Internet Control and Monitoring

Options and Complications

Loe Feijs and Mark Manders

In this position paper we address the topic of Internet control and monitoring of embedded systems. Rather than jumping into solutions, we wanted to establish a clear view of the potential advantages of networking embedded systems and the technical complications that arise. The paper summarises our main findings, the options and complications that have been identified.

Motivation

By an embedded system we mean a system that contains electronic subsystems, including one or more processors and their software playing an essential role in the system's functionality. Embedded software has the particular property that it is mostly invisible to its user, in contrast to for example office-software. Examples of embedded systems are a television set, a drive-by-wire system in an airplane, a shaver, a waferstepper, an ultrasound scanner for medical applications.

In the area of embedded systems it is not yet customary to take the Internet and Internet technology as a starting point. In this paper we investigate the viewpoint that this will happen sooner or later and that many technical advances will follow from that.

By Internet technology we refer to those developments that specifically enable the Internet, including its protocols such as TCP/IP, SLIP, FTP, HTTP, its naming schemes, the DVB-RC standards which will enable Internet access over cable networks, but also the software solutions for dealing with security and platform independence such as Java.

There are several reasons why many technical and scientific developments go faster when the Internet is involved:

- the round-trip delay of publishing ideas and results on the Internet, which are used, improved upon, and reported back, can be much smaller than with classical paper-based publishing. This is achieved by the combined effect of e-mail, FTP, newsgroups, but more than anything else by the world-wide web.
- many developments are carried out by several groups, which are in a kind of global competition; the best example of this is the competition for the user's attention in the area of search engines (Infoseek, Lycos, Altavista etc.), but there are many other examples. The competition model turns out to be very effective and stimulating.
- other developments are carried out in a kind of global cooperation model; the best examples are probably the GNU software tools and Linux. One of the key mechanisms is peer review and peer testing of implementations.

There are two assumptions which we make:

- the law of Moore will be continued,
- the Internet protocols will replace more and more proprietary protocols.

When developing an embedded system it is reasonable to anticipate on the achievements of Moore's law. This will reduce and eventually even solve the processor capacity and the communication bandwidth problems. But it will put special demands on the scalability of hardware and software solutions.

Moreover, many of the signal processing tasks that presently still demand dedicated electronics, will in future be done by general-purposes processors.[1] This means that the focus will have to be on the software aspects of the solutions. Most of the interoperability and compatibility problems of the next generation of open systems will concentrate in the "middleware layer". It should be noted that compatibility problems and user-interface problems are not solved by Moore's law (sometimes they are worsened).

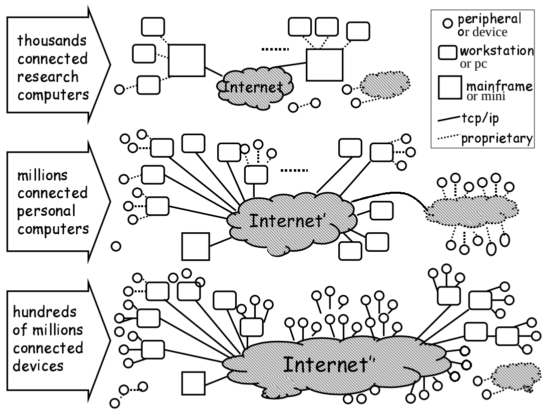
thousands connected research computers
millions connected personal computers
hundreds of millions connected devices
Internet
Internet'
Internet''
peripheral or device
workstation or pc
mainframe or mini
tcp/ip
proprietary

Figure 1: Penetration of the Internet into embedded systems.

The second assumption also means that the growth of the Internet penetration will be sustained, not only for connecting research computers, work stations and personal computers, but also devices, machines and computer peripherals. This will happen first in the domains of telephony and set- top boxes, but later more and more proprietary protocols will be replaced (or be put on-top-of) Internet protocol layers. The latter assumption is told in a pictorial way in Figure 1.

Our paper is structured as follows. We start off with problem statement that sets the goals for the rest of the paper. Then we present a list of options and a list of complications involved. Finally, we summarize our results and discuss related work.

