An In-Home Digital Network Architecture for Real-Time and Non-Real-Time Communication

Hans Scholten, Pierre G. Jansen, Ferdy Hanssen and Tjalling Hattink
University of Twente, Department of Computer Science
INF 4037, POB 217, 7500 AE Enschede, the Netherlands
email: {scholten, jansen, hanssen}@cs.utwente.nl

This paper describes an in-home digital network architecture that supports both real-time and non-real-time communication. The architecture deploys a distributed token mechanism to schedule communication streams and to offer guaranteed quality-of-service. Essentially, the token mechanism prevents collisions to occur in the network, thus making the network deterministic. The distributed token scheduler uses a preemptive earliest deadline first strategy, which guarantees a possible bandwidth utilization of 100 percent. To allow non-real-time communication however, only part of the available bandwidth is allocated by the scheduler to real-time traffic, typically 80 percent. The paper describes protocols to counter token loss and token duplication. The network is simulated and the paper shows some results from this simulation. Based on low-cost ethernet hardware, a prototype of the network is built and tested. Last, the paper describes future directions.

Keywords: in-home digital network, real-time network, token network, QoS, low-cost

Introduction

Lately, in-home digital networks start to enter our homes, offering possibilities and services that were unknown until recently. Terms like "ubiquitous computing" and "ambient intelligence" come to mind. Examples of such systems are HAVi [HAVI], Jini [JINI] and Universal Plug-and-Play [UPNP]. Often, these systems concentrate solely on multimedia streams and do not offer hard real-time guarantees. They require substantial resources in terms of processing power, memory and energy consumption. Additionally, they are too expensive for inclusion in small devices, like temperature sensors. The proposed real-time network is one of the innovations of a project that tackles these problems.

In the first section of this paper we will give an outline of the At Home Anywhere -@HA- project and its goals. The second section is dedicated to the network, the distributed token scheduler and network protocols. Then we will show some simulation results. The paper ends with conclusions and future directions. Where appropriate, we will comment on related work.

This work is sponsored by the Netherlands Organization for Scientific Research (NWO) under grant number 612.060.111

At Home Anywhere

In the context of in-home networking, where devices have to work together, several challenges do exist. To name just a few:

- *compatibility and interoperability* and
- *network incompatibility:* presently, a house has several separated distribution and communication infrastructures: telephone, cable TV and radio, satellite TV, PC network, connection between thermostat and central heating boiler, etc. In most cases these infrastructures are isolated islands that interconnect only on rare occasions. In general, these infrastructures can 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, WWW browsing, etc. This class uses bandwidth in bursts of data, 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 networks, including wireless networks. Most efforts till now concentrate on only one class of appliances, mostly entertainment.

The first objective of the @HA project is a network for entertainment, control and information that supports both real-time and non-real-time data, as well as streaming media. This network will be based on a new variety of rotating token, 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 worst case, timed-token networks have a utilization that is quite low, around 33 percent. Main properties are described in [SEVCIK] and [MALCOLM]. Examples are IEEE 802.4 token bus, IEEE 802.5 token ring and FDDI. Although not a primary objective, implementation of an IP interface is recommended to maintain compatibility with "normal" IP-based applications;

- *data storage redundancy and data incompatibility:* key issue for integration is a mixed-media storage server, that will provide storage for all appliances and devices at home. The storage server must support streaming media, as this is one of the important data types in the system. One of the functions the storage server must offer is simultaneous real-time recording and playback of multiple video and audio streams on harddisk and optical disc (DVD). Time shifting (the ability to pause live broadcasts) is one of the applications of such a storage server. Information is stored once and will be accessible by all. Because not all appliances use the same data format set, real-time data conversion is needed between storage and appliance. Efficient storage and retrieval of data is essential, as the amount of data in the file server is huge [BOSCH];

