Constrained RESTful Environments: M2M-Kommunikation auf Anwendungsebene

Patrick Bilic
Betreuer: Christoph Söllner
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: patrick.bilic@in.tum.de

KURZFASSUNG

Webservices ermöglichen die automatische Interoperabilität zwischen heterogenen Endgeräten im Internet. Diese auf Anwendungsschicht angesiedelten Services sind Wegbereiter in der Machine-to-Machine (M2M) Kommunikation unter dem Leitbild des Internets der Dinge (IoT). Die zunehmende Vernetzung sowie Miniaturisierung von Rechnern in Haushalten und Industrie, bestätigen dieses Leitbild [3, p. 2794]. Eine große Herausforderung dabei ist, die Erweiterung von Web-Architekturn auf die Domäne der ressourcebeschränkter Geräte und deren Umgebungen.

Diese Arbeit widmet sich den Inhalten der Constrained RESTful Environments Arbeitsgruppe (CoRE). Als Teil der Internet Engineering Task Force (IETF), hat sie es sich zur Aufgabe gemacht, eine Representational State Transfer (REST)-Architektur für ressourcebeschränkte Geräte, bzw. Knoten zu definieren. Im Folgendem wird dargestellt, wie CoRE den gegebenen Eigenschaften und Merkmalen der Zielplattformen begegnet und deren Lösungen umgesetzt sind.

Schlüsselworte

Constrained RESTful Environments, Constrained Application Protocol, IPv6 over Low power Wireless Personal Area Network, Machine to Machine, Internet of Things.

1. EINFÜHRUNG

Machine-to-Machine-Kommunikation (M2M) steht für den automatisierten Informationsaustausch zwischen technischen Systemen untereinander oder mit einer zentralen Stelle [2, p. 1]. Breitgefächerte Anwendungsfälle wie die Planung von Touren in der Logistik (Track & Trace), intelligente Stromnetze (Smart-Grids) und Stromzähler (Smart-Metering) sowie Car-to-X-Kommunikation geben wichtige Impulse für M2M als Schlüsseltechnologie der Zukunft [2, p. 8].

Laut dem National Intelligence Council (NIC), sollen bis 2025 Kommunikationsknoten in herkömmlichen Gegenständen wie Verpackungen, Möbelstücke und Papierdokumente in Form von eingebetteten Systemen zu finden sein [3, p. 1].

Dieser Trend ist bekannt als Internet of Things (IoT). Analysten rechnen damit, dass in Zukunft Billionen von eingebetteten Geräten mit dem Internet verbunden sein werden. [30, p. 1] Solche Systeme werden als Smart Objects bezeichnet. Diese Technologien ermöglichen es Alltagsgegenständen ihre Umwelt zu verstehen und darauf Einfluss zu nehmen. [17, p. 1]

Die zunehmende Integration eingebetteter Systeme in herkömmlichen Haushaltsgeräten sowie in der Industrie wird in den In den nächsten Jahren dazu beitragen, die Internetlandschaft zu verändern. Low-Power/Low-Cost-Recheneinheiten wie beispielsweise die 8-Bit Microcontroller Reihe ATtiny der Firma Atmel, ermöglichen unter einer Betriebsspannung bis maximal 5.5 Volt, Taktfrequenzen von bis zu 12 MHz und besitzen Flashspeicherkapazitäten von 512 Byte - 16 KiByte. Endkunden können Einzelstücke für 1 - 2 Euro erwerben. (Stand 2. Q. 2013) Neben den Eigenschaften wie Low-Power/Low-Cost-Recheneinheiten, angepassten Betriebssystemen, wie beispielsweise TinyOS, zählt die drahtlose Vernetzung und ein optimiert angepasstes Informationsmanagement solcher Systeme zu den Schlüsselfaktoren angehender ressourcebeschränkter M2M-Plattformen [12, p. 4-6].

