The FABRIC project

Peter van der Stok[1](PR), Jan_Jelle Boomgaardt (TNO), Helmut Burklin (TMM), Gabriele Cecchetti (SSSA), Jean-Dominique Decotignie (CSEM), Hermann de Meer (UP), Gerhard Fohler (MDH), Johan Lukkien (TUE), Gerardo Rubino (INRIA)

[1] Philips Research, prof Holstlaan 4, 5656 AA Eindhoven, Netherlands,
`Peter.van.der.Stok@philips.com`

Abstract. The FABRIC project aims at the integration of middleware standards used in home networks to provide high quality streaming over a heterogeneous network. without introducing new standards.

1 Introduction

At this moment we are confronted by a large set of communication and interoperability standards in the home-networking domain. Within the context of a single standard, devices can be introduced into the home by simply unpacking them and switching them on, possibly accompanied by connecting them to some wiring infra-structure (see [1]). Currently, devices from different standards are completely isolated from each other. Each standard offers specific advantages to the equipment adhering to it and we can safely assume that these standards are here to stay. At the same time the advance of new technology (like 3G roaming devices) will result in newer standards. The users and purchasers of this equipment are not interested and probably not aware of the incompatibilities of standards. Confronted with the cited incompatibilities future customers will NOT acquire these new technologies. Integration of the different standards is absolutely essential to allow a transition from the current islands of technology to one integrated provision of services (roaming or wired) within the home and between the home and service providers.

FABRIC aims at developing *an architecture* in which several interoperability standards and technologies in the home networking context can be integrated. More than integration alone, a FABRIC application manages the complete network to satisfy *Quality of service (QoS)* requirements. The design is guided in this process by the requirements of a chosen application: multiple roaming multimedia streams.

The FABRIC-architecture especially caters for dynamic network configurations allowing frequent additions, removals and roaming of devices.

The FABRIC-architecture supports the provision of real-time multimedia streaming. The timing specification and realization by the FABRIC communication media provides the end-to-end Quality of Service (QoS) required by the multimedia streaming. Reactions to system changes need to be done in a timely manner to support high quality video streaming over wireless media with rapid bandwidth fluctuations (smaller than 10 ms). Configuration changes and video stream settings need to be communicated to the involved devices within time scales of a few 100 ms.

1.2 HLA integration

The FABRIC project investigates the application of the High Level Architecture (HLA) standard [2] to provide the services mentioned above. HLA addresses the interoperability of applications in the large-scale (interactive) simulation domain. (e.g. simulation to support the training of a fighter pilot). HLA assists the integration of the middleware islands. An HLA *federation* is composed of collaborating federates. One federate can be equated to one set of devices belonging to a given standard. Federates can interact in a meaningful way by a common interpretation of the exchanged data. A communication layer, called Run Time Infrastructure (RTI), provides communication services with an Application Programming Interface (API) defined by the HLA Interface Specification. All transfer of data between federates passes through the RTI. The RTI permits to translate the communication between partners. In this way, members of a federate communicate according to their standard. Members of different federates communicate meaningfully a subset of translatable items. HLA supports the dynamic addition and removal of members of a federate. This is realized at the federate level by the standard used within a federate. Between federates it is supported by the ‘subscribe and publish’ facilities of HLA.

Once real-time constraints can be specified and realised under controlled conditions, the distributed decision making algorithms that support stream management, can be simplified and improved. Accordingly, the real-time character of the FABRIC network is exploited to change dynamically the setting of the network in a managed and timely manner and to communicate network changes in a timely manner. This timely management infrastructure is used to support the decision making process on many QoS aspects such as: timeliness, responsiveness, security, flexibility, and reproduction quality of A/V streams. The infrastructure allows a flexible response to changing user needs with limited (managed) network and computing resources.

2 Project organization

FABRIC is funded by the fifth framework IST program of the European Union. There are two industrial partners: Thomson (TMM) and Philips (PR), three large scientific institutes: Centre Suisse d’Electronique et de Microtechnique (CSEM), Institut National de Recherche en Informatique et Automatique (INRIA) and Netherlands Organisation for Applied Scientific Research (TNO), and four universities, Technische Universiteit Eindhoven (TUE), Maelardalen Hoegskola (MDH), University of Passau (UP) and Scuola Superiore St Anna (SSSA). The project is divided in three phases spread over an 18 month period. The project terminates end February 2004.

3 Middleware functionality

The network middleware serves to describe devices and services on the network in a standardized way. These descriptions are communicated to all other connected devices running the same middleware. According to the received descriptions, an application can control a device or service or invoke a service. Such middleware standards are necessary to support the plug and play properties of the connected CE devices in a home network (e.g. see [1]).

