Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Architectural Concept of Virtual Automation Networks

Peter Neumann, Axel Poeschmann, Ralf Messerschmidt

ifak Magdeburg, Germany (Tel: +49 39203 81077;
e-mail: peter.neumann|axel.poeschmann|ralf.messerschmidt@ifak.eu).

Abstract: Twenty years ago, fieldbus systems have been developed for the factory floor and are widely introduced in industrial automation installations in the meantime. In the nineties Ethernet-based technologies have been successfully introduced to office automation followed by the usage of Internet-based applications. Today industrial automation is getting penetrated by the same IT technologies used in office automation. Nevertheless, the factory floor, respectively automation science and practice, has special requirements that clearly differ from IT-world requirements; this especially concerns safety, security, real-time, wireless and public network integration aspects. Thus – to make information technologies applicable in all fields of production industries – these technologies have to be adopted, extended or even modified. This significant challenge is extensively tackled by the European Integrated Project "Virtual Automation Networks". The exchange of the industrial automation network base towards IT technologies also offers the unique change to build an open network platform providing easy integration and extendibility capabilities. This paper presents the open system architecture and first implementation steps for realizing a Virtual Automation Network. The concepts aimed at are characterized by modularity and intelligence, thus, enabling flexibility and re-configurability focusing on future knowledge-based and agile manufacturing enterprises.

Keywords: Industrial Communications, Distributed Control Systems, Heterogeneous Networks.

1. INTRODUCTION

Digital networking in industrial automation has a long history. For the last 20 years, digital communications have been widely introduced in distributed computer control systems within both the factory and the process domain. The proprietary communication systems within SCADA systems have been supplemented and partially displaced by fieldbus systems and sensor bus systems. The introduction of fieldbus systems has been associated with a change of paradigm to deploy industrial automation systems, emphasising the device's autonomy and decentralised decision making and control loops. Nowadays, (wired) fieldbus systems are standardised. They are the most important communication systems used in commercial control installations.

Ethernet won the battle at the same time as the most often used communication technology within the office domain. It results in low component prices caused by the mass production of these components. Nowadays, there is a large community inventing and introducing Ethernet-based communication systems to be used in the industrial automation domain, e.g. in the harsh environment, and in a real-time and safety-critical world. Thus, Ethernet-based solutions are dominating as a merging technology.

Additionally, wireless communications have been introduced in the meantime in both the office environment and the workshop area. Wireless technologies have been increasingly investigated and standardisation is continuing. Following the trend to merge the automation as well as the office networks, heterogeneous networks, consisting of local and wide area as well as wired and wireless communication systems, are getting important. But to use communication systems within the Industrial Automation they have to fulfil special requirements. The main requirements are (Neumann, 2007): (1) Guaranty of real-time behaviour; (2) Guaranty of functional safety; (3) Guaranty of security; (4) Location awareness.

2. BASIC DECISIONS

The aim of a Virtual Automation Network is to handle successfully the transfer of data through a heterogeneous communication network from the point of view of an automation application. The passed communication systems are of any technology, which can not be influenced by the automation expert. Thus, the heterogeneous technologies are given. VAN does not describe a new communication protocol. Thus, the aim is to use the greatest amount of legacy LAN, WAN and industrial communications mechanisms. From the point of view of an automation application, the specifics of the heterogeneous network have to be hidden. Due to this, the following basic design decisions could be made:

- VAN is an infrastructure for tested standard distributed industrial automation concepts in an extended environment. The application functions (productive automation functions) are described by their object

models used in existing industrial communications. The application service elements (ASEs), as they are specified in the IEC 61158 standard, can additionally be used. Thus, the results of the VAN development can be taken over in the further IEC communication standardisation.

- Web Services will be used for the establishment of the end-to-end connections between distributed objects within a heterogeneous network. Once this connection has been established, the runtime channel between these objects is equivalent to the runtime channel within the local area by using IEC 61784-2 CPF 3 runtime mechanisms.
- To avoid the use of IP and MAC addresses during establishing the end-to-end path between logically connected applications within a VAN domain, the VAN addressing scheme is based on names. This means for the connected application objects, the IP and MAC addresses remain hidden.
- None new specified application layer is necessary, since there is no new fieldbus or real-time Ethernet protocol. Thus, the approved models of industrial communications can be used. Only the additional requirements, caused by the influence of Wide Area Networks, have to be considered and lead to additional functionality following the mentioned design guidelines.

