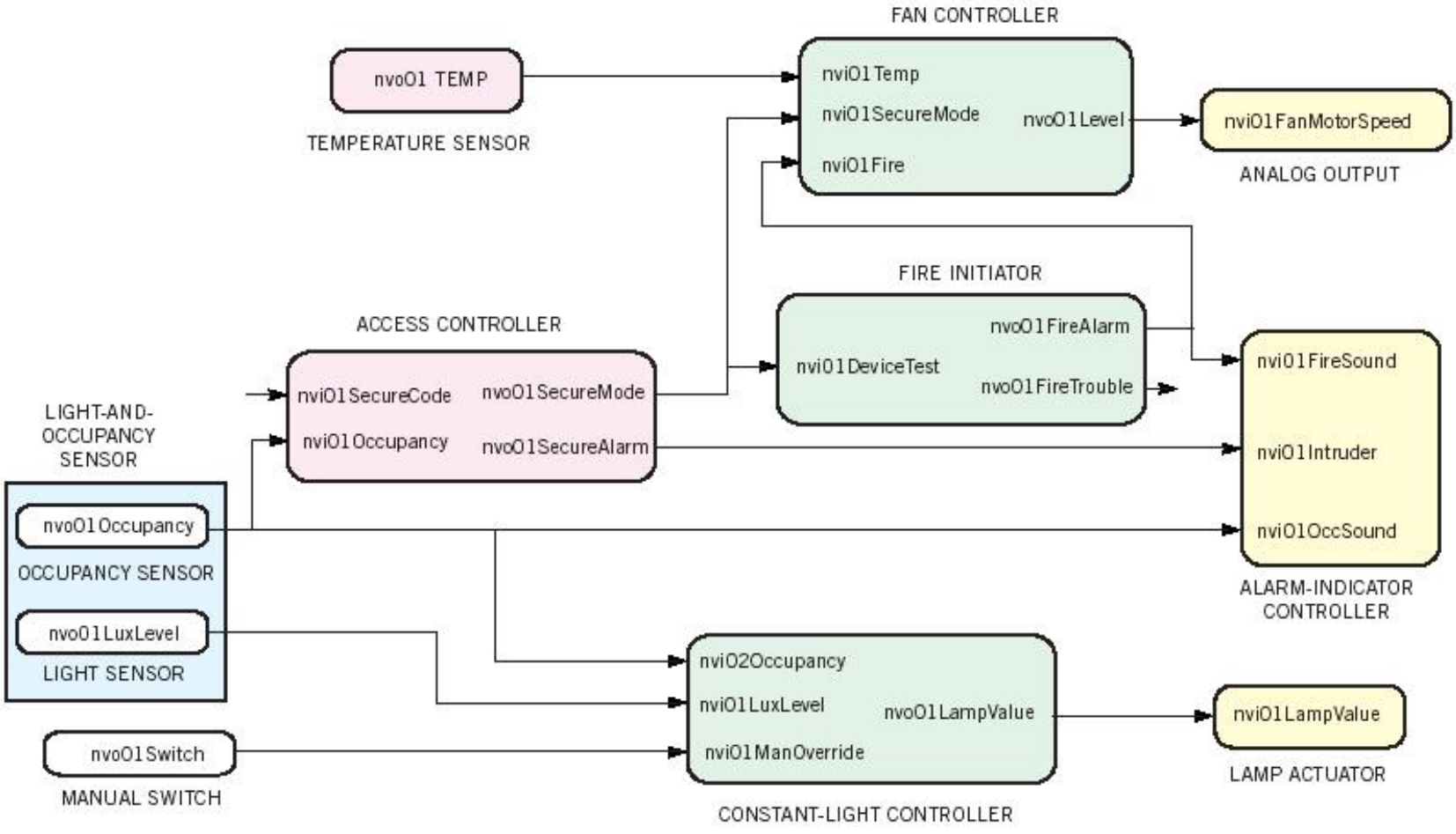
- The design is according to the published Functional Profiles for Light sensor and Occupancy sensor as shown.
- Neuron C program typically starts
  - with variable and I/O object declarations,
  - proceeds with perhaps some initialization instructions,
  - and then concludes with most of the program consisting of a series of “when” clauses.
- Example coding is shown on next slide.



```
int Go_occupancy = 1; //1= go with update, 0=debounce period
network output sd_string("@2|1") SNVT_occupancy nvo01Occupancy={OC_UNOCCUPIED};
network output sd_string("@1|1") SNVT_lux nvo01LuxLevel;
// Optional Light Sensor FP Configuration Property Declarations
config network input sd_string("&2,2,0\x80,52,0:65534") SNVT_time_sec
    nci01MinSendT={10};
config network input sd_string("&2,2,0\x80,49,0:65534") SNVT_time_sec nci01MaxSendT=
    {600};
IO_7 input bit io_Occupancy; //Occupancy sensor IO declarations  // I/O model function
IO_4 input quadrature io_dial;//light sensor IO declarations          // I/O model function
when (io_update_occurs (io_dial))
{
  static signed long tempLux;
  tempLux += input_value;
  currentLuxLevel=tempLux * 5; //scale reading
 if (Go_lux) //has minimum send time expired
{
  if (currentLuxLevel >=1000) nvo01LuxLevel=1000;
  if (currentLuxLevel <=0) nvo01LuxLevel=0;
  if ((currentLuxLevel >0) && (currentLuxLevel <1000))
     nvo01LuxLevel=currentLuxLevel;
 }
}
```