Problem statement

The problem we address is that we want to have a clear understanding of whether it will be useful to connect various kinds of embedded systems to the Internet, and if so, whether there are still technical difficulties that could be addressed and perhaps solved by means of scientific reseach. Therefore we made a list of the options and a list of the technical difficulties that have to be addressed. We treat the following groups of options:

- telemonitoring and control,
- software download,
- intelligent behavior.

We treat the following groups of complications:

- security,
- feature interaction,
- reliability and real-time behavior.

Options of web-controllable applications

The main options of telemonitoring & control are on-off control for energy & emergency, adjustment of process-control parameters and finally status monitoring & data collection.

On-off control for energy and emergency Switching it on and off is the most elementary operation on any system, but in many situations it is advantageous to do it remotely, for example if the system is in a hard-to-reach place, or if the system is in a fixed place whereas the controlling user is mobile. Some care is needed because the embedded processor and its communication should not be switched off completely since in that case it will not even be possible any more to switch it on again (it will not be listening to the "on" command). For emergency switching it is also necessary to ensure the reliability of the communication channel, where it is important to note that the Internet protocols themselves are not particularly unreliable, but the typical

[1] or programmable Multimedia processors, as now already the Philips Tri-Media chip, Matshushita's 8.3 Bops MCP, Sharp's Data Driven Media Processor, or Mitsubishi's D30V processor

shared channels and servers of the public Internet do not provide any real-time guarantees.

Adjustment of process-control parameters Any embedded system is via its sensors and actuators connected with its environment such that their combination performs certain useful functions. Typical categories of functions concern a production process, or a telecommunication service. Although there are examples of embedded systems which perform a fixed function for a production process with fixed parameters, most production processes are very complex and are parameterized because of differences in customer's orders, fluctuations in the raw materials etc. Because of the real-time demands of the process control it is customary that the embedded system provides autonomous reactive behavior so that the control-loop is a locally closed loop. In that case the remote control actions should not address the actuators themselves directly, but only indirectly via changes in the control process executed by the embedded system. Such changes can take two forms: either a complete replacement of the control algorithm (software download), or an assignment to certain parameters, which makes sense if the algorithm itself is parameterised or is table-driven.

Status monitoring and data collection These are important in many applications. They stem from the fact that embedded systems are not isolated devices but that they are part of a system which delivers a certain service. There is a tendency for society to have more service-industries next to the classical product-oriented industries The following sub-options are important:

- billing
- statistics
- quality control
- production data
- fleet management
- remote diagnostics
- back-up and recovery
- planning of maintenance

Of these sub-options, *billing* is perhaps the most important one. A regular or continuous contact between the service provider and the device can transfer the data necessary for the service provider to perform billing, and hence to make business. On-line connections will replace old, laborous systems such as manual (billing) data collection. The need for this will further increase because of the penetration-effect of Figure 1.

Fleet management refers to the situation that a service-oriented company exploits a number of similar devices, which are hired by customers and wich are installed at the customer's premises. In that case it is advantageous to have a survey of the precise status of each device: which software components are or are not installed, whether the device is in use or not, whether it is in error-state, etc.

Remote diagnostics can save travelling times of maintenance engineers. In the typical old situation there is a service man who has to travel to see the device and come up with a diagnosis, and then go back to office to get the right spare parts and then return to perform the actual repair work. Each time the diagnosis can be done remotely, one trip is saved.

Back-up and recovery can lead to cost reduction and savings too: the idea is that back-up data can be stored on remote computers, at any reliable computer on the Internet or on the company's Intranet. This means that the embedded system itself can be smaller (perhaps even having no hard disc at all) and cheaper.

Planning of maintenance means that the device's statistics and production history are used to predict the right time for (preventive) maintenance. By combining these data with the production planning, the best timing for maintenance can be found, minimizing production losses and minimizing the probability of device failure because of too late maintenance. Actually this is already an example of "intelligent behavior" (a more detailed discussion of intelligent behavior is given below).

