A Real-Time Ethernet Network at Home

F. Hanssen, P. Hartel, T. Hattink, P. Jansen, J. Scholten, J. Wijnberg
Distributed and Embedded Systems group
Faculty of Computer Science — University of Twente
E-mail: hanssen@cs.utwente.nl

Abstract

This paper shows the current state of our research into a home network which provides both real-time and non-real-time capabilities for one coherent, distributed architecture.

It is based on a new type of real-time token protocol that uses scheduling to achieve optimal token-routing in the network. Depending on the scheduling algorithm, bandwidth utilisations of 100% are possible. Token management, to prevent token-loss or multiple tokens, is essential to support a dynamic, plug-and-play configuration.

Our network will support inexpensive, small appliances as well as more expensive, large appliances. Small appliances, like sensors, would contain low-cost, embedded processors with limited computing power, which can handle lightweight network protocols. All other operations can be delegated to other appliances that have sufficient resources. This provides a basis for transparency, as it separates controlling and controlled object.

1. Introduction

At first there were only a few main-frames shared by many people. Then came the era of personal computing. People do not have to share a computer with other people. Life became much easier, but the relationship between human and computer is sometimes uneasy. One distinguishing characteristic is that we are very much aware that the computer is there and that we use it.

Slowly other types of computers entered our homes and replaced their electronic or mechanical equivalents: embedded systems in TV-sets, the central heating system, etc. This led to the idea to connect these appliances to make ‘something new’. This is called ubiquitous computing. Its highest ideal is to make computers “so embedded, so fitting, so natural, that we use them without even thinking about it”[11]. So, in theory we should be able to control or get services from the house, from any device, at home or via Internet or mobile phones.

Presently, a house has several separate distribution and communication infrastructures: telephone, cable TV and radio, PC network, central heating control, etc. In most cases these infrastructures are isolated islands that interconnect only on rare occasions. These infrastructures can generally be divided in three classes:

- *entertainment*: audio, video, games, etc. This class requires high bandwidth and real-time responses. Characteristic for this class is isochronous, streaming data.
- *control*: sensors and actuators, e.g. central heating control, fire detection, burglar alarm, etc. Control uses low bandwidth, but requires a high degree of dependability. Some devices may need real-time services.
- *information*: PC applications, Internet, etc. This class uses bursty traffic, and only needs best-effort responses.

The first step to connecting appliances is one common, inexpensive infrastructure that supports entertainment, control and information. This infrastructure may incorporate different types of wired and wireless networks.

The first objective of our network is to be a network for entertainment, control and information that supports both real-time and non-real-time data, as for streaming media. This network will be based on a new variety of a rotating token protocol, giving bandwidth to the appliance that has the token. In existing timed-token networks, every node in the network is visited once during one rotation of the token. In the worst case, timed-token networks have a low utilisation: ca. 33%. Main properties are described by Malcom and Zhao [6] and Sevcik and Johnson [10]. Examples are the IEEE 802.4 token bus, the IEEE 802.5 token ring and FDDI.

We propose a new type of real-time token protocol, that can be used on top of low-cost network hardware, e.g. IEEE 1394 (Firewire), Bluetooth or ethernet. Unlike other protocols, the token does not follow a fixed path, visiting every node during each rotation. Instead, the token is scheduled and follows a dynamic route, visiting only those nodes needing attention. The scheduler is decentralised and resides in every node. When needed, a node may reschedule the token and give it a new route through the network. This happens when new connections are added, or existing connections are changed in the network. Before applying the new schedule, the node will check if this schedule is feasible. Token management needs special attention to make this type of network robust. Special measures have to be taken if the token is lost (e.g. the device that holds the token is removed), or if multiple tokens exist.

Next we will give a short description of the project in which our research takes place, describe the token protocol we devised and discuss the simulation and prototype we have built, and the results that have come out of it.