• *resources:* even small appliances (resource-lean), like a temperature or light sensor, should be connected to the network, but their size and prize preclude "heavy" processors to accomplish that. @HA researches the concept of delegation, or controlled invocation: small systems use their limited processing power to implement at least a network stack to connect to the network and a small real-time operating system kernel to handle the -lightweight-protocols. But even such an implementation with a small footprint operating system kernel may be too resource-rich. In some cases a simple state machine that handles the network protocols is already enough to do the job. Hardware can range from simple PIC to full-fledged processor system, while software can range from a simple runtime system executing a state machine to an operating system executing a complex program. In our case we distinguish three classes of appliances:

3C (3+ cent) appliance: simple devices that implement only a network stack to connect to the system. There is even no processing power enough to implement the delegation protocol. Network handling is done by a PIC executing a state machine. Memory requirements for this class of devices are low (<1kB). Other devices can scan these devices to get their readings. A temperature sensor is an example of such a device;

3D (3+ dollar) appliance: medium complex devices that implement network stack and delegation protocols on a small, smart card-like embedded processor. Memory requirements are medium, between 8 and 64 kB. Examples are PC peripherals, like printers and scanners;

300D (300+ dollar) appliance: powerful devices, controlled by a complex embedded computer. These devices can scan 3C devices and are used for delegation by 3D devices. The embedded computer can be used for other functions as well, e.g. it can get an electronic program guide (EPG) from the satellite video stream, interpret the EPG and switch on and off the digital video recorder according to the preferences of members of the household. 300D devices have a full-fledged operating system (e.g. Linux) and contain peripheral devices, like disks. Their memory requirements are high (>1MB). Examples are TV and PDA. Delegation is the basis for location transparency: the user interface and underlying application of any appliance are available anywhere in the house. For instance, the settings of the thermostat of the central heating may be inspected and changed on any available display, be it TV set, PDA or PC. The concept of delegation is not new. The X-window system separates application and user interface, so each can execute on a different network node. But this system has a lot of information interchange between client and server and is not suitable. HAVi [havi] does something similar. It distinguishes between controlling and controlled device. Applications and user interfaces (called havlets) are written in Java and allow for flexible and powerful extensions and modifications. But some devices we have in mind might be to small, even for Java virtual machines. HAVi is heavily based on the IEEE1394 network standard [IEEE1394] and uses its underlying protocols. Additionally, HAVi is only meant for audio and video devices;

• *ad hoc configuration:* generally known as plug-and-play. Both network scheduling and lightweight protocols must support robust and fault tolerant dynamic (ad hoc) configuration.

Real-time network

In this section we will concentrate on the network proper: RT-net. First we will give a short-list with requirements and constraints for the network. Then an overview is given of the network: protocol, token, scheduling and feasibility analysis. Token passing and network faults come next. The last subject in this section is network management.

Requirements

The requirements for RT-net are as follows:

• *based on existing hardware and protocols where possible:* this will help to keep the costs down. For instance, ethernet -protocols and hardware- is proven technology, cheap and dependable;

• *QoS guarantees for real-time traffic:* for multimedia transports bandwidth reservations are necessary. Real-time constraints -meeting deadlines- is also important for "command and control" type of messages;

• *open for non-real-time traffic:* there is always a certain amount of non-real-time traffic present in the system. The network must at any time reserve a predetermined amount of bandwidth for this type of communication;

• *fault tolerance:* the physical layer does normally not guarantee that all network packets will arrive. So higher levels in the system should take care of error recovery. This is especially true for tokens. In general, however, it will be impossible to guarantee no deadline misses when network faults occur;

• *plug-and-play:* addition and removal of nodes must be recognized by the network and the token scheduler must react on these events.