Dazu, muss neben der Vernetzung solcher Rechensysteme, der Zugriff auf Informationen auf Applikationsebene geregelt sein. Das Hypertext Transfer Protokoll (HTTP) dient der Übertragung von Daten über Netzwerke. HTTP stellt (neben weiteren) Request-Methoden zur Verfügung um Ressourcen im Internet, die eindeutig anhand ihres Uniform Resource Identifier (URI) identifiziert werden können, anzufordern (GET), anzulegen (POST), zu aktualisieren (PUT), zu löschen (DELETE) oder deren Metadaten abzufragen (HEAD, OPTION) [14, p. 48]. Mit dem Webservice Architekturstil, Representational State Transfer (REST) lassen sich Verteilte Anwendungen mit Hilfe von Request-Methoden realisieren. Dabei definiert REST, welche Auswirkungen die Request-Methoden auf eine Ressource haben [27, p. 97-100].

Die Arbeitsgruppe für Constrained RESTful Environments (CoRE) der Internet Engineering Taskforce (IETF) hat es sich zur Aufgabe gemacht, den REST-Architekturstil Geräten und Netzwerken zur Verfügung zu stellen, die aufgrund ihrer ressourcebeschränkten Eigenschaften für HTTP nicht geeignet sind.

Sektion 2 beschreibt CoRE und geht dabei auf die Zielplattformen sowie Anforderungen ein. Anschließend wird der CoRE-Protokollstapel vorstellt. Daraufhin wird die CoRE-

doi: 10.2313/NET-2013-08-1_17

Architektur analysiert und wichtige Bestandteile erläutert. Desweiteren werden Sicherheitsmechanismen von CoRE dargestellt. Sektion 3 stellt einen Vergleich zu einem SOA-orientierten Ansatz dar.

2. CORE

Im folgenden Abschnitt wird CoRE ausführlich vorgestellt. Dabei werden zunächst die Eigenschaften der Zielplattformen beschrieben. Technische Rahmenbedingungen werden anhand des ISO-OSI Referenzmodells erläutert. Aufbauend auf diesem Wissen wird die CoRE Architektur vorgestellt und sicherheitsrelevante Aspekte geschildert.

2.1 Zielplattformen und Anforderungen

Dieser Abschnitt klärt welche Plattformen das CoRE-Framework adressiert. Außerdem werden Anforderungen dieser Zielplattformen belichtet um anschließend zu klären ob das CoRE-Framework dem gerecht wird.

2.1.1 Zielplattformen von CoRE

Ressourcenbeschränkte Geräte gelten als Zielplattform des CoRE-Frameworks. Dieser Abschnitt zeigt die Eigenschaften solcher eingebetteten Systeme. Unter der Berücksichtigung dieser Zielplattformeigenschaften soll CoRE folgenden Anforderungen gerecht werden

Wichtige Merkmale [19, p. 2-3] [35, p. 1]:

- Geringe Datenübertragungsraten (max. 127 Bytes/s)
- Geringe Bandbreiten (max. 250 kbps im 2.4 GHz Band)
- Geringe Rechenleistung (bis zu 12 MHz)
- Unterstützung von vermaschten, Stern- und Baumtopologie
- Batteriebetrieb der Knoten (2 AA Batterien für die Dauer von 2 Jahren)
- Ad hoc Netzwerkumgebungen
- Hohe Anzahl von Knoten
- Nicht umgebungsgebunde Ad-hoc Netzwerke

Um Voraussetzungen für moderne M2M-Anwendungen zu schaffen, sieht sich CoRE mit folgenden Anforderungen bezüglich der Zielplattformeigenschaften konfrontiert.

Wichtige Anforderungen [18, p. 1, 3-4]:

- Identifizierbarkeit der Knoten über das Internet
- Nahtlose Integration in bestehende Infrastrukturen
- Optimierung bestehender Anwendungsprotokolle
- Optimierte Codierung der Nutzdaten
- Geringer Konfigurationsaufwand

2.2 IETF Protokoll Stapel für CoRE

Dieser Abschnitt zeigt die wichtigsten Eigenschaften des CoRE Protokollstapels. Angelehnt am Open System Interconnection (OSI) 7-Schichten Referenzmodell der Internationalen Organisation für Standardisierung (ISO) sowie des TCP/IP-Stacks (siehe 1) wird der einzelne Schichten bishin zur Anwendungsebene sowie dessen Nutzdaten beschrieben.