Integrate among different devices

- Once you have a working LonWorks devices, you create network-variable links using a connector between LonWorks objects. (Use LNS tools, e.g. LM4W)

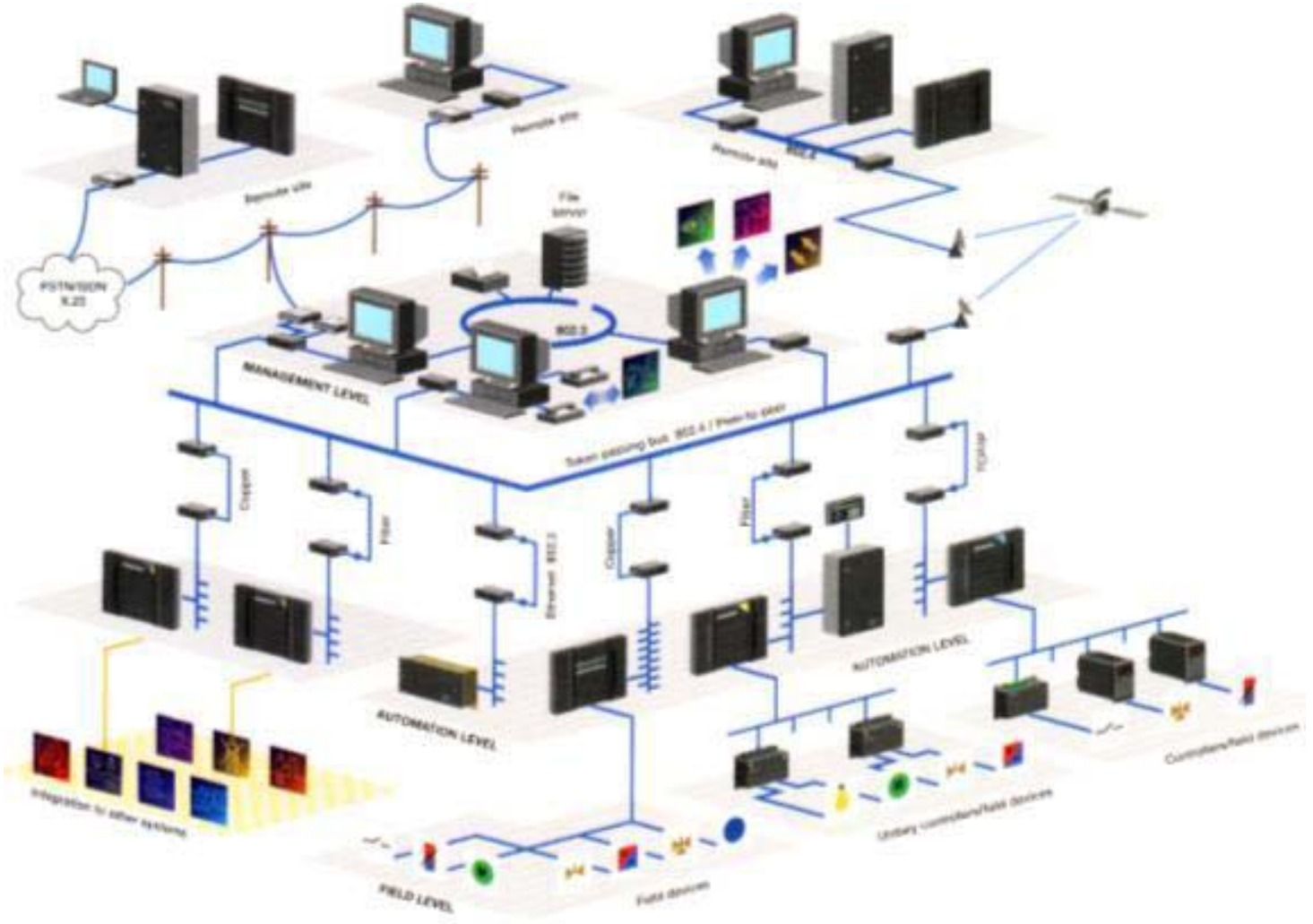


3. Integrating different Protocols – "HOW"

- In the 1980's, there wasn't a protocol that could meet the broad needs of various control industries.
- As we press forward into the 21st century, our problem is not one of "not enough" but one of "too much.".
- We've just seen several protocols that can meet the needs of an average building automation system.
  - however, integrating more than one protocol into a single system can be a challenge.
  - Although the data structures for the standard protocols are similar (e.g. OO approach), their implementations are quite different.
  - Gateways between any two of the standard protocols tend to be complex and less flexible, and bridging more than two can become unwieldy, as seen from the following figure for a typical automation system.