An in-home digital network consists of a set of interconnected *physical devices* (devices for short). A device is connected to the network via a digital connector that is realized with one or more chips. Examples of digital connectors are interface cards for an Ethernet cable, or a wireless transceiver. A device, A, is connected to another device, B, when:

a. The digital connector of A can send bit patterns to the digital connector of B such that B can interpret these bit patterns.
b. A is connected to some device C and C is connected to B.

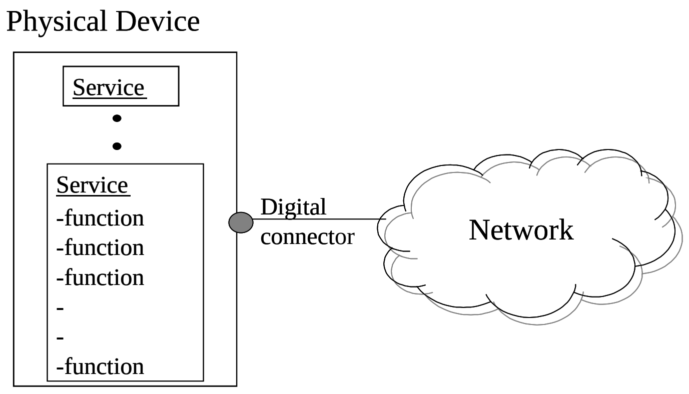

Figure 1, physical device connected to network

For example two devices are connected when they are attached to the same Ethernet wire, or the signal emitted by one device can be received and decoded by the second device. Two devices are also connected when there exists a path via a set of connected devices. On the network, a physical device is identified by the identifier of the digital connector. Therefore, a device has as many identifiers as it has digital connectors. For example a device with two Ethernet cards has two identifiers.

Device *resolution* is making the identifiers of the devices connected to a device, A, available to the applications running on device A. Device *advertisement* is the proclamation by a device of its identifier to a subset of all connected devices. *Discovery* usually covers both aspects.

On a device a set of *service*s (see Figure 1) can deliver a “service” to applications situated on connected physical devices. A service is an instance of a *service type* or service class. An example of a service type is a printing service or a clock service. The meaning and interface of service types is defined by a committee, or by manufacturers, or by individual people. E.G. an individual service type may be the returning of video images of the last 5 minutes taken by a home webcam. A service provides a set of functions that can be invoked, and a set of variables that can be read or set. Functions can modify or monitor the software and associated storage or may act on the physical operation of the device. Service *resolution* is making the identifiers of the services available to an application on a connected device. Service *advertisement* is the proclamation by a device of its services to a subset of all connected devices. Service advertisement and service resolution are often referred to as *Service Discovery*.

4 FABRIC architecture

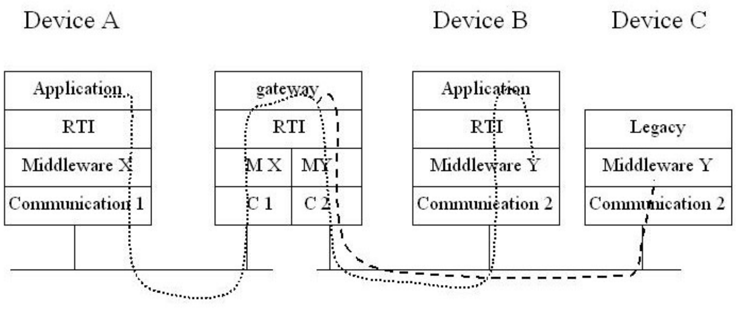

Figure 2 FABRIC architecture

Central to the FABRIC design is that all content (music, videos at different levels of quality) is published. This means that communication is anonymous: beforehand subscribers do not know the identity of the publisher and vice versa. The FABRIC architecture is shown in Figure 2. Devices of a given middleware are interconnected by a communication medium. Devices with different middleware are interconnected by a gateway. Four layers are identified within a device: (1) the application layer where applications publish objects (descriptions of a service instance) and subscribe to classes (description of a service), (2) the Run Time Infrastructure (RTI) that realizes the communication between applications, (3) the middleware layer of standard X or Y that provides and application with standardized access to the services of a device, and (4) the communication layer that interconnects devices. The dotted line shows an example thread of control for an application in device A that publishes information, subscribed to by an application in device B, and transmitted to the underlying service provided by the middleware. The dashed line shows an alternative when the application, for device specific reasons, is made middleware sensitive. The information is then sent directly to the middleware of the destination device. The dotted option is the recommended design. The dashed option is an alternative for specific cases where the functionality of the destination device cannot be handled by the RTI. Three essential FABRIC applications are described in more detail to explain the interoperability results: (1) streaming a video from a server to a renderer, (2) service discovery over middleware standards, and (3) QoS management.