2. At Home Anywhere

Within the context of building a real-time network for the home environment the following problems/challenges exist:

1) *Compatibility and interoperability*: connecting appliances does not necessarily mean that they can work together. Appliances use different data types, physical networks, network protocols, etc.
2) *Network incompatibility*: even if appliances could use the same network they would interfere with each other's proper operation. For instance, multimedia streams can monopolise the network because of their large volume and prohibit data transmission from crucial appliances, like a fire alarm.
3) *Data redundancy and incompatibility*: certain appliances have redundant storage with different formats, e.g. minidisc (MD) recorder and video recorder. As a consequence, MD music is only available through some MD player, and the same applies for video. It is difficult to distribute audio and video if there is no common data format among the appliances.
4) *Resources*: when even the smallest of appliances need to be connected, costs for the embedded computer system become increasingly important. To reduce costs, these systems must be lean and be as small as possible.
5) *Ad hoc configuration*: the configuration of the system must be dynamic. That is, a user of the system should be able to add appliances to the network or remove them without resetting the rest of the system. Setup and configuration must be done automatically.
6) *World wide interconnection*: at some point the house will be connected to the outside world, where different networks and protocols are used, and different levels of security are needed.

We will resolve, or give directions on how to resolve these problems/challenges within the *At Home Anywhere* (@HA) research project. The @HA architecture constitutes two types of rings (figure 1). The inner ring is the system at home (@Home). The second, outer ring is the Internet. As there are many houses, the Internet contains many @Homes, but from one house's viewpoint there is one @Home and all the rest belongs to the Internet. A gateway, or proxy, connects both worlds.

3. Resources

We focus mainly on item 4 (resources) and a little bit on item 5 (ad hoc configuration) mentioned in section 2. We use a token-based network protocol to handle these issues.

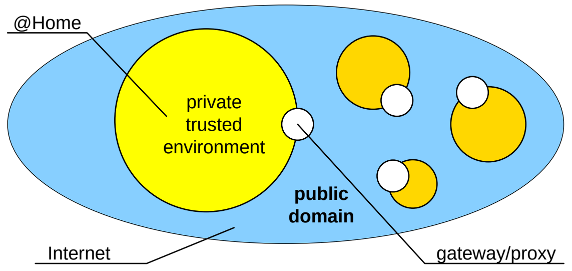

Figure 1. @HA architecture

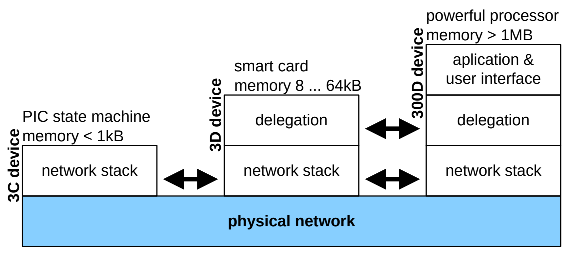

Figure 2. @HA classification of devices

Even resource-lean devices, like simple sensors, should be connected to the network, but their size and price preclude 'heavy' processors to realise that. We will study the concept of delegation, or controlled invocation: small systems use their limited processing power to implement a network stack to connect to the network and a small real-time operating system kernel to handle the lightweight protocols. But even this may be too resource-rich. Sometimes a simple, network protocol handling state machine is enough.

Depending on the type of appliance, a balance must be found between hardware and software. Hardware can range from a simple programmable IC to a microprocessor, while software can range from a simple state machine to a complex programme. In our case we distinguish three classes of appliances (figure 2):

- *3C (3+ cent) appliance*: simple devices that implement only a network stack to connect to the system. A temperature sensor is an example of such a device.
- *3D (3+ dollar) appliance*: medium complex devices that implement network stack and delegation protocols. Examples are printers and scanners.
- *300D (300+ dollar) appliance*: powerful devices, controlled by a complex embedded computer. Examples are TV-sets and PDAs.

Delegation is the basis for the first type of location transparency (home location transparency): the user interface (UI) and underlying application of any appliance are available anywhere in the house. This concept is not new. X [13] separates application and UI, so each can execute on a different network node. But this system has a high information exchange between client and server and is not suitable.

Our network at home constitutes a closed environment. Therefore, delegators can trust delegatees and vice versa, and devices understand a common base vocabulary. Then it should be possible to base a delegation protocol on the exchange of keyword-value pairs, with typing constraints to ensure consistency and robustness. This is essentially the same approach as taken in the CC/PP proposals [1], [7].

4. The token protocol

In order to schedule real-time traffic on a network, we use scheduling algorithms, which are originally meant for scheduling tasks on a processor. We view the network as the processor and the different streams flowing through the network as tasks. To make sure only one stream is being sent at the same time, a token is used. This token contains all information needed to make scheduling decisions.

When a token arrives at a node, the node looks in the token to determine which stream is up for transmitting its data. It also determines the length of time the stream may transmit, this is called the *token holding time* (THT). When this time has elapsed, the scheduler looks in the token to determine which node holds the next stream that is to be scheduled, and then sends the token there. So we route the token through the network based on scheduling decisions, and we call this *token-routing*.

