CONTROLLING HOME DEVICES BY USING THE VIRTUAL "MOST" AS CLIENT-SERVER COMMUNICATION PROTOCOL

Georg-Otto WAGNER,
Continental Automotive GmbH
Philipsstr. 1, 35576 Wetzlar, Germany
georg.otto.wagner@continental-corporation.com

Keywords: MOST, communication protocol, client-server model, HVAC

Abstract: *This article presents the communication protocol within the MOST (Media Oriented Systems transport) network and it will be demonstrated how is possible to control a home device, e.g. HVAC (Heating, Ventilating and Air Conditioning), by using the virtual "MOST" as client-server communication protocol.*

1. INTRODUCTION

The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests [1].

MOST is a function oriented high-speed multimedia technology to network a variety of devices (namely the MOST nodes). MOST defines mechanisms for sending streaming data and packet-based data, and provides a complete application framework to control interaction between devices in a clearly structured way [2].

The MOST specification[2] defines not only the lower layers of a MOST network, which provide the basics for the transmission of data and for the network management, but also the protocols and mechanisms for implementing applications on top of those [3].

2. "MOST" NETWORK ARCHITECTURE

A MOST device is defined as a physical unit that can be connected to a MOST network via a MOST Network Interface Controller.

In the MOST specification, a MOST device contains multiple components that are the interface of an application and are called function blocks (FBlock). It is also possible that there are multiple FBlocks in a single MOST device connected to the MOST network via a common MOST Network Interface Controller. Between the FBlocks and the MOST Network Interface Controller, the Network Service forms an intermediate layer providing routines to simplify the handling of the MOST Network Interface Controller.

Figure 1 illustrates the model of a MOST device as defined in the MOST specification. On the hardware level, a device has access to the physical transmission layer via the MOST Network Interface Controller (NIC). The Network Service implements a driver layer that controls the access to the interface controller chip and enables applications to carry out basic functions such as sending and receiving messages [3].

Each MOST device implements the Fblock NetBlock, which is required for administrative

tasks within the MOST system. Additionally, each MOST device can implement one or more FBlocks for the functionalities of its corresponding applications.

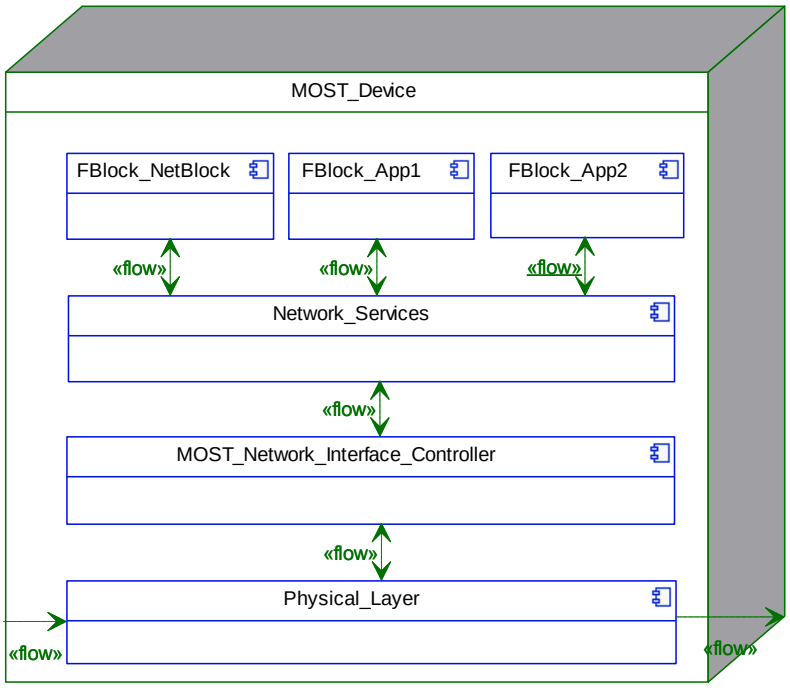

Fig. 1. Model of MOST device

According to the MOST specification, an MOST FBlock, whose functions are used and controlled by an application, is referred to as a Slave. In the figure 2 it is shown how the application, which uses the FBlock and is referred to as its Controller, is setting/reading the properties of the FBlock, is receiving the notification of changes sent by the FBlock, or is calling the methods implemented by the controlled FBlock[3].