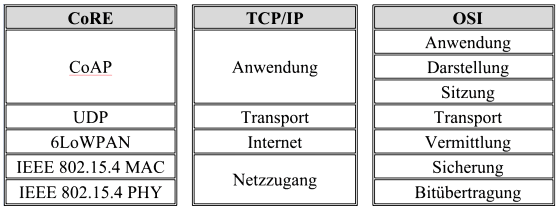

Abbildung 1: Einordnung des CoRE Protokollstapels im ISO-OSI sowie TCP/IP Modell. (Eigene Darstellung)

2.2.1 Netzzugang

In Hinblick auf die Vernetzung von Smart Objects werden ausschließlich schnurlose Netzwerke in Betracht gezogen. Der IEEE 802.15.4 Standard beschreibt die Bitübertragungsschicht und Media Access Control für Low-Rate Wireless Personal Networks (LR-WPANs). Geräten in LR-WPANs zeichnen sich durch geringe Reichweiten (bis 10m), niedrige Energieaufnahme, geringe Kosten sowie geringe Datentransferraten aus [20, p. 2]. LR-WPAN setzt auf ISM-Bänder (Industrial, Scientific and Medical Band) auf. Dabei kommen Übertragungsraten von 20 kbps bis 250 kbps, je nach Frequenzband zustande [19, p. 1-2]. IEEE 802.15.4 arbeitet auf dem selben Band wie WiFi, benötigt aber im Vergleich nur ca. 1% des Energieaufwandes. [1]

2.2.2 Netzwerkschicht

Durch die Verwendung des Internet Protokolls (IP) können Umgebungen wie CoRE in existierenden und bekannten Infrastrukturen verwendet werden. Somit ist der Aufwand, CoRE in bisherige Infrastrukturen miteinzubinden, gering [19, p. 4]. IP erfreut sich als offener Standard großer Akzeptanz und wird im Vergleich zu proprietären Technologien besser verstanden. Darüber hinaus sind keine Vermittlungsstellen wie Tranlsation Gateways nötig um ressourcebeschränkte Geräte und Teilnetze in bisherige Infrastrukturen zu integrieren [19, p. 3]. Die nahtlose Integration von eingebetteten Systemen in herkömmlichen IP-Netzwerken ist ein wichtiger Erfolgsfaktor für das IoT.

IPv6 macht es möglich, die nötige Anzahl von Geräten innerhalb des Internets auch in Zukunft zu adressieren. IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN) durch die IETF spezifiziert (RFC 4944) und definiert IPv6 in IEEE 802.15.4 basierten Netzwerken. Innerhalb des ISO/OSI-Schichtenmodells fungiert 6LoWPAN als Adaptionsschicht zwischen der Sicherungs- und Vermittlungsschicht. Diese Adaptionsschicht übernimmt drei primäre Aufgaben [24, p. 5-12]:

- Header Komprimierung: Sowohl IPv6 Header wird komprimiert, als auch der UDP Header. Speziell für die UDP Kompression hat 6LoWPAN die Ports 61616 – 61631 reserviert. Well-known Ports werden auf diese Ports abgebildet. Dabei werden bei der Übertragung im Header die ersten 12 Bits entfernt. Quell- und Zielport können somit innerhalb eines Bytes übertragen werden.