We do not use a fixed schedule to route the token from node to node through the network, the token is following a dynamic schedule. We use pre-emptive Earliest Deadline First (EDF) [5] as a scheduling algorithm for the token. But our protocol does not preclude the use of e.g. Deadline Monotonic [3] or Rate Monotonic scheduling algorithms, or their non-pre-emptive variants. When no real-time traffic is being transmitted, the token uses a simple round-robin strategy to travel across the network, visiting every node.

To prevent token loss, a monitoring node is introduced: the node that held the token last. When a node sends the token to the next node, it becomes the monitor. The previous monitor stops monitoring. When the node holding the token dies, the monitor will notice and send the token to the node the deceased node should have sent the token to.

Adding new streams to the network can only be done by the node holding the token, so we should make sure every node gets the token regularly. The original design, as used in the simulation, let the system take care of that during the non-real-time traffic round. But tests using the prototype showed that is is better to define a real-time stream for adding new streams, so it is guaranteed that every node gets an equal share of the time in order to add new streams.

The validity of the bandwidth reservation of existing streams is guaranteed by letting nodes, that wish to add a stream, perform a feasibility analysis first. This feasibility analysis determines whether the network will not be overloaded, so bandwidth guarantees can be maintained.

Adding new nodes to the network is done by regularly broadcasting messages to the network, indicating nodes wishing to join should reply to this broadcast. When they do, they will be added to the list of nodes in the token, making sure they will get the token after some time, so they too can add real-time streams.

In order to make sure non-real-time traffic gets a piece of the total network bandwidth available, we allocate only a portion of the total bandwidth as usable for real-time traffic. Currently we chose this portion to be 80%. This number was chosen randomly, we have not yet done any research on this, it may have to be chosen lower or higher, depending on the network situation and traffic offered or requested.

5. Simulation and prototype results

Although no final choices have been made yet, for demonstration of the prototype the two following areas of application are chosen:

integrated home system

- central storage server;
- sensors controlled from anything suitable at home;
- audio and video anywhere at home;
- PC applications.

health care ubiquitous computing

- real-time wireless monitoring and alarms;
- real-time remote assessment in case of emergencies;
- secure access to files and information;
- consultation by video and audio.

5.1. The simulation

To see if our ideas were feasible and to get an idea of the overhead our protocol would induce we have created a simulation of our real-time network token protocol on top of ethernet [2]. This simulation currently supports real-time and non-real-time data traffic, where the real-time data traffic is scheduled using pre-emptive EDF. Various recovery and network management techniques are implemented to handle network or node failures. Also ad hoc configuration of newly plugged-in network devices is supported.

In figure 3 we show some results we obtained by simulation [2]. The network simulates three real-time streams, with periods of 0.01, 0.02, 0.05, 0.1, and 1 seconds. The offered stream loads which are used, are 0.1, 0.2, 0.5, 0.7, and 0.8. Each simulation ran for five minutes, where the actual measurements were taken between the 10$^{\text{th}}$ and 290$^{\text{th}}$ second, and then averaged. The maximum real-time stream utilisation was limited to 80% of the total network bandwidth, so that 20% of the network bandwidth stays available for non-real-time traffic. The graph shows the actual utilisation measured, including protocol overhead.

Some utilisations are zero. The combination of period and offered load meant that all streams were rejected, because the computation time was below the minimal THT.

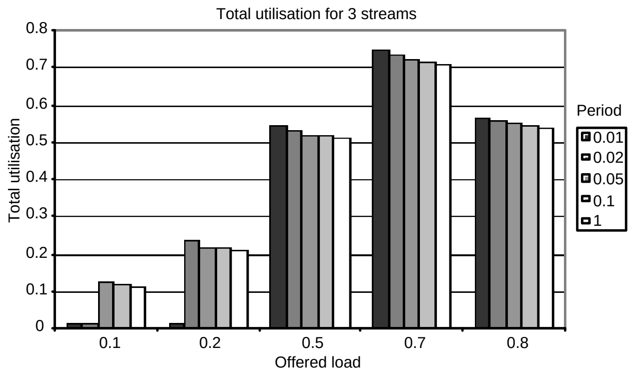

Figure 3. Simulation results

This limit prevents the THT from becoming too small to be of use for the transmitting node. Also, for an offered load of 0.8, not all streams were accepted because the utilisation was limited to 0.8.