It should be done a differentiation between two types of functions implemented by the FBlock[2]:

- Properties - describe a specific attribute of the FBlock(Slave) and its values can be read by the Application(Controller) and, if this is supported by the respective properties, modify them. Additionally, a notification mechanism is defined by the MOST specification – this notification mechanism allows the Application (Controller) to register its interest in particular properties of the FBlock (Slave) and to be informed by the FBlock (Slave) about any changes
- Methods - trigger a certain action within the FBlock (Slave) when they are called by the Controller(application); after performing the method, the FBlock (Slave) returns the results of the execution to the Application(Controller).

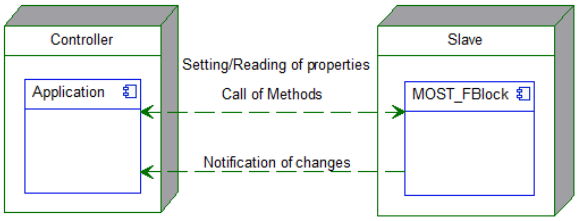

Fig. 2. Interacting with an FBlock (Slave)

3. APPLICATION PROTOCOL APPLIED IN A "MOST" NETWORK

The MOST Specification defines a communication protocol for the interaction between applications in a MOST system. This protocol determines how the services offered by an MOST application can be accessed, describing the data format for the messages, the operational sequences and conditions for

read/write access to the properties and methods of an FBlock, and the notification mechanism.

3.1. DATA FORMAT WITHIN "MOST" PROTOCOL

According to the MOST specification [2] the data format within the MOST systems consists of the following elements:

[DeviceID.]FblockID.InstID.FktID.OPType(Data)

Where the meaning of parameters is:

- DeviceID – this is an optional address identifier of the MOST devices having a size of 16 bits and describing, depending on the context, the receiver, a group of receivers, or the sender of a message. When programming on the application level, the device address is often not used and the addressing will take place only using the functional address
- FBlockID – this is the identifier of the MOST FBlock having a size of 8 bits. The FBlocks below 0xA0 are defined by the MOST specification as required for administrative tasks, called system Fblocks, or characterizing specific device functions or device types. Proprietary Fblocks can be defined by the system integrator (from 0xA0 to 0xEF without 0xC8) or by the manufacturer of the device (from 0xF0 to 0xFE without 0xFC)
- InstID – the instance of the FblockID has a size of 8 bits and is used for a further differentiation between more instances of an FBlock. Both FBlockID and InstID are called the functional address of an FBlock. In case a device implements several Fblocks of a specific type, the InstID=0x00 adresses any instance of the corresponding FBlock on the given device (*don't care*), and InstID=0xFF address all instances on the device (*broadcast*). These wildcards can only be used in the MOST requests, while the response messages must include the correct InstID of the answering FBlock
- FktID – this is the function identifier of the addressed function and has a size of 12 bits. The function type (property or method), the operations supported by the function and the used parameters are defined by the interface description in the FBlock specification. According the MOST specification [2] the structure of the FktID area is illustrated in the table bellow:

Table 1. Structure of the FktID area

Name	**FktID area**	**Description**
Coordination	0x000 to 0x1FF	Used for administrative purposes
Application	0x200 to 0x9FF	Represent the main application functionality of the FBlock
Unique	0xA00 to 0xBFF	Defined unambigously in the entire system
System specific	0xC00 to 0xEFF	Used by a System Integrator
Supplier specific	0xF00 to 0xFFE	Used by a device manufacturer

- OPType – the operation to be applied to the function depending on its type (property or method). The size of OPType is 4 bits. The table below shows all 16 individual operations can be applied to a function:

Table 2. Operation types

	OPType for Properties	**OPType of Methods**	**OPType**
Request	Set	Start	0x0
	Get	Abort	0x1
	SetGet	StartResult	0x2
	Increment	Reserved	0x3
	Decrement	Reserved	0x4
	GetInterface	GetInterface	0x5
	-	StartResultAck	0x6
	-	AbortAck	0x7
	-	StartAck	0x8
Responses	ErrorAck	ErrorAck	0x9
	-	ProcessingAck	0xA
	Reserved	Processing	0xB
	Status	Result	0xC
	-	ResultAck	0xD
	Interface	Interface	0xE
	Error	Error	0xF

- Data – the data field consisting of a length indication and the data area for the parameters of the function. According the MOST specification [2], the parameter types are defined as following:

Table 3. Parameter types of the MOST application protocol

Parameter	Size	Description
Boolean	1 byte	Boolean value
BitField	1,2,4, or 8 bytes	Bit field with Mask
Enum	1 byte	Enumeration
Unsigned Byte	1byte	Integer value (without sign)
Signed Byte	1 byte	Signed integer value
Unsigned Word	2 bytes	Integer value (without sign)
Signed Word	2 bytes	Signed integer value
Unsigned Long	4 bytes	Integer value (without sign)
Signed Long	4 bytes	Signed integer value
String	Variable	Null terminated string with encoding information
Stream Classified	Variable	A stream of arbitrary data
Stream	Variable	A stream with type information
Short Stream	Variable (up to 255 bytes)	Short stream with length information

3.2. THE DYNAMIC BEHAVIOR WITHIN "MOST" PROTOCOL

In the following section is described the dynamic behavior within the MOST protocol by using the UML following sequence diagrams.

Figure 3 below shows the querying of property's values with the operation "Get"

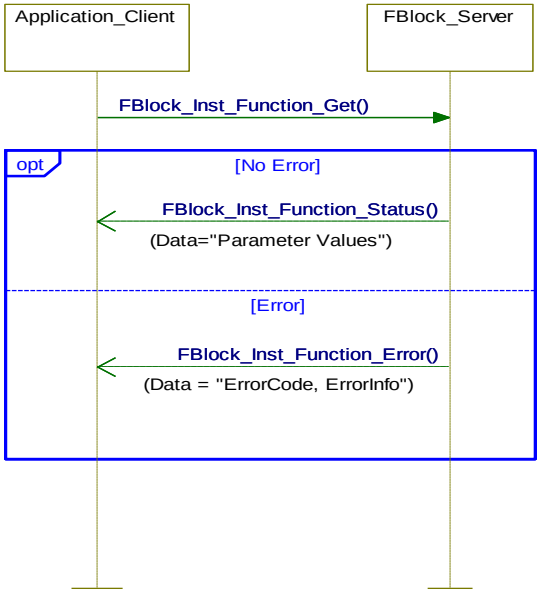

Fig. 3. Querying the values of a property with the operation "Get"

In the figure 4 is presented how is possible to set the value of a property with the operation "Set"

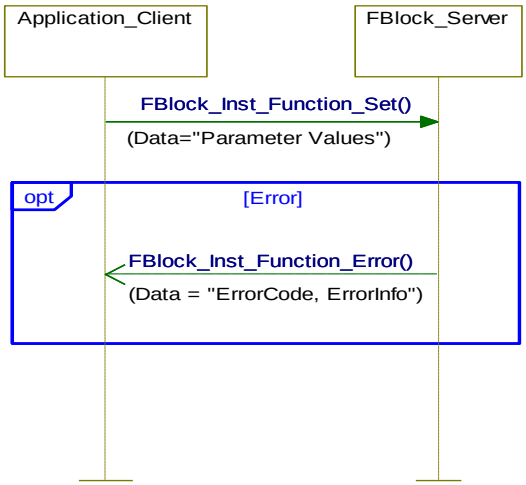

Fig. 4. Setting the value of a property with the operation "Set"

In the figure 5 below presents how to set the value of a property with the operation "SetGet":

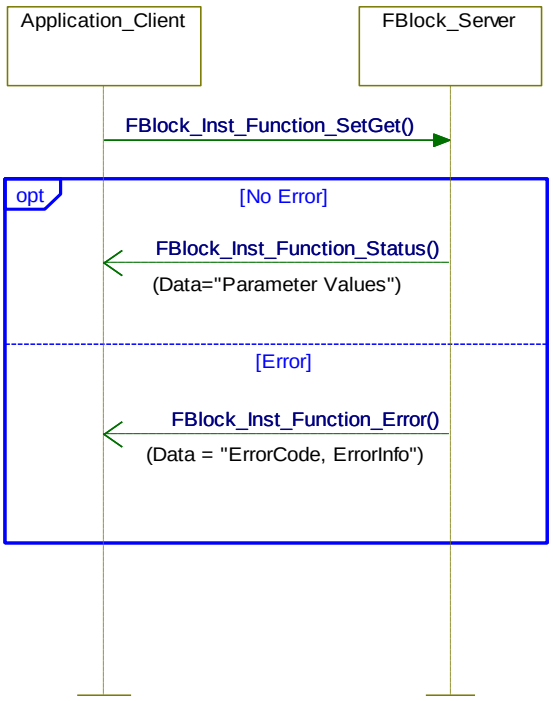

Fig. 5. Setting the value of a property with the operation "SetGet"