Software download

The main options are

- bug fixing
- customisation
- product family architecting

- software component marketing
- selling new features in software
- incremental delivery of software

Several general advantages apply to each option, viz. platform independence of programs (obtained by using Javascript, Java, and similar languages), and platform independence of GUIs (obtained by using HTML, XML, VRML and similar languages).

Bug fixing Of course it is much better to avoid bugs altogether, but in the practice of software development it turns out that when the software is large (in the sense of hundreds of thousands lines of code) there is always a certain number of residual bugs [6]. Beizer calls it the pesticide paradox: every method used to prevent or find bugs leaves a residue of subtler bugs against which those methods are ineffectual. Early removal of the a bug prevents the users from developing work-arounds or exploiting the bug as a feature (so that later it can not even be repaired at all). The typical traditional software update policy is to accumulate a number of bug fixes and to distribute them via tapes or discs together with the next functional upgrading of the system. The whole procedure of sending these physical carriers can be replaced by automated installation procedures via the Internet. The software provider can find out which software version is installed, to determine the proper upgrade software. Upgrading can be done at a suitable time, e.g. at night, when the embedded system is not used anyhow—by using a timed automated procedure or by an installation engineer working in another time zone.

Customisation For products in the *consumer electronics* domain the marketing trend is to allow the user to adapt the product to his or her personal taste and the user's personal way of working. Presently such customisation is done by fabricating the product in a variety of colors, having separate versions for different languages etc. Late customisation is advantageous over production-time customisation because it greatly simplifies production planning and stock-management. The solution is to perform as much customisation in software as possible. Typical examples are concerned with user-interface style, language, installation of frequently used software functions and de-installation of unused functions. For set-top boxes the cable exploitation company or the contents provider want to have new services, forms, menus and cryptographic algorithms downloaded. Another example concerns hand-held mobile communication devices that can be used for e-commerce applications (ordering things). The sales company wants to have a dedicated order-form to be installed in order to make products look as attractive as possible and make the purchasing process as easy as possible (the WAP wireless application protocol [7, 8] is a forerunner of the latter class of examples: WAP's WML is a light-weight and telephony-aware version of HTML but Internet compatibility is maintained by WAP gateways which take care of conversions).

In the *professional* domain there are important examples of machines which are installed in hard-to-reach places such as clean rooms and spacecrafts. New scheduling algorithms, control algorithms, data compression algorithms or data filtering algorithms should be downloaded on an as-need basis to fit changes in the system's tasks and its environment.

Product family architecting This is best explained in the description of the ESAPS project: to have a large commercial diversity with a relatively small technical diversity [4]. Actually this is more like an ambitious R&D goal rather than a single technique but the inflexibility of large monolithic software systems led to the quest for modular domain assets which can be configured to fit the market's demands. It is our claim that for many of these assets it will be easier to make them by Internet technology. Moreover they are inherently portable.

Software component marketing Traditionally software was sold as complete packages. But recent developments show at least two different business models: the software component market and the open source movement. The *software component market* is best illustrated by the VBXs and later OCXs that can be used in development environments such as Visual Basic. Technically the creation and usage of components has become possible by a component framework, viz. ActiveX, based on

COM and DCOM. The component framework defines the applicability of the component: a given OCX will fit into Visual Basic, but not in Haskell or in a Unix shell; moreover as a binary it will run on any member of the Intel Pentium family of processors, but not on any other processor. There are OCXs for almost any software function (for data mining, for data compression, for using the ping protocol, for using the HTTP protocol). In 1997 Gartner Group estimated the market size for components to be $500 million in 1996 and $7 billion in 2001 (software $2 billion and services $5 billion), or 'only' $267 million and $1.12 billion, respectively, as estimated by IDC (reported in [9]). This is already a big market and therefore we expect that a larger market will arise for programs that have an even wider applicability such as applets and beans because they are more portable. The *open source* movement offers software components which are attractive because they are for free, have been tested by a large community, and which are transparent, hence more safe, because the source code is known. The business model is different, however: payment is to be given for added services and not for the component's code itself. References are GNU en Linux.