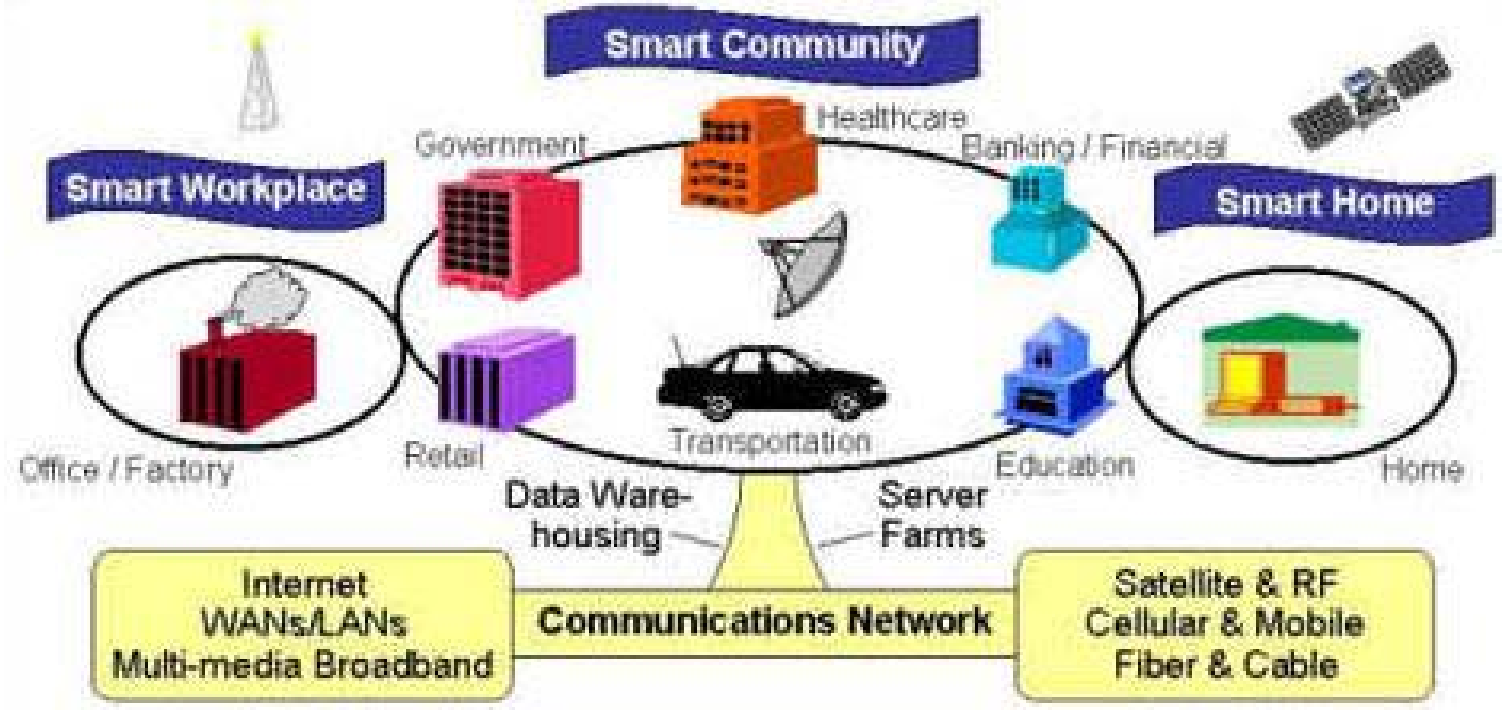
Control Network ENG3502

Typical Automation System to serve different subsystems functions



Migration needs – control data and information system data

- Automation systems are increasingly seen as a part of a much larger information system.
- The requirements for "information integration" are now much broader than when the standard protocols and their corresponding data models were conceived.



Unified model – Through Management Level Integration

- what is needed is a unified information model to include these protocols as well as other automation-related models.
- Web Services type of development may be the way forward
- Web Services
  - Is a self-contained, modular applications that can be run over the Internet and can be integrated into other applications.
  - Allow any authorized client to actually run an application on the host computer and generate data that didn't previously exist.
- If Web Services is to serve as an integration platform
  - It has to be able to combine content from many different sources.
  - It has to serve all sorts of devices, platforms, and browser types, delivering content over a wide variety of connection types for a wide variety of purposes.

What's in Web Services

- For Web services to address all of these needs, flexible technologies are crucial: XML(eXtensible Modeling Language) and SOAP(Simple Object Access Protocol) .
  - An XML document includes a definition of the data structure together with the raw data, so the receiving computer knows what information is contained in which fields. XML is basically a file format.
  - SOAP is a way of using XML over a network. SOAP provides a computer application with a tool that can read the data definitions in an XML document and extract the required data.
  - SOAP is to XML what HTTP is to HTML.

Simple example – email service using HTML

To: ssptang@vtc.edu.hk

From: tslam@vtc.edu.hk

CC:

Subject: Turn off the light

It's Friday. Don't forget to turn light off in B110.

When saved as HTML format:

```
<html>
    <title>Email</title>
    <body>
        <b>To:</b> ssptang@vtc.edu.hk<br>
        <b>From:</b>tslam@vtc.edu.hk<br>
        <b>CC:</b><br>
        <b>Subject:</b> Turn off the light<br>
        It's Friday. Don't forget to turn light off in B110.
        <br>
    </body>
</html>
```

An HTML tag describes how text is to be displayed in a browser

Simple example – email service using XML

An XML tag describes the meaning of the text

When saved as XML format:

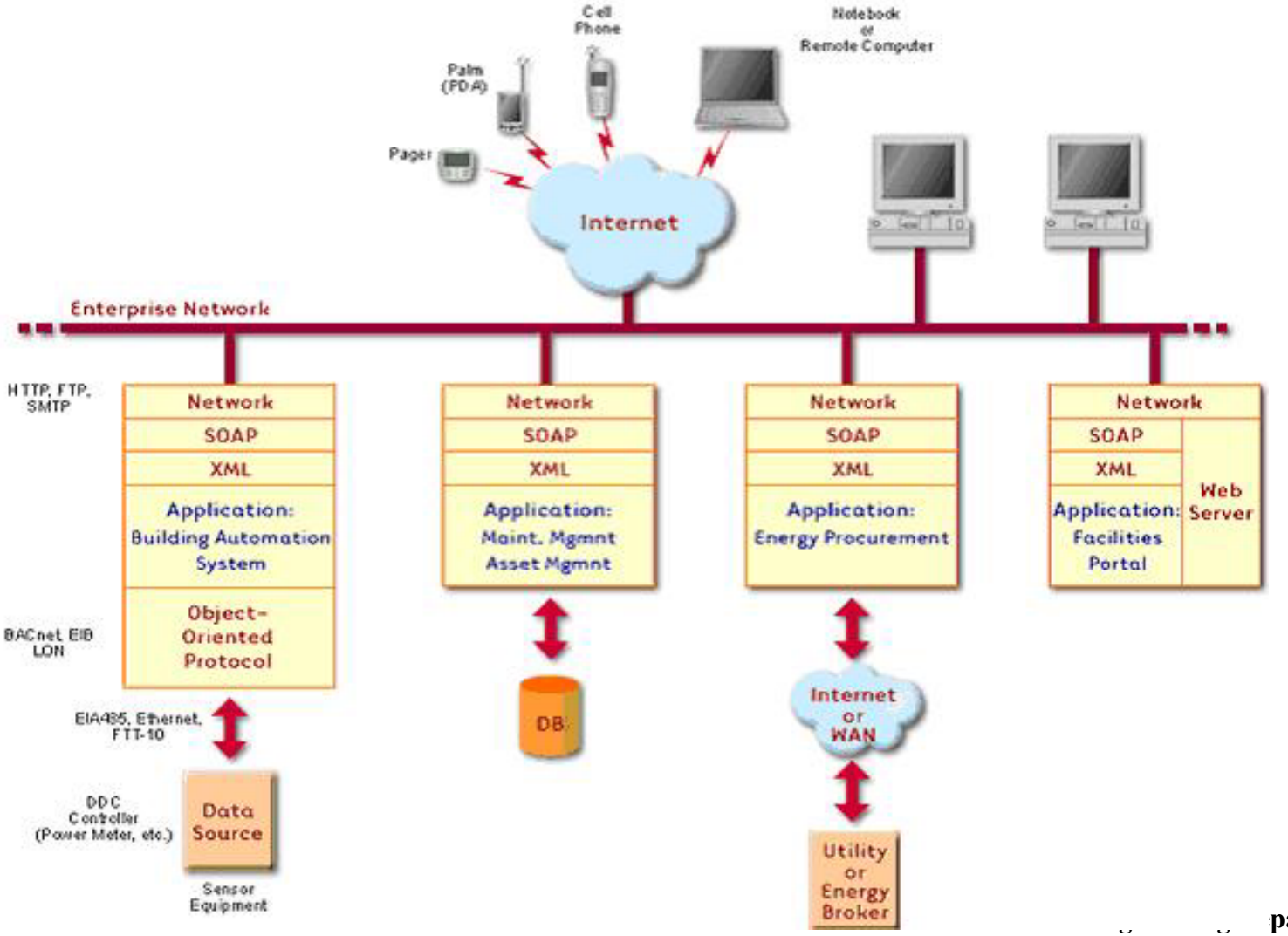
```
<email>
    <title>Email</title>
    <header>
        <to>ssptang@vtc.edu.hk</to>
        <from>tslam@vtc.edu.hk</from>
        <cc></cc>
        <subject>Turn off the light</subject>
    </header>
    <body>It's Friday. Don't forget to light off in B110.</body>
</email>
```

In XML "to" is not displayable text, but a *tag* that identifies something; in this case a destination address. The data is: ssptang@vtc.edu.hk. The metadata (or data about the data) is what to do with ssptang@vtc.edu.hk. In this case use it as the destination of an email. E.g. the "to" field.