Figure 6 below shows the setting of a method with the operation "StartAck" :

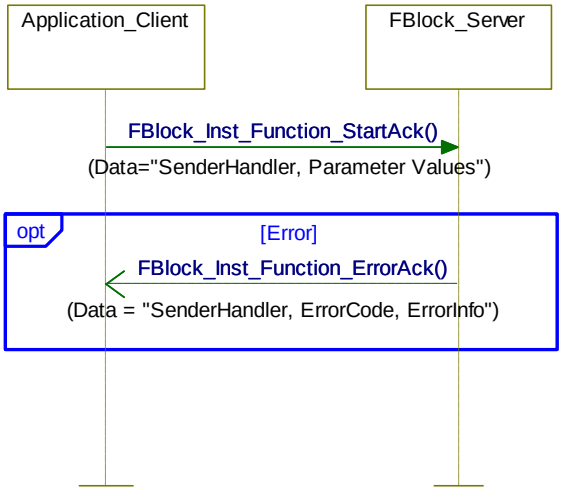

Fig. 6. Setting a method with the operation "StartAck"

Figure 7 below shows the setting of a method with the operation "StartResultAck" :

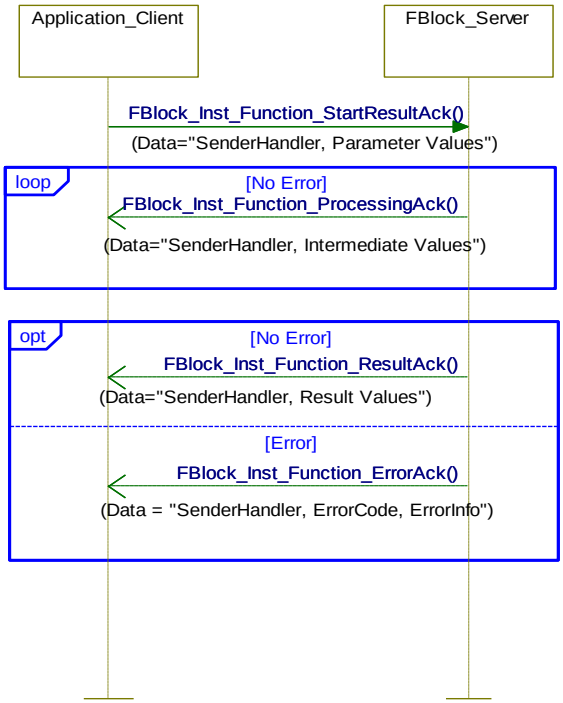

Fig. 7. Setting a method with the operation "StartResultAck"

4. NOTIFICATION MECHANISM WITHIN "MOST" PROTOCOL

As defined by the MOST Specification [2], in order to avoid constantly polling of a specific property performed by the Application (Controller) by using a "Get" message, a notification mechanism was implemented within the MOST protocol. The properties that are supporting a notification mechanism are determined when an Fbock is defined.

The following notification operations are currently defined within the MOST notification mechanism [2]:

Tabel 4. Notification operations

Notification operation	Description
SetAll	Request notification of device for all the FBlock's properties for which a notification is supported
SetFunction	Request notification of device for all the FBlock's properties indicated in a subsequent list
ClearAll	Delete all notifications of the indicated device for all notified properties of the FBlock
ClearFunction	Deletes notification of the device for the properties of FBlock indicated in a subsequent list

Figure 8 below illustrates how the notification mechanism is working according to the defined MOST protocol:

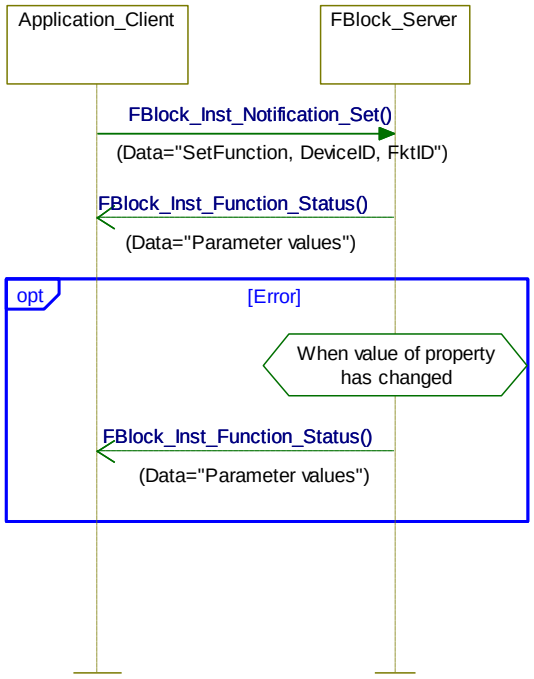

Fig. 8. Notification mechanism

5. CONTROLLING AN HVAC DEVICE BY USING THE VIRTUAL "MOST" PROTOCOL

The figure 9 illustrates the main components involved in the proposed communication based on the virtual MOST protocol:

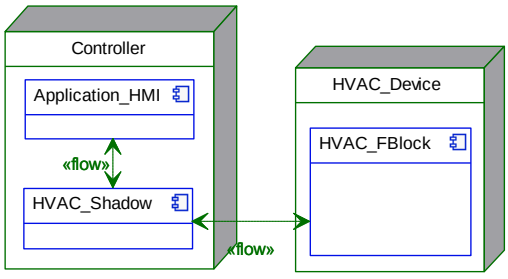

Fig. 9. Main components involved in the virtual MOST communication

- HVAC Device – the controlled HVAC device
- Controller - which consist of two subcomponents: the application/HMI and the HVAC Shadow

The following use-cases are requested to be considered:

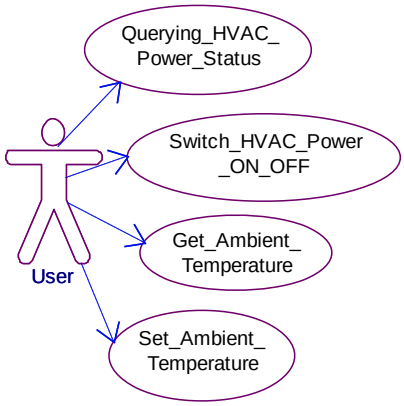

Fig. 10. Use-cases for HVAC application

In the figure 11 is presented the UML sequence diagram for controlling an HVAC device by using the virtual "MOST" protocol.

The main actors involved in the communication are:

- the Application/HMI – which represent the virtual Controller within the MOST communication
- the HVAC FBlock – which represent the logical Slave within the MOST communication
- HVAC – the device itself

Fig. 11. Sequence diagram for controlling an HVAC device using virtual "MOST" protocol

As illustrated in the figure 11, before the application tries to get the status from the HVAC device by calling the MOST property "Get", it registers all FBlock's methods for which a notification is supported with help of the MOST notification mechanism. If the power status of the HVAC device is Off, then it will be switched On by calling the MOST property "SetGet". The Application will get the ambient temperature from the HVAC. If the user changes the ambient temperature to 23°C, then the MOST method "StartResultAck" can be called in order to transmit the new temperature value to the HVAC device through its associated MOST FBlock. If the ambient temperature is not yet reached, then a MOST "ProcessingAck" will be sent to the application every 500msec, otherwise a MOST "ResultAck" with the indication "ambient temperature reached" will be sent as answer to the application. After the ambient temperature is

reached, the application can switch off the HVAC device and the notifications for all the notified functions can be deleted from the Notification Matrix.

6. CONCLUSIONS

This article presented the communication protocol within the MOST (Media Oriented Systems transport) network and demonstrated that it is possible to control a home device, e.g. HVAC(Heating, Ventilating and Air Conditioning), by using the virtual "MOST" as client-server communication protocol.

In conclusion, MOST is not only a network that has mechanisms to transport all the various signals and data streams that occur in multimedia and infotainment systems, but it can be also successfully used for controlling the home devices.

7. REFERENCES

[1]. Wikipedia, the free encyclopedia, "Client–server model", accessed 06.06.2011: http://en.wikipedia.org/wiki/Client%E2%80%93server_model

[2]. MOST Cooperation, "*MOST – Media Oriented Systems Transport*" MOST Specification Rev. 3.0 E2, 07/2010.

[3]. A. Grzemba, "*MOST – The automotive multimedia network*", ISBN 978-3-645-65061-8, 2011.