doi: 10.2313/NET-2013-08-1_17

- Fragmentierung: IPv6 Pakete werden beim Sender in mehreren Schicht 2 Frames fragmentiert. Somit können IPv6 Pakete mit der minimalen Größe von 1280 der Maximum Transfer Unit (MTU) auf IEEE 802.15.4 basierten Netzen mit der vorgesehenen Framegröße von 127 Byte übertragen werden.
- Intra-PAN Weiterleitung: Sowohl Aufgrund der geringen Reichweite als auch Umgebungseinflüsse können dazu führen, dass einzelne Geräte eines Subnetze stehts von allen Geräten innerhalb der Broadcast Domäne wahrgenommen werden können. Es wird jedoch davon ausgegangen, dass alle Geräte innerhalb eines Subnetzes ein vermaschtes Netz bilden und es somit einen stets einen Pfad vom Sender zum Empfänger innerhalb dieses Subnetzes gibt. Aus diesem Grund ermöglicht die Adaptionsschicht neben den MAC-Adressen des Senders und Empfängers weitere MAC-Adressen der Subnetzknoten zu speichern.

Neben 6LoWPAN arbeitet die IETF Arbeitsgruppe Routing over Low Power and Lossy Networks (ROLL) an dem Routing-Protokoll RPL [39, p. 1]. Das Distanzvektorprotokoll RPL berücksichtigt laut Spezifikation besondere Eigenschaften für LR-WPans; den Trickle-Algorithmus [21] zum Aktualisieren der Routingtabellen, besondere Routingmetriken und Objective Function Zero (OF0) zur Validierung der Konnektivitäten zu entsprechenden Elternknoten [37].

2.2.3 Transportschicht

Im gegensatz zum Transmission Control Protokoll (TCP) verwendet UDP keine Erkennungs- oder Korrekturmechanismen. Darüber spart der UDP Header 12 Bytes gegenüber des TCP Headers ein [23, p. 121]. Besonders für eingebettete Systeme wirkt sich die Verwendung von UDP positiv auf den Datendurchsatz aus. Darüber hinaus bietet UDP Multicastunterstützung. Im Gegensatz zum Transmission Control Protokol (TCP) besitzt das User Datagram Protokol (UDP) beide Eigenschaften. Daher ist es gut für kurzlebige Transaktionen wie sie in LR-WPANs und auftreten geeignet [20, p. 2].

2.2.4 Anwendungsschicht

CoAP ermöglicht RESTful Web Services für ressourcebeschränkte Geräte und Netzwerke. Da auf der Transportschicht UDP verwendet wird, bietet CoAP Mechanismen zur Congestion Control [20, p. 2].

Die Qualität der Netzwerkverbindungen innerhalb LR-WPANs nicht konstant sind. HTTP kann auch in 6LoWPAN verwendet werden. Im Vergleich zu CoAP doppelter Energieverbrauch und zehnmal mehr Daten die übertragen werden ist jedoch das Ergebnis [8, 3]. Dies ist darauf zurückzuführen, dass CoAP einen kompakten Header mit einer fester Länge von 4 Bytes besitzt. Insgesamt hat ein typischer Request eine Größe von 10 - 20 Bytes [8, 3]. Nach weiterer Kapselung durch UDP, 6LoWPAN und MAC besitzt ein Frame eine Größe von maximal 127 Bytes [9, p. 3].

Das Interaktionsmodell von CoAP ist ähnlich dem HTTP Client-Server Modells. Mit dem Unterschied, dass eine Implementierung von CoAP beide stets Rollen unterstützt [32, p. 8]. CoAP besitzt zwei Schichten. Die Transaktionsschicht ist verantwortlich für den Informationsaustausch zwischen zwei Endpunkte. Request/Response Schicht ist verantwortlich für den Austausch von Requests und Responses [32, p. 9]. Desweiteren werden Methoden zur Überlastkontrolle, wie Default Timeout und exponentielles Back-off zwischen Retransmissions bis der Empfänger Bestätigungsnachrichten (ACK) sendet, unterstützt [32, p. 10]. Diese Mechanismen ermöglichen asynchrone Kommunikation, eine essentielle Anforderung von IoT und M2M Anwendungen [9, p. 2-3].