The graph shows that the protocol overhead is higher with smaller periods, but not very high.

5.2. The prototype

We have built a prototype [12] of our token protocol on top of Linux [4], where our token protocol is situated between the network and link layer of the OSI model, or between the IP and the ethernet layer in Linux. The token is scheduled using EDF. This prototype revealed that certain modifications to the original design were necessary: adding of nodes and streams to the network was achieved using non-real-time traffic in the original design, but using real-time traffic for this turned out to give better results. Also the fact that different computers have a different sense of time (all clocks are not equal) made changes necessary. These are not discussed here, since clock synchronisation protocols are still being looked into, and the subject of clock synchronisation is not within the context of this text.

It also showed that Linux is not good in providing accurate timing. Tests conducted using RT-Linux [9] and RTAI [8] provided much better results in this area, making them much more suitable for building our network.

Figure 4 shows results of bandwidth measurements conducted between two hosts, using UDP to create four streams [12]. The graph shows the average bandwidth obtained against the amount of data transmitted. The actual bandwidth is lower because of overhead caused by our token protocol and UDP. The graph shows this combined overhead to be ca. 10%. The large fluctuations on the left side of the figure are due to computational inaccuracies.

We also conducted tests where the network was deliberately disturbed. These showed that our token recovery procedures work well. And a test lasting for one month, in which numerous node additions and removals occurred,

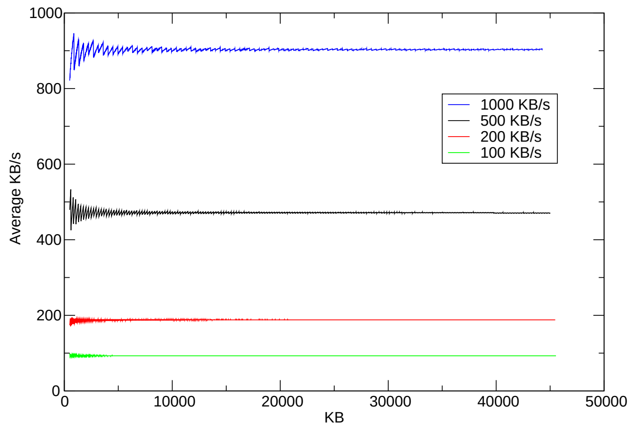

Figure 4. Prototype test results

showed that collisions, which ruin predictable behaviour of ethernet, did not occur during this period.

6. Concluding remarks

This paper describes our current state of research for a real-time network at home. Already we have successfully built a simulation and a prototype of our networking ideas. These can achieve a network utilisation of 100%, have a straightforward feasibility analysis for the real-time streams, and can quickly recover from network errors while doing a good job of maintaining soft real-timeness of the real-time streams.

We are currently looking into IEEE 1394 (Firewire) and IEEE 802.11b (Wireless LAN) as a base network protocol for our network, and we are looking into time synchronisation protocols to solve the timing problems between nodes.

References

[1] CC/PP Working Group web site, `http://www.w3.org/Mobile/CCPP/`

[2] Tjalling Hattink, HOTnet, a real-time network protocol, Master's thesis, University of Twente, 2001

[3] J. Leung and J.W. Whitehead, On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks, *Performance Evaluation*, **2(4)**:237–250, 1982

[4] Linux web site, `http://www.linux.org/`

[5] C.L. Liu and J.W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, *Journal of the ACM*, **20(1)**:40–61, 1973

[6] N. Malcom and W. Zhao, The Timed Token Protocol for Real-Time Communications, *IEEE Computers*, **10(1)**:35–41, January 1994

[7] Resource Description Framework web site, `http://www.w3.org/RDF/`

[8] Real-Time Application Interface for Linux web site, `http://www.aero.polimi.it/~rtai/`

[9] Real-Time Linux web site, `http://www.fsmlabs.com/`

[10] K.C. Sevcik and M.J. Johnson, Cycle Time Properties of the FDDI Token Ring Protocol, *IEEE Transactions on Software Engineering*, **SE-13(3)**:376–385, March 1987

[11] Mark Weiser, Hot Topics: Ubiquitous Computing, *IEEE Computer*, **26(10)**:71–72, October 1993

[12] Jurriaan Wijnberg, Prototyping the real-time 'HOTnet' network protocol on Ethernet, Master's thesis, University of Twente, 2002

[13] X Window System web site, `http://www.x.org/`