Network overview

RT-net is based on the CSMA/CD ethernet protocol [IEEE802]. In effect this means that RT-net can be build on any existing ethernet specification and regular ethernet hardware can be used without modifications. CSMA/CD uses an exponential back-off mechanism to resolve collisions, which makes the network non-deterministic, and thus non-real-time. To make it deterministic, a token-based protocol is deployed to avoid collisions, just like RETHER [RETHER] in development by the University of New York. RETHER distributes its token among the nodes in a simple static round-robin algorithm. RT-net however, uses a more sophisticated algorithm, where the token is allocated to nodes according to their bandwidth demands. A preemptive earliest deadline first (EDF) scheduler is used to determine the route a token follows in the network. This type of scheduler has the advantage over other schedulers that it can achieve a utilization of 100 percent. However, in case of overloading its performance will degrade dramatically. Whenever a node has the token -the active node- it may use the network for some time: the token hold time or THT. THT is determined by the scheduler and is likely to be different for each node. Typically the node will send one frame of a periodic multimedia stream. The EDF scheduler is distributed over the nodes and the token is the place where the schedule is kept -nodes will keep backups of the schedule though-. If a node has the token and it wants to add or remove a stream, it calculates a new schedule and acts upon it. Calculating a new schedule in case of stream addition means the node does an EDF feasibility analysis to determine if a newly added real-time stream will meet its deadlines without making other streams miss theirs. EDF feasibility analysis is simple [BUTTAZZO]:

$$U = \sum_{i=1}^{n} \frac{B_i}{B} \leq 1$$

where B_i is the network bandwidth of stream i and B is the maximum bandwidth of the given network. Actually, the maximum bandwidth is slightly less because of the overhead for token transmission and ethernet packet overhead.

Token passing and network faults

When the scheduler at the active node decides that another node must become active it stores the global schedule information in the token and sends the whole package to the new node. This is a critical action and it should never occur that the token becomes lost, or worse, duplicated. When a token is lost the schedule is lost too and the network stalls. Duplicated tokens mean that more than one node will make schedules and collisions will occur, causing deadline misses and non-deterministic behaviour. To prevent this the concept of monitor is introduced. When the active node relinquishes the token, this node becomes the monitor for the new active node. Monitoring is a three step process: (1) the monitor sends the token to the new active node. If the token is garbled the active node will request a new token. This happens only once. If a second token is garbled too, the active node stays inactive, while the monitor times-out and recovers the token. (2) The active node is now in the "transmission state" for the duration of THT. While the active node transmits, the monitor waits for a reply from the active node. It sets a timer for the duration of THT. (3) At the end of THT the active node must send a reply to the monitor signifying that it is still alive and sends the token to a new active node. The old monitor now ends its activity, the old active node becomes monitor, and the new active node begins transmitting. Then the process starts all over. Many things may go wrong. Detectable token loss situations are: token does not arrive at new active node, reply from active node is lost, active node dies before sending a reply, and the monitor dies. Undetectable token loss occurs when: the active node dies after sending a reply, and monitor and active node die simultaneously. Token duplication is difficult to detect, but there are some hits the network will give: a token arrives at a node that already holds a token, and streams miss their deadlines because nodes cannot resolve bandwidth requirements. If a node detects this, it will delete the token. The network will probably end up without token and initiates a reset.

Network management

Besides scheduling RT-net has to perform some network management as well. We will only summarize network management operations here. Before the network can do anything it has to be initialized. This also happens if a token is lost (network reset). Other management operations are: adding nodes, removing nodes, adding streams and removing streams.

RT-net simulation and prototype

To validate correctness, robustness and usability of RT-net a simulation of the network protocols is made. After that a prototype is realized for demonstration and calibration of the simulation. As simulation tool OMNeT++ [OMNET] is used. Reasons for choosing this package are: it is open source; it uses a well-known implementation language (C++); and it provides dynamic visual feedback and statistics of the simulation. The simulation model has three layers:

- *the ethernet layer:* this layer provides basic support for ethernet packets and the CSMA/CD algorithm. Typical bandwidth is 10 or 100 Mbps. Packet loss ratio is configurable;
- *the RT-net layer:* the actual RT-net is simulated in this layer. It uses the underlying ethernet layer to send and

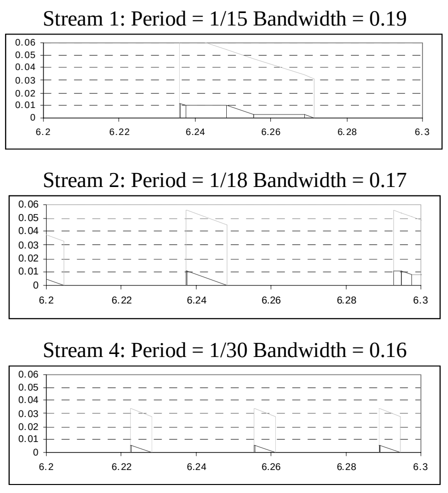

Figure 1: Example simulation output

receive packets, tokens and control messages. It provides a control interface to the application layer for passing events and for creating and deleting real-time streams. The model is decentralized: every node is simulated by a separate module and is governed by its own parameters. Only start-up configuration parameters, like number of nodes and network bandwidth) are shared between all nodes;