Intelligent behavior

One of the key features of the Internet is global availability of information. With hundreds of millions connected devices, as depicted in Fig. 1, the amount of available data is enormous. Using this information efficiently to increase the performance of embedded systems will be one of the most interesting challenges in the near future. The two options considered are using data resources on the web and connecting to other embedded systems.

Using data resources on the web More and more data resources are becoming available on the Internet. Much can be gained by using these to improve the efficiency of an embedded system or to provide new features, as will be clarified by the following sub-options:

- customer behavior analysis
- competitor behavior analysis
- environmental data
- raw material costs
- traffic predictions
- weather predictions

The first sub-option treated here is *customer behavior analysis.* This refers to the situation that the embedded system uses various customer behavior data resources, which enables it to, for example, anticipate the users requests to come, probably resulting in better quality of service. Not only the user could benefit from this sub-option. Connected embedded systems could also collect all kinds of information about its users behavior and add this to the data resource, thus improving it. Of course privacy will be an issue here.

An example of Embedded systems in the telecom field could benefit from *competitor behavior analysis data*, for instance to avoid heavy traffic connections.

Using *environmental data resources* is potentially the sub-option allowing the embedded system to provide additional features. This means features, which were not originally implemented but became available once the Internet-connection was established.

In case of an embedded system, able to use different raw materials, *raw material costs data resources* on the Internet could be used to minimize the costs of operation or production. An example of this is a car, driving on both LPG and petrol.

Information from data resources containing *traffic predictions* could be very useful for embedded systems depending on this kind of information. Directly benefiting from this kind of information could be for instance navigation systems. A nice example of such a system is Carin, a car-navigation system developed by Philips. Carin uses the cars current position, measured by 3 satellites, current speed and orientation, using the speedometer and a gyroscope, compares this with a digitized map and gives you directions towards your destination. Traffic related data resources on the Internet could be used to avoid traffic jams and roads under construction. More information about Carin is available at http://www.carin.com. Other examples of this category are traffic lights and systems providing dynamic road assignment and recommended speed indication.

Indirectly benefiting could be systems depending on deliveries. These data resources could help them predict the arrival time of these deliveries.

Weather predictions are especially useful to embedded systems in weather depended fields like horticulture, greenhousing and many kinds of industry. For instance in the field of petrochemical industry distilling columns are used to decompose crude oil into thousands of useful fractions. This process is very costly, $ 100M/year, heavily dependent on temperature and humidity. Therefore if weather predictions could help improving the efficiency of this process by only a few percents this would mean millions of dollars cost-reduction per year.

Connecting to other embedded systems Obviously, all embedded systems connected to the Internet, are somehow connected to each other, thus enabling the following sub-options:

- warehousing
- just-in-time delivery
- cashing and sharing of large datasets

Warehousing: refers to the situation that several embedded systems in a production and/or storage environment are interconnected and exchanging relevant information and using this information to increase the overall efficiency of the facility. The main system of such a facility is an embedded system keeping track of supplies in all parts of the production process. An example of this could be the automation of order picking in assembly plants.

Just-in-time delivery is closely related to warehousing. Production machines or robots could use information from a warehousing system to adapt their throughput to minimize buffer capacities. Another example of this in the logistics area would be Albert Heijn, a large chain of supermarkets in the Netherlands. They are using supply information, which enables them to shift from using warehouses to using trucks, and so optimizing the distribution of their products.

cashing and sharing of large data sets. Several production lines can make use of one large dataset. The obvious advantage of this is consistency of all relevant data in use in the production facility. This dataset does not necessarily have to be located inside the facility. Global availability is the expression to remember, allowing production facilities or embedded systems to be geographically far apart but still able to work together. Also backup facilities are much easier implemented using a single data set, with Internet-access.

Complications

Whereas the previous section shows the potential benefits, the present section is about the technical difficulties that certainly will arise and that will have to be addressed and perhaps solved by means of scientific research. We treat the following groups of difficulties: security, conceptual integrity and feature interaction, real-time constraints and resource constraints.