2.2.5 Nutzdaten

Abhängig von der REST basierten M2M-Applikation werden verschiedene Nutzdatenformate unterstützt. Für Ressoursenbeschränkte Geräte eigenen sich jedoch nicht alle. Extensible Markup Language (XML) ist weit verbreitet jedoch wegen der Datenrepräsentation und dem Fakt, dass das Parsen dieser Dokumente Rechenleistung benötigt, nicht geeignet [41, p. 2]. Java Script Object Notation (JSON) dagegen bietet eine schlankere Datenrepräsentation [41, p. 3]. Im Vergleich bietet XML jedoch mehr Flexibilität [41, p. 2-3].

Verschiedene Mechanismen zur Komprimierung von Beschreibungssprachen wie XML wurden verglichen und das Efficient XML Interchange (EXI) Format lieferte vielversprechende Ergebnisse. Forschungen zeigen, dass die Kompression durch EXI im Gegensatz zu reiner XML eine bis 50-fach kleinere Nutzdatengröße erreichen kann [7, p. 5]. Bereits jetzt lassen sich Implementierungen zu CoAP zusammen mit dem EXI Format im Betriebsystem TinyOS wiederfinden [7, p. 6].

CoRE Architektur Das CoRE-Framework liefert Werkzeuge, um den Anforderungen zu begegnen. Diese Werkzeuge bauen auf dem CoRE-Protokollstapel auf und bilden einen wichtigen Bestandteil der Architektur von CoRE. Der zugrunde liegende Architekturstil REST, stellt Kriterien zum Design ressourcenorientierter Architekturen zur Verfügung [27, p. 79-80].

RESTful Web Services erfüllen insgesamt folgende Kriterien [4, p. 9-10]:

- Eindeutige Identifikation jeder Ressource durch URI
- Pull-basierter Nachrichtenaustausch, nach Request-Response Muster
- Zugriff/Manipulation von Ressourcen per GET-, PUT-, PATCH-, POST- und DELETE-Methode
- Abfrage von Metadaten bezüglich Ressourcen per HEAD- und OPTIONS-Methode
- Zustandslosigkeit der Client-Server Verbindung erfordert selbstbeschreibende Nachrichten
- Flexible Präsentation der Ressource (z.B. XML, JSON, PDF)
- Repräsentationen von Ressource können auf weitere Ressource verweisen
- Reduzierung des Datenverkehrs durch Caching (optional)

CoRE ermöglicht die Integration einer RESTful Architektur für ressourcebeschränkte Knoten (Smart Objects) und deren Netzwerke (6LoWPAN) in bestehende Infrastruktu-

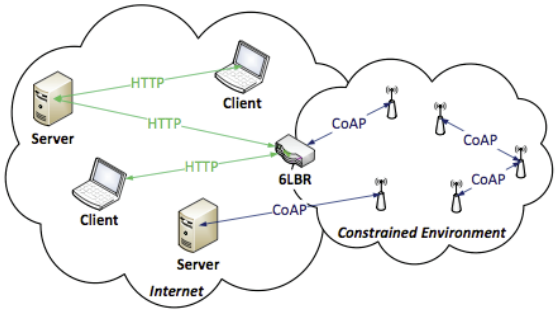

Abbildung 2: Überblick CoRE-Architektur [6, p. 1].

ren. Grafik 2 zeigt wie solche Architekturen in bestehende Infrastrukturen eingebunden werden.

Knoten (Smart Objects) innerhalb von CoRE werden als Host bezeichnet. Hosts verfügen über einen eingebetteten Webserver. Ein Host kann als Client oder Server einer Ende-zu-Ende-Verbindung auftreten und ist eindeutig durch seine IP identifiziert. Hosts verfügen über eine oder mehrere Ressource, auf die mit entsprechender URI zugegriffen werden kann [32, p. 27]:

”coap://”host [”:”port] path-abempty [”?”query].