System Models needed

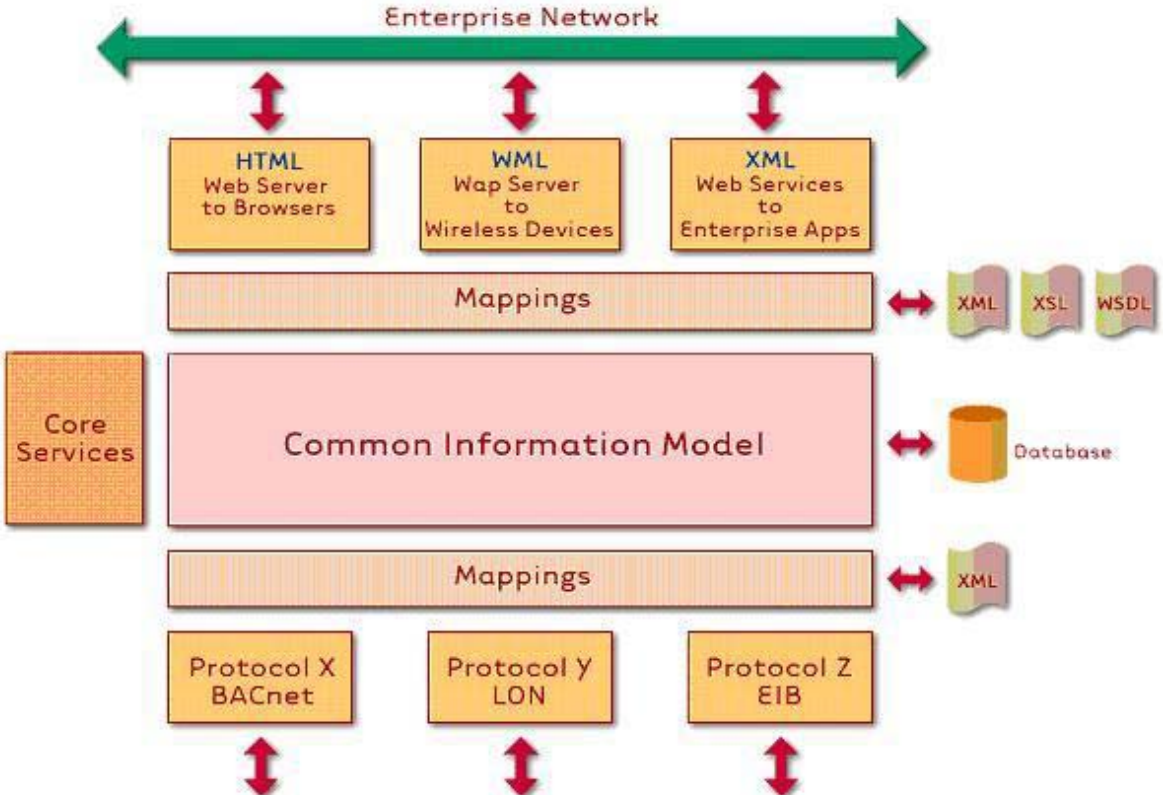
- A more complex file which taking energy data using iLON100 devices can be found in the lecture notes pg.12.
- All of the major Enterprise Software vendors are fielding products and platforms that support the Web services architecture using XML and SOAP, including
  - Microsoft.NET
  - IBM WebSphere
  - Sun Microsystems SunOne
  - Echelon Panoramix
  - Hewlett Packard Web services platform
  - Oracle 9i
- We could express the high level information models of BACnet, LonMARK, etc. in XML and in so doing make them compatible with the emerging Web services architecture.
- We also need a unified system model, in XML, that can be used by any building automation protocol for one example.

Web Services Architechture



