Prof. Dr.-Ing. Dipl.-Ing. Dipl. Wirt.-Ing. Axel Sikora
(axel.sikora@hs-offenburg.de)
leitet das Institut für verlässliche Embedded Systeme und Kommunikationselektronik (ivESK) der Hochschule Offenburg. Außerdem ist er stellvertretender Institutsleiter am Hahn-Schickard-Institut für Mikro- und Informationstechnik in Villingen-Schwenningen, wo er für den Bereich „Software Solutions" verantwortlich ist.

emb::6 – ein Open-Source-Protokollstapel für das IPv6-basierte Internet der Dinge mit 6Lo

Das Internet der Dinge ist auf dem Vormarsch. Immer mehr mikroelektronische Systeme werden vernetzt und bilden cyber-physische Systeme (CPS). Allerdings gibt es weiterhin eine Unmenge von Protokollen – vor allem für den Nahbereichsfunk, die sogenannten Short-Range Wireless Networks. Seit Jahren ist auch IPv6 über IEEE802.15.4, bekannt als 6LoWPAN, am Start. Nachdem die ersten Jahre eher holprig verliefen, sind nun wesentliche Weichen gestellt und seit etwa drei Jahren die grundlegenden Standards stabil. Auf dieser Grundlage sind bereits zahlreiche quelloffene und kommerzielle Lösungen verfügbar. In diesem Artikel wird das emb::6 Open-Source-Projekt des Instituts für verlässliche Embedded Systems und Kommunikationselektronik (ivESK) der Hochschule Offenburg vorgestellt, das mit einer Reihe von Alleinstellungsmerkmalen dazu beitragen möchte, eine effiziente, sichere, stabile und einfach zu nutzende Implementierung voranzubringen, die eine übergreifende und medienunabhängige Vernetzung für das Internet der Dinge bereitstellt. Dabei bietet die vorgelegte Umsetzung eine Reihe von einzigartigen Eigenschaften, wie z. B. die Betriebssystemunabhängigkeit, ein guter Kompromiss zwischen Modularität und Effizienz, eine exzellente Portierbarkeit und sehr umfangreiche Schnittstellen, die einen flexiblen Einsatz zur Laufzeit erlauben.

IPv6 als integrierende Netzwerkschicht

Die Grundidee von IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN, sprich six-louhpähn) besteht darin, das IPv6-Protokoll auch für lokale Funknetze zu nutzen, um hierbei homogene und durchgängige Ende-zu-Ende-adressierte Netzwerke zu ermöglichen. Auf diese Weise kann das Internet der Dinge eine Verschaltung mehrerer IP-basierter Netze unter Nutzung individueller IPv6-Adressen werden, sodass zur Kopplung nur Layer-3-Router, aber keine Kopplungselemente höherer Schichten, insbesondere keine Gateways, benötigt werden. Darüber hinaus wird die Standardisierung in den öffentlicheren Raum der Internet Engineering Task Force (IETF) gehoben.

Auf der Ebene der Funknetze soll vor allem der IEEE802.15.4-Standard zum Einsatz kommen. Allerdings sind auch Portierungen auf andere Funkprotokolle in Aussicht gestellt. Implementierungen liegen u. a. für Bluetooth, WiSun, LoRaWAN und ULE vor.

Auf diese Weise besteht die Möglichkeit, dass sich IPv6 als medienunabhängige Netzwerkschicht auch für die Vielzahl von Funksystemen durchsetzt und das Internet der Dinge dementsprechend zukünftig vollständig auf dem Internetprotokoll basiert („IP everywhere"). Eine mögliche zukünftige Installation ist in Abbildung 1 gezeigt.

Die Vorgaben des IETF sind seit nunmehr etwa vier Jahren stabil. Sie sind auch die Grundlage für mehrere Implementierungen, deren Interoperabilität in einem ersten Plugtest verifiziert wurde.

Marktübersicht

Bevor die neue Implementierung vorgestellt wird, soll kurz erläutert werden, warum die Autoren der Meinung sind, dass eine weitere quelloffene Implementierung sinnvoll ist. Denn schließlich gibt es bereits verschiedene kommerzielle und auch quelloffene Implementierungen für 6LoWPAN, wobei im Rahmen dieses Beitrags vor allem die quelloffenen Varianten betrachtet werden sollen. Allerdings sind diese Implementierungen aber aus Sicht der Autoren alle mit gewissen Nachteilen behaftet.