• *the application layer:* this layer controls the RT-net simulation and simulates the behaviour of applications. The current application layer can send the following control messages: go on-line, go off-line, and add stream. This set is far from complete, but sufficient for the simulation.

Figure 1 contains dynamic graphical output from the simulator where the EDF scheduling of a set of periodic streams is shown. Only three streams from this set are shown. The grey (light) line depicts the time left until the next absolute deadline for that stream. This time decreases linearly and will cross the horizontal axis at the start of the next period for that stream.The black line is the remaining streaming time for that stream during that period. When the stream is transmitted this line decreases linearly with the number of bytes sent. A horizontal black line clearly shows where the stream is preempted by another stream, so no collisions occur in the network.

From the simulation we found the following characteristics: *the smaller the stream periods, the higher the overhead:* this is caused by the token overhead. When a period is low streams become ready more often and a token has to be sent; *the higher the offered load, the higher the overhead (with some exceptions):* this is the effect of ethernet packet overhead. If a stream uses more bandwidth, more ethernet packets are used. The exception occurs when the offered load exceeds 0.8 maximum bandwidth. Streams will be rejected and the number of running streams is lower than the number of offered streams.

Some figures calculated from the simulation are: average overhead per stream related to total bandwidth: 0.54%; average worst-case overhead per stream related to total bandwidth: 0.78%; average overhead per stream related to effective stream utilization: 1.33%; and average worst-case overhead per stream related to effective stream utilization: 1.95%. Feasibility analysis always calculates the worst-case overhead and uses that value to check that total utilization does not exceed the maximum real-time utilization. This was confirmed by the simulation, where worst-case overhead is higher than the real overhead in every case.

A prototype is realized, based on the Linux operating system and ethernet hardware. Measurements in this prototype confirm the validity of the simulation and its parameters.

Conclusion

This paper describes a new type of real-time network. Unlike other token-based networks RT-net uses a preemptive EDF scheduler to calculate the route of the token and thus the priorities of network communications. The network is based on ethernet hardware and ethernet protocols. Because the token mechanism prevents collisions to occur, the network becomes predictable and suitable for hard real-time purposes. Preemptive EDF scheduling enables a 100 percent utilization of the network. Simulations show, confirmed by a prototype, that average overhead per stream is around 0.5 percent of the total network bandwidth. Currently, work is in progress to port RT-net to IEEE1394 (Firewire). Simulation is almost ready and prototype building starts shortly.

References

[HAVI] Home Audio and Video Interoperability web site: http://www.havi.org
[JINI] JINI homepage: http://www.jini.org or http://www.sun.com/jini
[UPNP UPnP homepage: http://www.upnp.org
[SEVCIK] 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
[MALCOLM] N. Malcolm and W. Zhao: "The Timed Token Protocol for Real-Time Communications", IEEE Computers, 10(1):35-41, January 1994
[BOSCH]H.P.G. Bosch: "Mixed-Media File Systems", Ph.D thesis, University of Twente, ISDN 90-365-1277-8, 1999
[IEEE1394] IEEE1394 web site: http://www.1394ta.org
[IEEE802] Carrier sense multiple access with dollission detections (CSMA/CD) access method and physical layer specifications, ISO/IEC 8802-3 (4th edition), IEEE, 1993
[RETHER] RETHER homepage: http://www.ecsl.cs.sunysb.edu/rether.html
[BUTTAZZO] G.C. Buttazzo: "Hard real-time computing systems", Kluwer Academic Publishers, ISDN 0-7923-9994-3, 1997
[OMNET] OMNet++ homepage: http://www.hit.bme.hu/phd/vargaa/omnetpp.htm