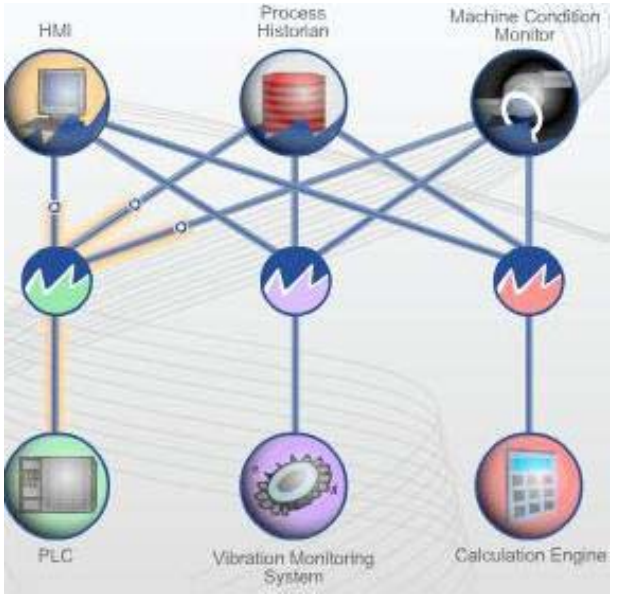
Integrating platform

- XML provides the HOW in high-level abstraction of what information is to be communicated, BACnet, EIB, and LonMark all provide the WHAT component of interoperability.
- Web services can provide an information platform that is high-level, cross-platform, cross-discipline, and multi-vendor as shown below.

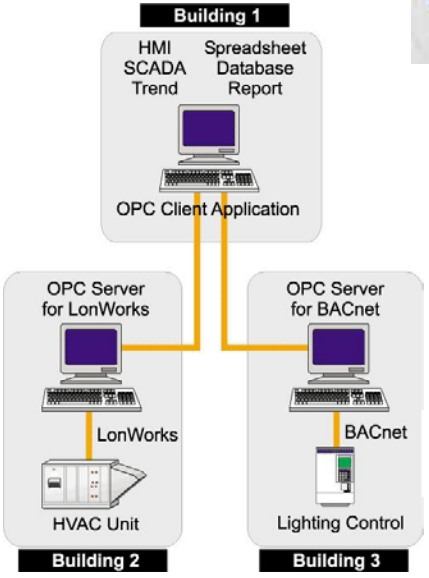


OLE For Process Control (OPC)

- OPC is to provide a standard-based infrastructure for the exchange of process control data from different systems.
  - This data can be available through different connections such as serial, Ethernet, or wireless.
  - Different operating systems like Windows, UNIX,, etc. can work with OPC as OPC clients or OPC servers based on COM and DCOM technology.