Security

We consider several aspects, as indicated by the bullets below. For a survey of the general problems and solutions for Internet security we refer to [5].

- Data confidentiality
- Access control and authentication
- User-friendliness of key handling
- Keeping pace with hackers and crackers
- Keeping pace with trends and standardisation SSL, DES, RSA, PGP, AES, firewalls, etc.
- Respecting national and international regulations

Data confidentiality actually we should refine this: there are attacks of several different kinds: denial of service, eavesdropping, false information being injected into the system, stealing of passwords, credit card numbers etc., false authentications.

The ways of undertaking an attack include guessing passwords, forging packets, replaying packets and installing trojan horses. Usually a successful attack is organised as a sequence of these ways, succesively exploiting a series of weaknesses in the system, each of which on its own seems innocent. For example (step 1) guessing a password and thus becoming a normal user having read-access to lots of system files, (step 2) reading the sources of the CGI scripts of the web server and figuring out the weak spots in there (step 3) attacking the web server and gaining execute access over modified CGI scripts, etc.

Many of the current issues in Internet security are related to the protection of routers. If adversaries can manipulate the routing tables then many options for malversation are available (see [5] Section 4). Progress goes along the way of integrating cryptographic techniques into the protocols. For example the PSF protocol which is used to calculate routing paths had no security, whereas the second version OSPFv2 supports passwords and other cryptographic techniques. For other standards such as I-ISIS (intermediate system to intermediate system) routing protocol, an enhanced version has been proposed, but then the cryptographically protected version is not in widespread use (yet). These issues will have to be dealt with mostly by backbone operators and router vendors rather than by the end-users and the terminal embedded systems providers.

Access control and authentication and user-friendliness of key handling These aspects are closely related. As [5] put it: “in most security systems, the single most important component is key management. If the adversary knows the key(s) in use, then the innate strength of the cryptographic algorithm protecting the data just does not matter”. Manual key distribution is problematic because it leads to clumsy procedures or to approaches where users embed the keys in normal e-mails which are vulnerable to eavesdropping. There are standards such as IKE (Internet key exchange) protocol. Keys may have been attributed with additional data such as expiration dates. If keys are to be distributed by a certification authority (CA) then the communication with CA needs to be secure as well so there may be chains of CAs to deal with these nested key distribution problems. The embedded systems developers will have to follow these technologies but their application is certainly complicated by the fact that the embedded system itself often is not accessible, and unlike a normal PC it has no keyboard for typing passwords and keys. So most of the authorisation procedures have to be handled by the remote user and auxiliary servers. This is why the SNMP protocol is interesting, because this is meant for this type of situation. But SNMPv1 uses clear-text reusable passwords and the improved version SNMPv2 is not in widespread use.

Keeping pace with hackers and crackers the field of security can be seen as a continuous fight between system providers and system managers on one hand and the hacker community on the other hand. As experience learns, this is not a static battlefield: as new protection mechanisms are developed, hackers find new weaknesses. For example, the problems of dial-in modems which dominated security several years ago, have been replaced by the problems of attacking web-servers. New problems have emerged, such as the circumvention of the classical mechanisms that protected the intellectual properties of music companies and movie companies. MP3 and software DVD players have shown that protections agreed among a closed consortium of equipment providers are not sufficient any more when the monopolies of player construction are bypassed by software solutions.

Keeping pace with trends and standardisation A very attractive approach to improve security is to have a security-related layer close to the application software. For example SSL (secure socket layer) is an emerging standard developed by Netscape to protect application-level data such as credit card numbers. The nice thing about SSL is that it does not put special demands on the routers or the router protocols; it only requires a reliable transport layer, as TCP is (changing the routers and router protocols is not in the realm of the embedded system developer). SSL hides the key generation and the negotiating techniques. The services offered include: (1) authentication of the client and the server to each other, (2) encryption of the traffic over the communication channel, (3) data integrity at the socket record layer and at the message layer, (4) record freshness (to avoid replay attacks).