An alternative approach could be to develop a new VAN-specific application layer. It means that the establishment of the runtime tunnel as well as the establishment of the connection between the distributed application (automation) objects are merging. As a consequence, the name-based addressing will not be available. Furthermore, the application layer protocols, which are well introduced on the market, cannot be used further.

3. NETWORK TOPOLOGY

The VAN characteristics are defined for domains (VAN, 2006a, and b). A domain is a logical group of VAN devices. The expression "domain" addresses areas and devices with common properties / behaviour and mainly common application purposes. The aim is to connect different

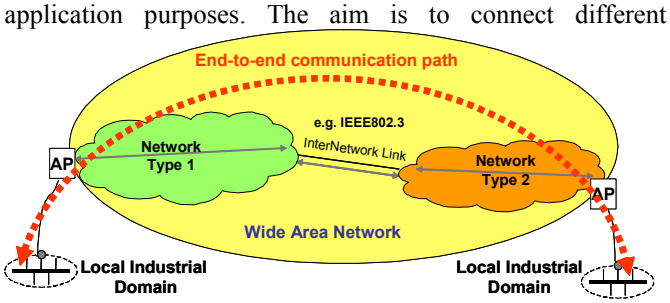

Fig.1 Communication path (example) (AP – Access Point)

industrial sites by means of different network technologies with Quality of Service for industrial control. The devices related to a VAN domain may reside in a homogeneous network domain (e.g. local industrial domain in Figures 2 and 3). Depending on the application, additional VAN relevant devices may be only reached by crossing other network types (e.g. Wide Area Network type communication, Figure 1) or they even need to use proxy technology to be represented in the VAN domain view of a complex application.

3.1 VAN domain and addressing concept

A VAN domain covers all devices which shall be grouped together on a logical or virtual basis to represent a complex (e.g. industrial) application, e.g. forming a VAN domain by designing a distributed control application using existing devices (e.g. PROFINET devices). A domain uses relations between VAN devices, which are the end points of a configured communication line, the local domain and the related other servers (VAN PnP etc.). The type of network and the location of the devices may be of any kind and somewhere distributed over a physical environment that shall be covered by the overall application. All devices that have to exchange information within the scope of the application (equals to a VAN domain) must be VAN aware or VAN enabled devices. Otherwise, they are VAN independent and are not members of a VAN domain. Figure 2 depicts VAN domain examples representing three different distributed applications.

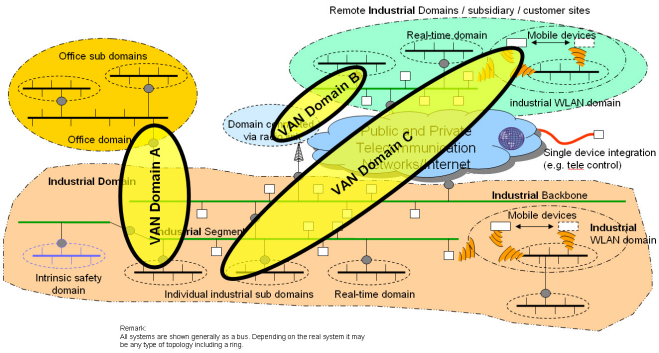

Fig.2: Different VAN domains related to different automation applications (VAN, 2006a)

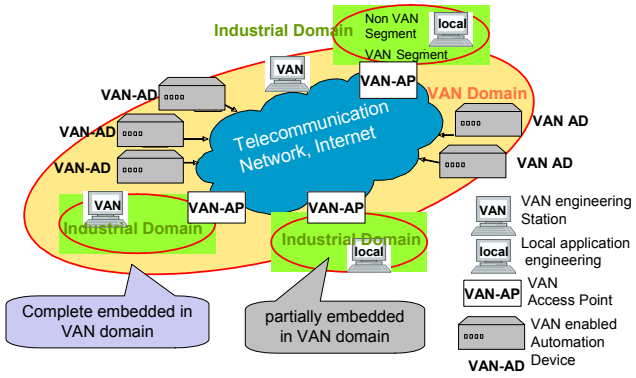

Fig.3: Parts of a domain

The VAN domain can include given industrial domains (equipped with automation devices and internally connected via industrial communications, e.g. fieldbus) completely or partially. Thus, an industrial domain can consist of segments related to a VAN domain (VAN segment) and/or of segments which are not related to a VAN segment (Figure 3).