4.1 Streaming video

A streaming application involves the following collaborating entities. The *renderers*, which display a video on a connected display, publish their availability and state. The *video servers*, which produce video, publish the content they can deliver. Consequently, all applications that are connected to a given content or display via the network can subscribe to information about that content or display. A *controller*, which can connect servers to renderers, can decide to show a given content on an available rendering device without knowing beforehand which rendering device to choose. After establishment of the relation between video content and renderer by the controller, the renderer subscribes to the video. Irrespective of the presence of the controlling device, the video will be shown on the rendering device as long as there is a path of sufficient bandwidth between video server and rendering device, and rendering device and video server remain active. New controlling devices may be switched on and connected to the network. By subscription to the contents and rendering devices, the new controlling device can stop or modify on-going video streams.

The following scenario is taken as example

> *"Owner O arrives home with his PDA, which he uses as remote control and positioning device. The home network and the PDA discover each other and an alert appears on the TV. It states that the new DVD movie in the multimedia jukebox is available. O uses the PDA to start the video, which is presented on the TV. The quality of the video is low due to the limited bandwidth. Unfortunately, the jukebox is not able to provide the video in different qualities. After watching, O closes the video stream."*

To realize the scenario, the PDA has subscribed to the published objects in the home network to which it has connected. After reception of the alert, a browser application shows the video with its quality attributes. The video player subscribes to requests for video. On reception of a request from the PDA, the player publishes the video. The chosen renderer receives a request and subscribes to the specified video.

It is important to note that a specific player is selected. The player identity has been communicated to the PDA in the object that is published by the player service.

4.2 service discovery

The purpose of the FABRIC service discovery is to push interoperability as far as possible. The discovery service should hide the differences between the different middleware standards without any loss of functionality provided by the discovered services. Although not encouraged, for diverse reasons an application may want to exploit specific attributes provided by the description of a service by a particular middleware standard. The proposed FABRIC discovery allows application programmers,

at a cost, to define their **own** (local) service interface, possibly conformant to a given middleware standard.
We use the HLA classes to provide the translation between application objects and discovered devices and services. Applications or services can publish objects instantiated from a given class. Applications can subscribe to a set of attributes of a given class. A FABRIC-enabled application subscribes to the descriptions of the services and devices present in the heterogeneous network. It is not known beforehand which devices and services will exist in a network. Consequently, it seems reasonable to define an abstract service and device class that has attributes that allow the identification of the service and contains enough information to reconstitute the complete service description for the application.
The discovery design is split in two parts; (1) FABRIC-enabled Directory Services (FDS-X) that is part of the RTI that links to the discovery middleware of standard X, and (2) the FDS-application (FDS-A) that can publish and subscribe to discovery objects, has access to RTI functionality and coexists with the FDS-X. The service-descriptions and the device-description are published by the FDS-A on the hosting device.

4.3 QoS management

This section introduces a more detailed view of the QoS management approach, called *Matrix*. The Matrix will be composed of several entities that constitute an effective mechanism for monitoring and scheduling available resources in the system. The constituting parts are:

(1) Resource manager schedules and reserves resources
(2) Status Matrix contains information about available resources
(3) Order Matrix contains requests for resource allocations
(4) Order manager allocates resources at a particular device to a specified application
(5) Local scheduler allocates local CPU resources and schedules packets
(6) Local monitor will perceive changes in bandwidth availability.

The resource manager subscribes to quality requests, expressed in the status matrix, that are published by streaming applications. The order manager subscribes to the order matrix to receive orders from the resource manager. Locally the order manager, local scheduler and local monitor enforce the orders, observe the system and publish changes in bandwidth availability in the status matrix.

5. Evaluation

The FABRIC project has successfully completed what it set out to do. The anonymity of the publish/subscribe paradigm can be overridden elegantly when requests needed to be sent to a specific renderer. The HLA standard is very rigid in a predefinition of published classes. This is particular to the development process specified by the HLA standard, but is not a technical necessity. A dynamic addition of new types of devices will need a change to the HLA standard.

Acknowledgements

The FABRIC project is funded in the context of the IST program of the EC fifth framework. Hugh Maaskant and Michael van Hartskamp have improved the presentation of this paper.

References

1. Universal Plug and Play Device Architecture, version 1.0, 2000.
2. IEEE, Standard for modelling and Simulation High Level Architecture, IEEE standard 1516-2000.