Respecting national and international regulations this is very much dependent on the application domain. For example in the medical domain, all embedded software must satisfy high standards with respect to the software quality and the quality of the software development process. Usually this can be guaranteed by the embedded systems provider, who therefore has to exercise a sufficient level of control over its subcontractors. But what happens when software download is technically possible? This is not only a technical issue, but also a legal issue.

Conceptual integrity and feature interaction

distribution whereas in a classical embedded system all state information about the system and its user is embedded inside the system itself, this state information will now become distributed over different computers which are at distinct locations. We have to consider at least the embedded system itself and one or more remote users, but in the general case there are also service providers such as a central database of fleet-management data and such as a back-up facility. The problems of distributed systems are well-known: because of the unpredictable transfer-time for data transfer from one computer to another, there is no concept of "the state of the system", and there is not even a notion of global time. If one piece of information is duplicated the problem is that it will have one value in one computer and at the same time another value at another computer. Avoiding this is what we mean by conceptual integrity. The problem can be addressed by the solutions known from the area of databases: locking and transaction management. If no information is duplicated at all, the problem of conceptual integrity disappears but information must be retrieved from remote sites which leads to unacceptable delays.

several access levels for users security and safety demand that access to the system be restricted to authorised users only, but there is not just one class of authorised users but several classes. We distinguish the *normal users* and the *operations-and-maintenance users*. For complex and professional systems each of these classes has several subclasses. For example in a PBX (private branch exchange) a normal user corresponds to a single telephone extension; there are subclasses such as extensions which are allowed to make company-internal calls only, extensions which are allowed to make local calls and similarly national calls and international calls.

It is expected that many of the operations and maintenance tasks can be solved by using SNMP (simple network management protocol). This is one of the Internet protocols. It is intended for managing the network aspects, but it can be used for other aspects of a system as well. The SNMP concepts include *managed devices* such as computers, routers, etc., *agents* which collect management information such as number of packets received, and *managed objects*, usually described by tables. Version 2 of SNMP includes an authentication protocol to check a message's integrity, making sure that the sender's identity is correct and a DES-based encryption protocol to ensure a message's privacy. Specific network component providers have added additional mechanisms, for example Cisco uses access lists to prevent SNMP messages from traversing certain router interfaces.

Although SNMP and its concepts are reusable, the definition of access levels will contribute significantly to the overall complexity of the embedded system's design and its user-interface.

many new possibilities and options if more and more services, user controls, remote controls, and intelligent behaviors are added to a given system it is unavoidable that a problem appears which is well-known in the world of telephony where it is named 'feature interaction'.

multiple simultaneous users this refers to two related problems. The first problem is that the physical hardware and the the local data structures of the embedded system's control need *protection against simultaneous access* and against inconsistent use by multiple users. The situation of multiple users can arise because there is one local user (for example using buttons on the machine itself) and one or more remote users, or because there are two or more remote users. Please note that it is not enough to serialise individual requests: for most applications there is a concept of "session" such that each session is devoted to one user or a given set of users. While the session is ongoing, certain commands are restricted to the session participants only, while other commands such as status-information requests are still available for all users. The protocols may become quite non-trivial if timeouts are allowed as a way of ending sessions or changin a session's state.

The second problem is that a change in the *system's state must be reflected* on the remote user's display, not only for the user who has initiated the change (which can be done as a straightforward application of the HTTP protocol), but also for the other users that are monitoring the system. One of the problems is that the HTTP protocol is memoryless, so once the HTTP transaction (which is of the request-respond type) is over, the server does not have that client's data any more. Moreover users may have left the monitoring page by surfing to another webpage or by logging off without notifying the embedded server. The same problem appears already for a single user when the data change is a spontaneous change, that is, a change induced by the physical system or by a time-out in the server's software. Standard HTTP is not intended for this. The so-called Server Push technology can solve this by keeping the connection open and by inserting a special trailing boundary to indicate that a page update is arriving. But at present some important browsers do not support Server Push. Another solution could be to use special packets such as chat box packets to inform the other users of the fact that a change has taken place. But this may give additional problems such as the fact that most often firewalls may be configured to block the chat box messages while the HTTP requests would go through. Yet another easy 'solution' is to build automatic refresh timers into the web pages. Essentially, this amounts to polling the embedded server, generating unnecessary traffic and possibly still leaving the remote users with a certain latency (depending on the timer's value). In summary, better solutions are needed.