For a VAN, there are defined various components:
(1) VAN infrastructure components
- VAN Access Point (VAN-AP)
- VAN Server Device (VAN-SVD)
- VAN Security Infrastructure Device (VAN-SID)
- VAN Engineering Station

(2) Automation-specific components
- VAN Automation Device (VAN-AD)
- VAN Proxy Device (VAN-PD)
- VAN Virtual Device (VAN-VD).

Of interest in the context of this contribution are:

- VAN Access Point: It connects VAN network segments, but it does not contain an automation function or an automation application process. It can work as a gateway or a router to automation devices that are members in a VAN application context (VAN domain). A VAN-AP may handle different transmission technologies, which contains all relevant administration functions, necessary for configuration and parameter setting in connected sub-networks. It enables the switching between available communication paths and finds the best route through the different sub-networks of the heterogeneous VAN network.
- VAN Automation Device: Containing an automation function and connecting it through its integrated VAN communication capabilities.
- VAN Proxy Device: Connecting a network segment (e.g. connecting automation devices using any industrial communication system).

They distinguish by their internal structure as shown in Figure 4.

3.2 Addressing concept

A VAN domain represents a name space of a distributed automation application following the design decision to use logical addresses (names). It is independent of the used technologies in the installed equipment as well as the used address schemes (e.g. IP and MAC addresses) within the heterogeneous network. For the establishment of an end-to-end connection and the management functions, a logical name addressing is used. It means: The VAN-Domain is a Name Domain and not an IP Address Domain. Each VAN device shall have a unique name according to the naming conventions. To address VAN devices in a VAN domain, a unique name space is used. If a provider switching to another provider occurs (automatic change by a VAN-AP internal event or initiated by an external event), the unique name of a VAN device within a VAN-Domain shall not be changed. A VAN-AP object "VAN-switching" is responsible for handling this. The naming conventions within a VAN-Domain shall be conform to DNS unique indicator and conform to IEC 61158 Type 10 naming definitions to avoid naming conflicts (IEC 61158).

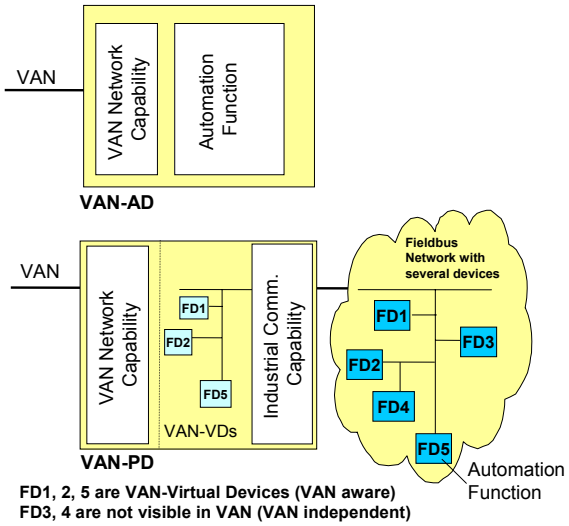

Fig.4: Automation-specific components (AD – Automation device; PD – Proxy device; VD - Virtual device; FD – Field device)

Address resolving in a VAN with sub domains and communication paths with different IP-subnets will be done by DNS service, following the rules as follows. Each area with different IP subnet addressing connected via a VAN access point has to be a unique sub domain. This is necessary to access the sub domains by using standard DNS mechanism. Inside a sub domain, additional sub domains can be defined depending on the physical and logical structure of the network installation. The name space can be a public name space or a local (private) name space.