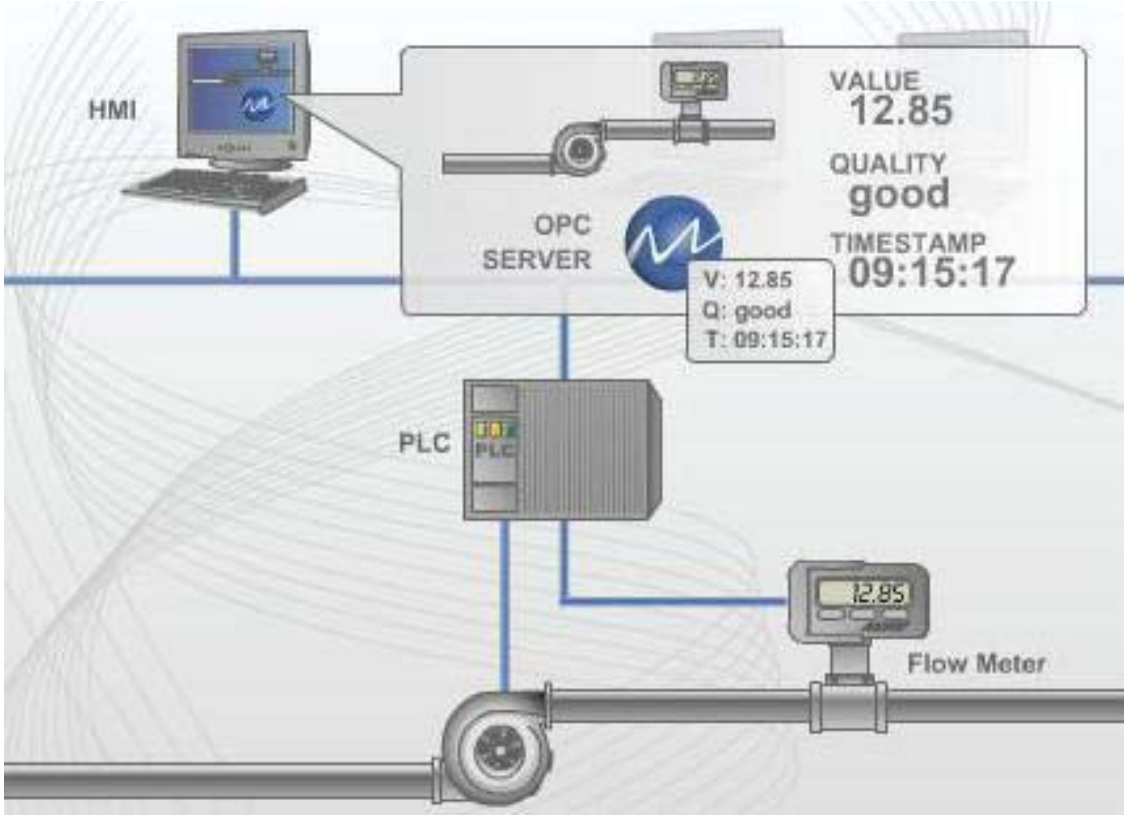
OPC drivers as the middleware



OPC Architecture

OPC Interface

- OPC defines a common set of interfaces
- OPC is as an out-of-the-box, plug and play communication solution if OPC drivers for each manufacturers' devices are provided.



OPC Interface example (*Value, Quality, Timestamp)*

OPC Present & Future

- OPC is not as flexible as XML that we have just mentioned
  - at least manufacturers have to provide its own set of driver to the other manufacturers' devices
  - OPC is more for process control, and is not originally targeted for broader application such as enterprise network
- Recently, OPC has come out with other specifications like OPC XML, OPC Security, OPC for Complex data, and OPC for ERP systems. Therefore, we may expect more functionality of OPC in the future.