Real-time constraints

We discuss two issues, viz. speed/predictability of high-level languages and robustness and locality of control loops.

speed and predictability of high-level languages
The advantages of high-level languages are well-known: higher programmer productivity and instruction-set independence (modulo recompilation). Moreover certain languages have a built-in garbage collector, which is a good solution for two well-known problems:

1. avoiding run-time errors because of attempting to dereference a *nil* pointer when traversing a dynamic data structure;
2. avoiding a steady increase of allocated memory because unused parts of the data structure are not freed. This is called *memory-leakage*.

It is easy to solve just one of the problems: for instance, never performing a 'free' reduces the risk of run-time errors but it may create the memory-leakage problem. Careful programming generally leads to good solutions, but sometimes at the expense of explicit bookkeeping of the data. The Java approach (following a tradition of LISP and other functional languages) takes a certain burden away the programmer. It works with a run-time program called an *automatic garbage collector* which finds out which data are unused and then collects these and defragments the memory. The general disadvantages of high-level languages are:

1. compilers tend to generate larger and slower programs than assembly-level or certain C-level programs. In general the effect is not important any more, although in specific cases a directed manual effort still could outperform a compiler;
2. the execution of a high-level program is harder to predict with respect to memory usage and execution time. The main problem is with data memory (not program memory). To prevent unbounded memory usage some care is needed when using recursion. The garbage collector is unpredictable because at some point in time the system happens to run out of memory and then a garbage collection of several seconds, perhaps,

begins during which the program is suspended.

It is also interesting to note the effect of using an interpreter instead of a full compiler (or a mixed approach with preprocessing by the compiler which produces so-called byte code). The byte code tends to be more compact than machine code but it runs slower. An advantage is that the byte code still is instruction-set independent. The interpreter of Java is called Java Virtual Machine. For Java a range of variants has been developed (see [10]): Java, Personal Java, Embedded Java and JavaCard, whose applications range from Enterprise servers via traditional embedded systems to even smart cards. One of the techniques used with Embedded Java is to analyse the application and to strip the standard libraries, only keeping the code for the functions really needed.

robustness and locality of control loops In general the environment of an embedded system expects timely responses. As was always the case, the operating system or real-time kernel must be able to serve interrupts timely and the context switches between parallel tasks must be fast enough with respect to the maximal allowed response latency. A wide variety of scheduling and analysis techniques is available for allocating the processor-time, such as the popular Rate Monotonic Analysis. But now the Internet connection must be taken into account too. It must get a priority such that FTP, HTTP requests etc. are dealt with properly, but never at the expense of the responsiveness towards the local environment. The control program best runs locally and not in the remote client since there are no guarantees about the response time of the Internet connection. A hybrid solution could be to have certain parts of the control program running on the client, but to continuously measure the round-trip delay (just as the PING protocol does) and switch to a safe situation when the round-trip delay exceeds a certain threshold.

Resource constraints

Unlike the desktop PC one of the major issues regarding embedded systems is resource constraints. Obviously making an embedded system web-controllable will demand extra resources. Two complications will be treated, viz. size and costs of processor(s) and memory, and the problem of the last meter (communication bandwidth).

Size and costs of processor(s) and memory Formerly the core of embedded systems usually consisted of dedicated hardware, therefore adding Internet capabilities asked for adding extra hardware and/or software thus bringing on problems with interfacing and higher energy consumption. The latter is a very important complication considering portable embedded systems. Nowadays there is a trend towards using multi-purpose hardware, thus using software to provide the desired behavior. Usually interfacing with such a system is relatively easy and minimal additional hardware is needed, but usually there is a higher demand on processing power and memory [11]. We believe it to be reasonable to anticipate on the achievements of Moore's law to solve the latter. On the other hand the Internet connection could help solving these problems, for example by storing large data files or doing remote processing on some also connected computer, with more processing power and storage capacity. Of course real-time behavior will be much more difficult to guarantee then. Especially for backup-files and log-files, which typically tend to be very large, this remote storage option could be very useful. Buffering these data and only sending them during times of a surplus of processing power allows a trade-of between buffer capacity and real-time delay.