Once a runtime tunnel has been established and the exchange of productive data takes place, the IP/MAC addresses, which have been negotiated during the connection establishment phase, should be used.

4. SYSTEM ARCHITECTURE

4.1 Logic components

The logic components of a system to be implemented within the distributed equipment of an automation project are normally not allocated to a single device. To give an overview, they should be presented in a common system architecture block scheme (Figures 5 and 6). The blocks have to be allocated to specific devices partially, related to the device type and communication profile.

4.2 Functionality

A lot of standard functionality can be used: Standard Network Technologies; Internet; IEC 61158.

The VAN-specific functionality is related to: VAN Common Communication Application Service Elements (ASEs) (VAN, 2006a); Runtime object tunnel and runtime object dispatcher.

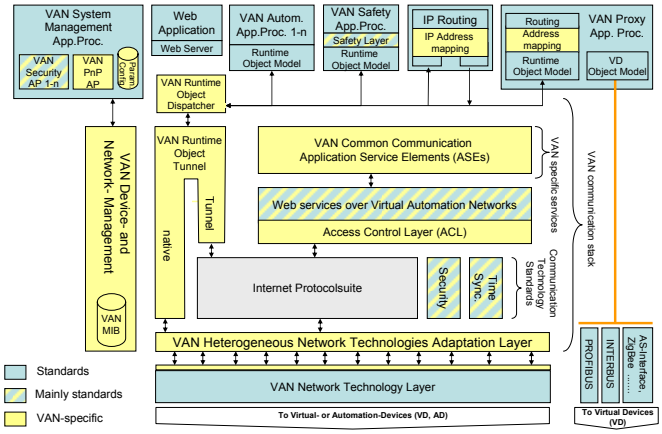

Fig. 5: VAN Architecture

The object-oriented ASEs are crucial components, since they contain all the data for the VAN functionality. All attributes of an ASE can be accessed by uniform defined Get and Set services. The access rights are defined in the Access Control List (ACL). Fig. 6 depicts the ASEs defined in VAN.

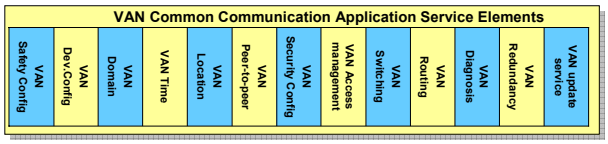

Fig. 6: VAN Common Communication ASEs

Additionally, there are VAN-specific modifications and extensions of standard functionality: (1) Adaptation of the network technologies in the lower layers; (2) WEB Services over VAN (interface between Internet and VAN Common Communication ASEs with an underlying Access Control Layer to secure the access); (3) Time Synchronisation; (4) Security mechanisms (VAN Network and VAN System management Application Process); (5) Safety mechanisms (embedded in VAN Safety Application Process).

4.3 End-to-end communication

To prepare the productive data exchange within an end-to-end communication of application data objects, two phases of the connection establishment have to be considered:

(1) The establishment of the runtime tunnel, i.e. to organise the VAN infrastructure, comparable with laying a (virtual) wired line between the application processes (e.g. automation application process handling of the automation objects to be exchanged). For this Web Services are used. As a result of this phase the runtime tunnel will be activated. After this follows:

(2) The establishment of the connection between the application objects itself (e.g. IEC 61784-2 CPF 3 application objects to be exchanged) using the established runtime channel. This connection establishment follows the rules of the protocols which are used to realise the distributed application processes (e.g. IEC 61784-2 CPF 3), and is therefore not in the scope of VAN (instead within the scope of e.g. IEC 61784-2 CPF 3). The VAN architecture allows the use of any application layer object model. For the VAN prototyping, the runtime object model of IEC 61784-2 CPF 3 will be used as a common object definition (IEC 61784-2).