Verbreitete Implementierungen wie µIPv6 (Contiki), BLIP, RIOT sind sehr stark an das zugrunde liegende Betriebssystem bzw. an betriebssystemähnliche Strukturen gebunden und damit nur beschränkt in andere Systemumgebungen portierbar. Hierbei er-

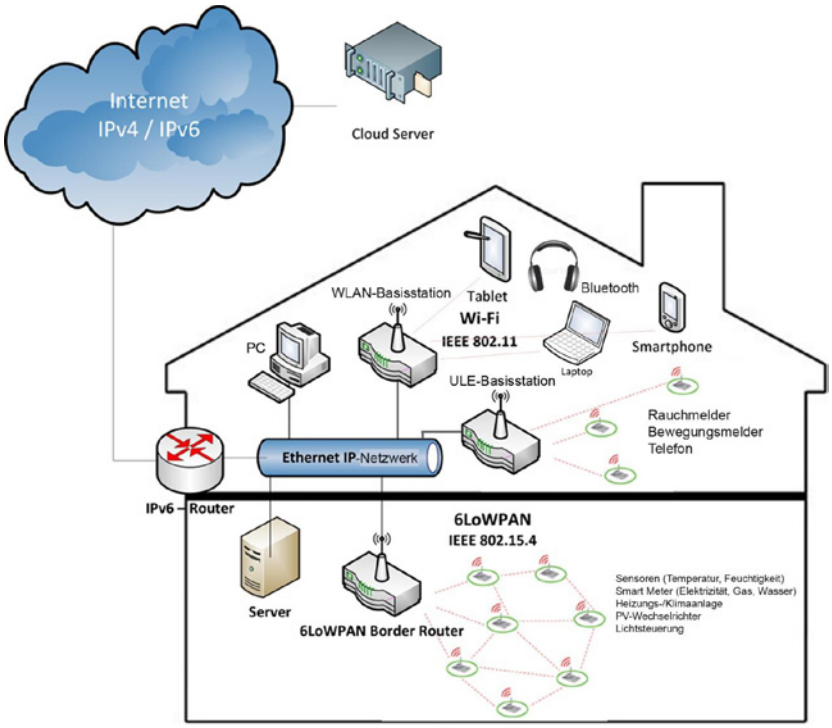

Abb. 1: Eine All-IP-Architektur ermöglicht eine einfachere Netzwerkintegration durch vereinheitlichte Adressschemata und die Kopplung nur auf Routerebene.

füllen diese Betriebssysteme oft nicht die Anforderungen an professionelle Systeme und finden außerhalb der Kommunikationsanwendungen auch praktisch keine Verbreitung.

Hinzu kommt gerade bei µIPv6 die Verwendung von sehr betriebssystemspezifischen Konstrukten (protothreads), die eine Portierung erschweren. BLIP 2.0 ist gar in dem C-Derivat NesC programmiert und somit unmittelbar gar nicht übernehmbar.

Darüber hinaus versuchen viele der aktuellen Aktivitäten in diesem Umfeld mit viel Aufwand komplette Systemumgebungen mit dezidierten Simulatoren und weiteren Werkzeugen aufzusetzen. Aus Sicht der Autoren ist es viel effizienter, hier auf bestehende Werkzeuge zurückzugreifen und lediglich die Einbindung der eigenen Implementierung stabil und einfach vorzubereiten. Auf diese Weise ist dann auch die Integration in Gesamtsysteme viel einfacher möglich.

Projekthintergrund

Aus diesen Gründen fiel die Entscheidung, eine eigene Implementierung vorzulegen, die folgende Anforderungen erfüllt:

- reine Implementierung in nativem ANSI-C, sodass die Ausführung komplett ohne Betriebssystem oder auch als ein Task in einem Betriebssystem erfolgen kann;
- sehr einfache Portierbarkeit auf unterschiedliche Mikrocontroller- und Transceiver-Plattformen durch konsequenten Einsatz von Board Support Packages (BSP) und Hardware Abstraction Layern (HAL);
- modularer Aufbau, der auch für die Portierung auf andere Funkprotokolle vorbereitet ist;
- modularer Aufbau durch Herauslösen der Basisfunktion in generische Utility-Module;
- effiziente, aber gleichermaßen stabile und gut testbare Implementierung durch den Einsatz eventgetriebener Programmierung;
- statische Speicherverwaltung mit skalierbarem Buffer-Management;
- einfache Anpassbarkeit und Parametrierbarkeit durch kombinierten Einsatz von Compile- und Run-Time-Optionen. Hierbei werden die Compile-Time-Optionen mithilfe des SCons Building-Systems verwaltet;
- Bereitstellung einer an den klassischen BSD-Socket angelehnten Socket-Schnittstelle;
- Vorbereitung für die Integration eines sicheren Transport Layers, vor allem für den ebenfalls im ivESK entwickelten (D) TLS-Stack.

In einem vom Bundesministerium für Wirtschaft und Energie (BMWi) geförderten Projekt wurden Implementierung und Verifikationen des sogenannten emb::6-Stacks durchgeführt. Hierbei diente die Implementierung des µIPv6-Stacks aus dem Contiki-Projekt als Grundlage. Dieser wurde in Bezug auf die internen Abläufe und die Struktur der Funktionsschnittstellen (APIs) so umgebaut, dass die oben erwähnten Anforderungen erfüllt sind.

Zusätzlich wird mit einem teilautomatisierten Verfahren sichergestellt, dass weitere Updates und Bugfixes der Community für die Ursprungssoftware in den neuen Protokollstapel übernommen werden. In Absprache mit den Autoren des Contiki-Projektes wurde die nunmehr erstellte Implementierung als quelloffenes Projekt mit einer identischen, sehr einfachen Lizenzregelung bereitgestellt.

Mittlerweile liegen Portierungen auf verschiedenen Hardware-Plattformen vor. Hierzu zählen gegenwärtig Mikrocontroller wie Atmel AVR8, TI MSP430 sowie zahlreiche ARM-basierte MCUs mit Cortex-M0+- und M3-Architekturen. Als Transceiver werden gegenwärtig at86rf2128(b) und at86rf230 von Atmel und TI's cc1200 unterstützt.

Architektur

Abbildung 2 zeigt den Aufbau der emb::6-Protokollimplementierung. Im Zentrum der Entwicklung steht natürlich die Netzwerkschicht, die nach oben für die Anwendungsschichten die Schnittstellen für die Kommunikation anbietet und nach unten die Dienste der Sicherungs- und der physischen Schicht nutzt. Die einzelnen Schichten werden in den folgenden Abschnitten detaillierter erläutert.

Neben der Netzwerkfunktionalität sind alle wichtigen Basisfunktionen, wie Timer, Buffer-Management und Speicherverwaltung herausgelöst und in generische Utility-Module zusammengefasst.

Zur flexiblen Unterstützung unterschiedlicher Mikrocontroller und Transceiver auf unterschiedlichen Boards wurden alle hardwarespezifischen Bestandteile in einem Board Support Package (BSP) zusammengefasst, das letztendlich den Hardware Abstraction Layer (HAL) bedient.

- *Anwendungsschicht*: Die Anwendungsschicht (Application Layer, APL) ist ein optionaler Bestandteil des emb::6-Stacks. Hierbei kann in Abhängigkeit von der angestrebten Funktionalität zwischen verschiedenen Implementierungen ausgewählt werden, wobei bislang nur CoAP im Open-Source-Bereich bereitgestellt wird:

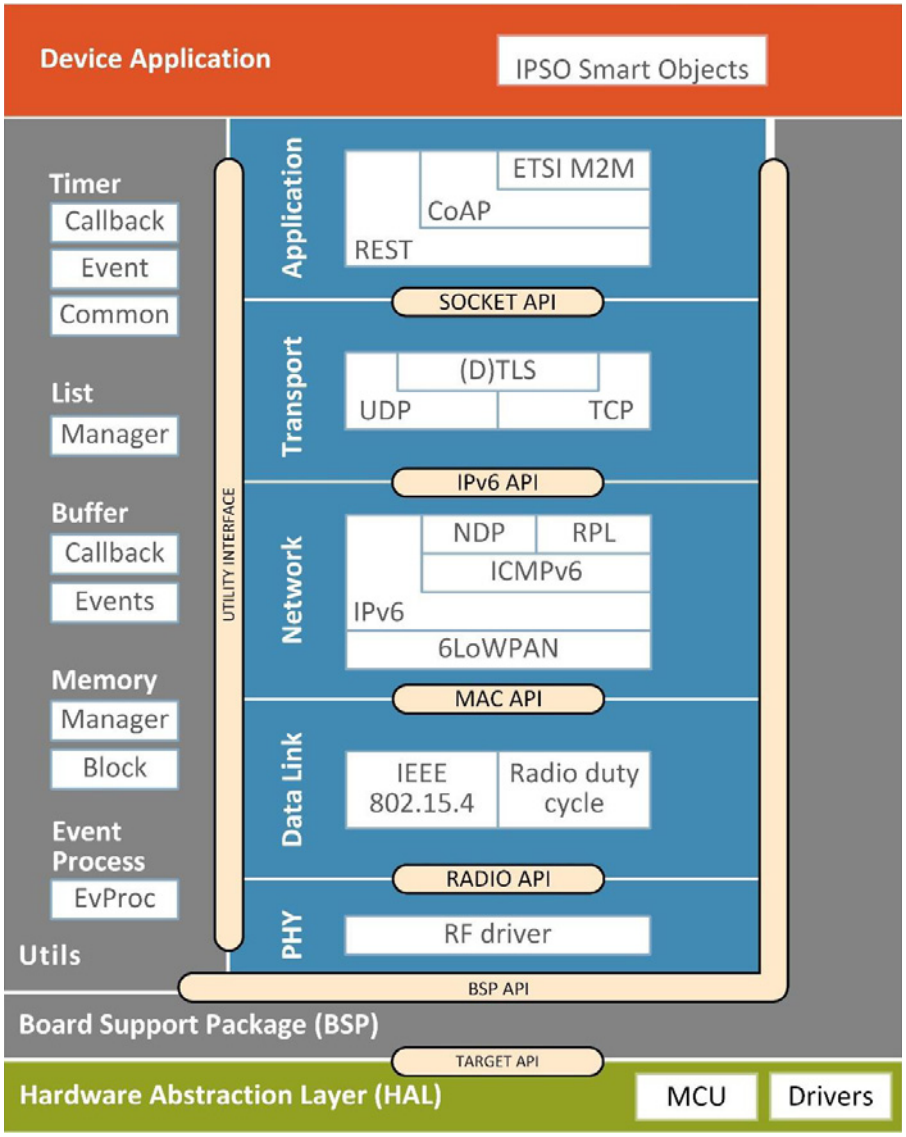

Abb. 2: Der emb::6-Protokollstapel weist einen strukturierten Aufbau auf, der leicht erweitert werden kann.

- CoAP. Das CoAP-Protokoll ist in Bezug auf das Request-Response-Paradigma an http angelehnt. Im Unterschied zu http wird bei CoAP aber ein codierter Methodenaufruf auf einen Uniform Resource Identifier (URI) verwendet, der eine direkte Abfrage und Zuordnung im Server zulässt, ohne dass – wie bei http – aufwendig geparst werden muss. Die CoAP-Spezifikation wurde von der Constrained RESTful environments (CoRE) Working Group erarbeitet. Wie der Name dieser Gruppe schon signalisiert, war ein Kerngedanke die Organisation durch RESTful Services, also durch Dienste, die der Representational State Transfer (REST)-Abstraktion folgen.Darüber hinaus werden auch Multicast-Dienste unterstützt, was für viele reale Anwendungen praktische Vorteile bringt. Auch werden grundsätzliche Möglichkeiten zur Ressourcenerkennung (Resource Discovery) und zur Beobachtung (Observing Resources) von Ressourcen unterstützt.
- ETSI M2M. Das European Telecommunications Standards Institute (ETSI) hat mit großem Aufwand eine horizontale Diensteplattform definiert, die die Interoperabilität verbessern und die M2M-Marktfragmentierung verringern soll. Insbesondere wird ein Service Capability Layer (SCL) definiert, der über offene RESTful-orientierte Schnittstellen zugänglich ist.
- LWM2M. LightweightM2M wurde von der Open Mobile Alliance (OMA) als Protokoll zum Gerätemanagement spezifiziert, kann aber auch zum Austausch von Dienste- und Anwendungsdaten genutzt werden.
- Parallel zum LWM2M-Stack, der die Managementebene beschreibt, hat die IPSO-Alliance die Version 1.0 ihrer Objektbeschreibungen veröffentlicht. Sowohl LWM2M als auch IPSO verwenden dieselbe Funktionsstruktur und beide Definitionen können direkt auf CoAP aufgesetzt werden.