The problem of the last meter (communication bandwidth) This problem refers to the problem of how to realize the physical connection between the embedded system and the Internet. We make the following observation. If conventional devices are to be attached to the Internet, additional cables are not acceptable. Therefore, the Internet connection has to be established by already existing cables (read power supply cable) or by wireless networks.

What has been achieved

In general there is little doubt that the evolution from traditional embedded systems to open systems whose architecture is completely web-oriented has begun. This view is not only supported by the extensive list of options and potential benefits, but also by the appearance of various demonstrators, and products (usually still with very restricted web-access).

In this article, we have provided an enormous list of potential benefits, many of which lead to savings in energy, savings in travelling costs, and reduction of production loss. Others can be summarised as improvements of the quality of the service in which the device plays a role. The added ways in wich the users and the service provider can communicate with the device are consistent with the tendency to have more service-industries next to the classical product-oriented industries. In other words: improved feedback will lead to an improved service level as well to software improvements for the next software version (by rapid application development, or a spiral software development model). Moreover software download makes it possible to use new results coming available in the application domain; they can be turned into new software which can be downloaded to upgrade an existing system. New insights in the application domain can be used to upgrade an existing embedded system).

When the design is approached well, the development of embedded systems can become cheaper. The developers can benefit from the ongoing developments in Internet technology. In addition, more benefits will follow according to the fact that technical and scientific developments go faster when the Internet is involved, as explained in Section *Motivation*.

But the potential advantages cannot be fully realised unless certain key roadblocks have been removed. This has been discussed in a systematic way in Section *Complications*. We found that we need better solutions for the following issues: security, feature interaction and conceptual integrity, real-time constraints, resource constraints.

As a concluding remark, we think that the Internet will have a large impact on the the way development and evolution of software architectures for embedded bsystems has to be done. If we exaggerate then we could state that the design of an embedded system can be approached as the design of a web-site instead of a a microprocessor based control loop.

References

[1] H. Aalderink. Web controllable devices, Concept and design, Master's thesis TUE ICS/EB, coaches: A. Lommen (TNO Institute of Industrial Technology), A. Verschueren (1999).

[2] M. Kucera, I. Smaili, E. Fuchs, A lightweight Ethernet protocol to connect a time triggered real-time system to an Internet server, report TU Wien, Treitlstrasse 3/182/1, A-1040 Vienna, Austria (1998).

[3] F.J. Van der Linden (Ed.), Development and Evolution of Software Architectures for Product Families, Springer LNCS 1429).

[4] F.J. Van der Linden, J.H. Obbink. ESAPS – Engineering Software Architectures, Processes and Platforms for System Families (to appear).

[5] R.J. Atkinson, J.E. Klinker. Progress in Internet Security, in: Advances in computers, vol.48, Academic Press (1999).

[6] B. Beizer. Black-Box Testing, Techniques for Functional Testing of Software and Systems, John Wiley & Sons (1995).

[7] WAP forum. The WP white paper. http://www.wapforum.org/what/WAP_white_pages.pdf

[8] WAP forum. The WP architecture. http://www.wapforum.org/what/technical/ SPEC-WAPArch-19980430.pdf

[9] J. Sutherland. Jeff Sutherland's object technology web site (Object Magazine morphs into Component Strategies), http://jeffsutherland.org/news0798.html

[10] J. Janssen, J.-D. Van Doorn. Java veelbelovend voor communicatie tussen ingebedde bsystemen, Elektronica, nr 7/8, pp. 29–31, Kluwer (1999).

[11] L.M.G. Feijs. Ingebedde systemen en hun evolutie, Elektronica, nr. 3, p. 44-47, Kluwer (1999).