4.4 Quality of Service

There are several aspects to guarantee a scalable Quality of Service:

- Temporal behaviour (real-time performance), see separate paper (Beran J, Zezulka F, 2008).
- Availability of the end-to-end connection between distributed application (automation) objects. To guarantee that, there are introduced VAN-specific switching and routing mechanisms. VAN switching means: change of alternative transmission technologies (lines) between VAN Access Points, or the change of providers. The runtime tunnel remains open (working). VAN routing means: a metric based finding of a complete end-to-end VAN route through the heterogeneous network and also applies in case of interruption and newly establishment of the runtime tunnel. Precondition for all these mechanisms is a QoS monitoring of an actual link.
- Functional Safety has been realised by a specific safety protocol on top of the application layer.
- Security has been investigated in different facets; see separate paper (Wolframm M, Adamczyk H, 2008).

5. IMPLEMENTATION ASPECTS

As described before, there are two main communication principles in VAN: the Web Service based for the connection establishment, and the Runtime Object Tunnel (e.g. using openVPN) for the runtime data exchange. The following chapter deals with the Web Service related implementation aspects.

5.1 Web Service related Software Architecture

VAN communication is Peer-to-Peer communication, which means it is not hierarchical; therefore each VAN device implements a Web Server and a Web Client to be able to receive and send Web Services.

The Web Service related VAN software architecture of a VAN device is shown in the next figure. The WS Server, Broker with Registry, and Client are central instances of any VAN device. The WS Server terminates each incoming WS, checks for authorization against the Access Control List (ACL) and if allowed the broker distributes the WS to the corresponding object of an ASE (VAN, 2007). For this the WS broker has an Object Registry where all implemented VAN ASE object instances shall register to be enabled to

receive messages via Web Services. The object registry is dynamic. The respective object will register/unregistered at the Object registry with its ASE type, Object Instance port and object reference. Thus the Object Registry of the WS Broker

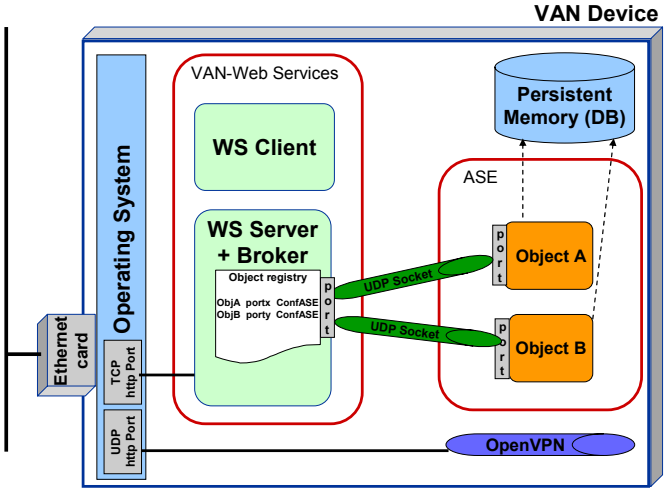

Fig. 7: VAN Web Service related Software Architecture of a VAN device

keeps a list with entries of all objects that have registered themselves (see figure 8 for an example). Each object reference within an ASE is defined as unique.

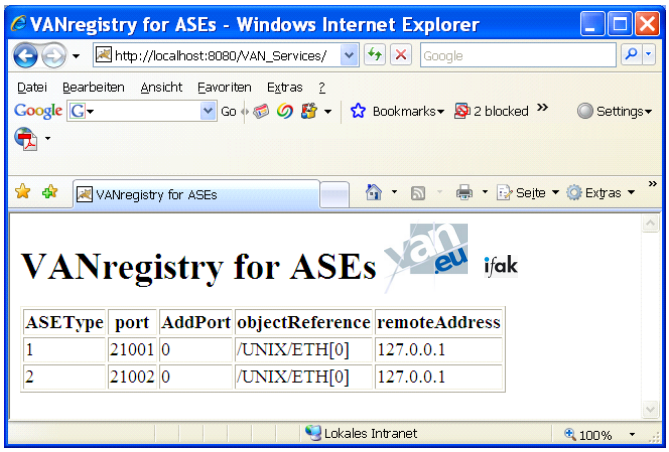

Fig. 8: Example of entries in the VAN Registry for ASE objects

5.2 Interface Definition

The figures 9 and 10 illustrate the two needed WS related Software Interfaces:

- one for receiving the WS from the network/OS and
- one for the internal ASE objects which are subject of finally receiving, processing and replying to the WS contained messages.