- *Transportschicht*: Die Transportschicht (transport layer) stellt eine an den klassischen BSD-Socket angelehnte Socket-Schnittstelle zur Verfügung. Standardmäßig wird – wie in vielen Anwendungen üblich – das verbindungslose User Datagram Protocol (UDP) unterstützt. Eine Integration von TCP ist möglich.

- *Netzwerkschicht*: Die Netzwerkschicht (network layer) ist der bisherige Kern der Aktivitäten. Er enthält folgende Teilmodule:
  - Das obere Modul realisiert das standardmäßige IPv6-Protokoll.
  - Dieses wird dann mithilfe der unteren 6LoWPAN-Teilschicht an die Anforderungen des IEEE802.15.4 adaptiert. IPv6 erfordert eine maximale Rahmengröße (Maximum Transmission Unit, MTU) von mindestens 1280 Bytes, während beispielsweise IEEE802.15.4 nur maximal 127 Bytes erlaubt. Die geringere Rahmengröße des IEEE802.15.4 Standards erfordert zudem eine effiziente Einbettung, weil der fixe Teil des Standard-IPv6-Headers bereits 40 Byte beträgt. Würden diese in jedem Rahmen zusammen mit den bis zu 25 Bytes MAC-Overhead des IEEE802.15.4-Standards übertragen, wären mit bis zu 65 Bytes schon mehr als die Hälfte des Rahmens mit Steuerinformationen gefüllt. Entsprechend erscheint sowohl eine Kompression des Headers als auch eine Fragmentierung sinnvoll und notwendig. Diese Anpassungen werden zusammen mit einigen anderen Funktionen als 6LoWPAN in den RFCs 4944, 6282 und 6775 beschrieben. Hierbei werden dann IPv6 und UDP Header komprimiert.

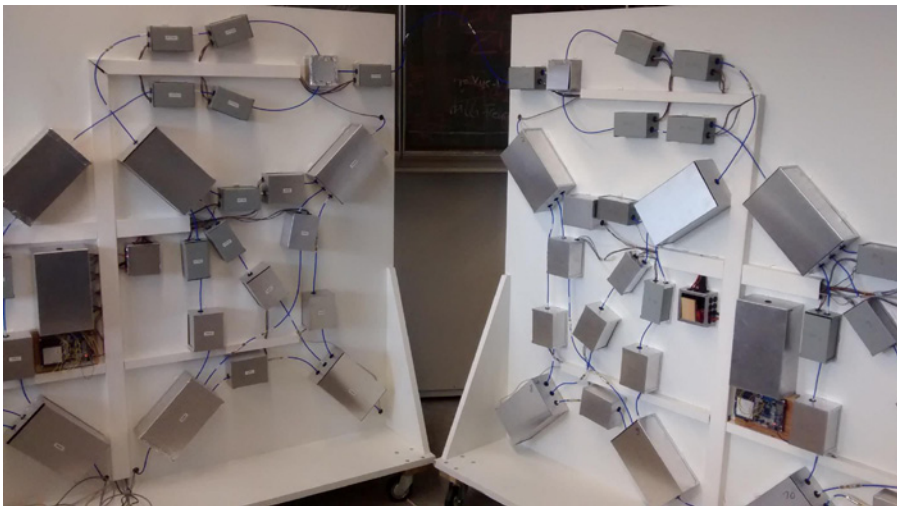

Abb. 3: Das Automated Physical Testbed (APTB) erlaubt den Test räumlich verteilter Hardware über emulierte Funkverbindungen.

Die IP-Pakete mit einer maximale Rahmengröße (Maximum Transmission Unit, MTU) von 1280 Bytes werden fragmentiert und in die maximal 127 Byte großen IEEE 802.15.4-Rahmen verpackt.