The interaction of components for a request sent by a VAN-device A to a VAN-device B and the resulting confirmation is shown in figure 10. The VAN ASE process of Device A uses its WS Client for sending out the WS Get request. Device B receives the WS Get request via its External WS interface. As described above targeted object and via the internal socket interface the object registry provides the information how to reach the

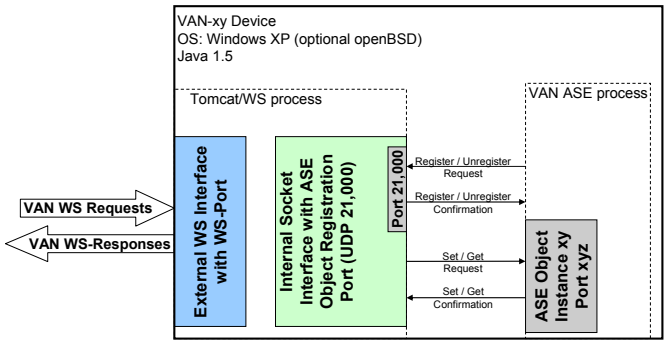

Fig. 9: Illustration of the Web Services related Software Interfaces

Message, which passed to its destination. The destination ASE responses with a confirmation, which is send the same way back.

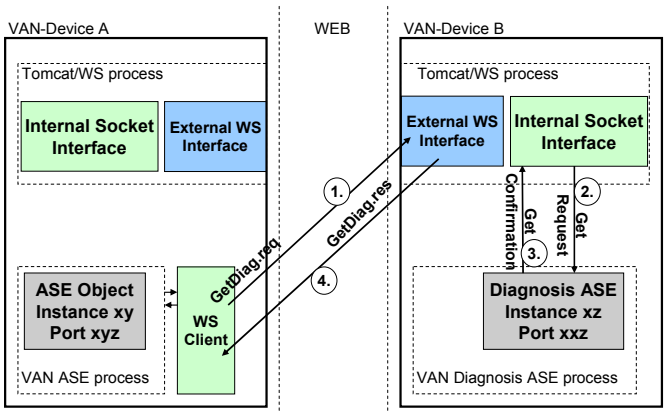

Fig. 10: Get Request handling between two VAN devices

In the following subchapters the two Interfaces are described more in detail.

5.2.1 External Web-Service Interface

The External Web-Service Interface receives all incoming WS Requests from the network/OS and sends out the related WS Responses. Therefore it is bound to the TCP http(s) port. This interface implements the VAN Web Service server according to a WSDL File that is valid for all VAN devices.

The nature of a Web Service communication is symmetric, that means it is expected that a request is followed by a related response. In case that the Web Server receives a request from another VAN device and can not distribute it to the targeted object, the request will be sent back to the requesting device without any error code.

At the time of sending of the Web request a timer has to be started in the sending VAN device. This is for the case that a VAN device's Web Client sends a Web Service and the

targeted VAN device's Web Server receives the request and distributes it to the targeted object, but by any fault the object does not reply. If the timer runs out (and no related response is received) the respective thread has to be finished. If a response is received from the according object, the timer (and the related process) has to be stopped.

5.2.2 Internal WP2 Socket Interface

The interface between Web Server and ASEs bases on UDP Sockets. All VAN ASE object instances of a VAN device shall register at the ASE Object Registration Port - UDP Port: 21,000 - by means of the Register service. The registration shall contain a parameter set for the unambiguously allocation consisting of: object reference, ASE type and ASE Object Instance UDP Port.

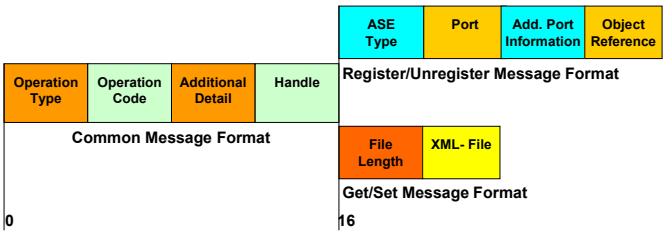

Fig. 11: Internal WP2 Socket Interface Message Formats

This interface defines two different message types, on the one hand for Registering and Unregistering of ASE object instances and on the other hand for the exchange of ASE related Get and Set messages for the WS based exchange between ASEs and devices. As shown in Figure 11 the interface related messages are divided in a common part (Byte 0-15) accomplished by two special parts (starting from byte 16), one defining the format used for Register/Unregister, the other defining the format used for Get/Set.

The above described Web Services are used to establish the Runtime Object Tunnel connecting the application objects of different VAN devices (see Fig. 12, and also chapter 4.3).

5.3 VAN Runtime Object tunnel

The communication of the automation application objects is referred to as VAN runtime tunnel. If a connection target is on a VAN device within the local net, native protocols can be used (see also Fig. 5). For crossing any WAN, private or public, a tunnel will be established between the end points. The VAN project decided on using openVPN, it also provides security features and is firewall friendly. To be able to transport both MAC and IP based packets TAP devices are used at the termination points of a tunnel. The tunnel itself uses the UDP protocol. Usually an end-to-end tunnel will consist of several cascaded single tunnel segments. The related tunnel configuration data for a connection are contained in respective DeviceConfig ASE objects in the end points; operational parameters are kept in correlated Domain ASE objects. Latter Domain ASE objects for each connection also have to be held on all VAN devices which are part of the path, to be able to interconnect the right tunnel segments. Since the path is not predefined, but will be defined dynamically during the tunnel establishment by the VAN Routing also the distribution of that information has to be handled by the tunnel establishment.

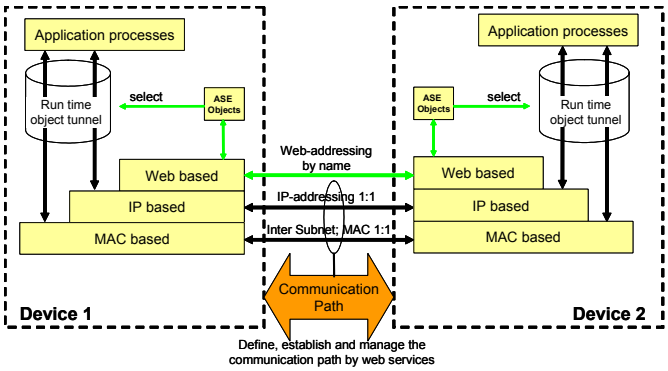

Fig. 12: Optimised Runtime Communication Stack

Details of the tunnelling and the closely related security items are described in (Wolframm M, Adamczyk H, 2008); performance aspects of the VAN Runtime Object tunnel are discussed in (Beran J, Zezulka F, 2008).

REFERENCES

Neumann, P. (2007). Communication in Industrial Automation. What is going on? *Control Engineering Practice* 15 (2007), pp. 1332-1347.

VAN (2006a). Specification of the Open Platform for Automation Infrastructure. Deliverable D02.2-1. Topology Architecture for the VAN virtual Automation Domain. European Integrated Project VAN FP6/2004/IST/NMP/2 - 016696 VAN Virtual Automation Networks.

VAN (2006b). Specification of the Open Platform for Automation Infrastructure. Deliverable D02.2-2: VAN Open Platform API-Specification. European Integrated Project VAN FP6/2004/IST/NMP/2 - 016696 VAN Virtual Automation Networks.

VAN (2007). Prototype Implementation Integration. Deliverable D02.4-1: Software Architecture and Interface Specification. European Integrated Project VAN FP6/2004/IST/NMP/2 - 016696 VAN Virtual Automation Networks.

IEC 61784-2, Industrial communication networks Profiles Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3.

IEC 61158 (all parts) Industrial communication networks Fieldbus specifications.

Beran J, Zezulka F (2008). Evaluation of real-time behaviour in Virtual Automation Networks. Invited Session "Virtual Automation Networks". IFAC World Congress 2008, Seoul.

Wolframm M, Adamczyk H (2008). Secure Virtual Automation Networks based on Generic Procedure Model. Invited Session "Virtual Automation Networks". IFAC World Congress 2008, Seoul.