- Das obere IPv6-Modul wird ergänzt durch das Routing-Protokoll, das die Verkehrsweiterleitung in vermaschten Topologien übernimmt. Zum Einsatz kommt hierbei das in 2012 spezifizierten „Routing Protocol for Low power and Lossy Networks“ (RPL, sprich: rippl), welches von praktisch allen wichtigen Marktteilnehmern anerkannt und verwendet wird. Das Routing-Protokoll selbst ist im RFC 6550 beschrieben. Weitere wichtige Bestandteile sind der Trickle-Algorithmus (RFC 6206) für das Timing der Informationspakete, die Definition der Routing-Metriken in RFC 6551 sowie die Objective Function Zero (RFC 6552), die für die Wahl des bevorzugten Elternknotens benötigt wird.
- Hinzu kommen noch die Managementprotokolle Internet Control Message Protocol (ICMPv6) und das Neighbor Discovery Protocol (NDP).

- *Sicherungsschicht*: Hier wird im Wesentlichen die Funktion des IEEE802.15.4-MAC realisiert. Für weiterleitende Knoten (Routing Nodes) wird die Funktion des Full Function Devices (FFD) benötigt, während für Endknoten, sogenannte Leaf Nodes, bzw. Non-Routing Nodes die einfacheren Reduced Function Devices (RFD) ausreichen. Die Kopplung mit der physischen Schicht ist recht eng, weil viele Transceiver bereits Funktionen für den Kanalzugriff, wie z. B. CSMA oder Auto Retransmission, in Hardware anbieten. Hervorzuheben sei in diesem Zusammenhang, dass die IEEE802.15.4-Funktionalität des emb::6-Stacks um sogenannte Wake-On-Radio (WOR) -Funktionen erweitert wurde, die ein Aufwecken der Transceiver über die Funkschnittstelle erlaubt. Allerdings wird diese Funktion zum gegenwärtigen Zeitpunkt nur von Transceivern effizient unterstützt, die nicht kompatibel zu den Vorgaben des IEEE802.15.4 sind. Über diese Ansätze wird in einem weiteren Beitrag berichtet werden.

- *Physische Schicht*: Aufgrund der Hardwareunterstützung vieler Trans-ceiver für MAC-Funktionen besteht die unterste Schicht des Protokollstapels im Wesentlichen aus der Ansteuerung der Transceiver-Funktionen. Diese werden im sogenannten Radio Driver zusammengefasst.

Test & Verifikation

Test und Verifikation von Implementierungen für verteilte Netzwerkfunktionalität stellen immer wieder eine Herausforderung dar. Deswegen wurden verschiedene Bestandteile entwickelt, um diese Aufgaben in einem mehrschrittigen Verfahren zu unterstützen:

- *Simulation*: Es liegt – insbesondere für die Einbindung des MAC-Layers – eine Integration in den Netzwerksimulator NS-3 vor.
- *Virtualisierung*: Der HAL wurde so angepasst, dass der Protokollstapel auch in einer virtualisierten Umgebung ausgeführt werden kann. Die Knoten werden dabei in getrennten Prozessen ausgeführt, die über eine Interprocess Communication (IPC) gekoppelt werden. Eine Ankopplung an reale Hardware ist hierbei möglich.
- *Emulation*: Mit dem Automated Physical Testbed (APTB) steht am ivESK ein Aufbau zur Verfügung, mit dem reale Hardwareknoten über emulierte Funkkanäle verbunden werden können (vgl. Abbildung 3).

Darüber hinaus entsteht gegenwärtig eine Testumgebung auf der Grundlage der „Testing and Test Control Notation“ (TTCN-3).

Ausblick & offene Themen

Das Internet der Dinge ist unaufhaltsam auf dem Vormarsch. Die Community hat hierbei zu großen Teilen verstanden, dass ein wirklicher, nachhaltiger Markterfolg sich nur durch offene, übergreifende und interoperable Lösungen und durch die Bereitstellung von vorentwickelten, getesteten und einfach zu nutzenden Plattformen erreichen lässt.

Der in diesem Beitrag vorgestellte emb::6-Protokollstapel ist eine solche Plattform. Wir laden alle Interessierten ein, an dem unter https://github.com/hso-esk/emb6 verfügbaren Projekt mitzuarbeiten, um diese herstellerunabhängige, leistungsfähige, aber portabel und flexibel einsetzbare Lösung weiter entwickeln und nutzen zu können.

Zukünftige Entwicklungsthemen sind sicherlich die Erstellung weiterer Portierungen sowie zusätzlicher Applikation Layer. Darüber hinaus steht auch die Integration mit dem emb::TLS-Protokollstapel zur Gewährleistung von Ende-zu-Ende-Sicherheitslösungen auf der Tagesordnung. ■