Beispielsweise beim Auslesen von Sensordaten antwortet der Server mit der entsprechenden Präsentation (z.B. EXI). Auf zu bestätigende Anfragen wird zusätzlich mit einer Bestätigungsnachricht geantwortet. CoAP bietet dafür zusätzliche Mechanismen zur zuverlässigen Übertragung von Daten per UDP.

2.2.6 CoRE Link-Format

Die Verwaltung von Ressource mit eingebetten Webserver in ressourcebeschränkten Umgebungen erfordert spezielle Regeln. CoRE spezifiziert hierfür ein angepasstes HTTP Link Format (RFC 5988). CoRE verwendet Web Linking um herauszufinden, welche Ressource Hosts mit Hilfe deren eingebetteter Web Server anbieten [31, p. 3-4]. Dabei werden Linkinformationen innerhalb des CoAP Payloads übertragen. Das CoRE Link-Format spezifiziert drei wichtige Mechanismen.

- Resource Discovery (1) ermöglicht es Clients herauszufinden welche Ressource von Server gehostet werden. Ist die Adresse des Servers bekannt so wird eine Unicast GET-Anfrage gestellt. Mit GET ”/.well-known/core”. Der Server antwortet mit Nutzdaten im CoRE Link-Format. Abhängig des Resource Type, Interface Description und möglichen Media Type (Multipurpose Internet Mail Extensions (MIME) [11]) nimmt der Client Informationen entgegen. Um den Datenverkehr zu begrenzen, werden Suchanfragen mit Hilfe der geeigneter Filter (?query) eingegrenzt [31, p. 12-13]. Ist die Adresse des Servers nicht bekannt können Multicastanfragen verwendet werden um herauszufinden welcher Host die Ressource anbietet [31, p. 4-5].
- Resource Collections (2) dienen dazu um Links zu ähnlichen, zusammengehöriger Ressource in gebündelter Form (Kollektion) darzustellen. Beispielsweise besitzt ein Sicherheitssystem eine Liste aller angebundenen Alarmanlagen. Hosts, die solche Kollektionen verwalten stellen diese über ”/.well-known/core”bereit. Darüber hinaus werden noch Schnittstellenbeschreibungen der Ressource mit angeboten [31, p. 5].
- Resource Directories (RD) (3) werden verwendet, um Beschreibungen von Ressource anderer Server in Form von Web Links zur Verfügung zu stellen. In ressourcebeschränkten Umgebungen ist eine direkte Resource Discovery meist ineffizient: Knoten müssten ständig zur Verfügung stehen und könnten nicht in Schlafzustände wechseln um Energie zu sparen. Ohne RD ist die Anzahl an Multicastanfragen weitaus höher und würde das Datenaufkommen unnötig erhöhen [33, 3]. Deshalb registrieren Hosts ihre Ressourcen bei RDs um sie als Ablage zu verwenden.

Meist befindet sich ein RD innerhalb einer Gruppe von Hosts. Das RD besitzt eine REST-Schnittstelle für die Hosts um deren Daten innerhalb des RD verwalten zu können. Neben der Registrierung bietet die Schnittstelle Funktionen (Resource Function Set) zur Aktualisierung und zum Entfernen der Ressource an. Außerdem existiert noch eine Methode zur Validierung der im RD gespeicherten Links Ressource. Dabei Überprüft das RD dessen registrierte Ressource aktuell sind. [33, p. 9-15]

Ein mächtiges Werkzeug von RD sind Group Function Sets. Innerhalb RDs ist es möglich Gruppen von Ressource zu registrieren. Beispielsweise lassen sich Ressource von Lichtstärkenregler innerhalb eines Raumes als Gruppen zusammenfassen. Dieser Gruppe kann nun eine Multicast-Adresse zugewiesen werden, um Funktionen (Registrieren, Aktualisieren, Entfernen, Validieren) auf alle Ressource anzuwenden. CoRE stellt mit Resource Discovery, -Collections und -Directories Voraussetzungen zur Ressourcenverwaltung für moderne M2M-Anwendungen zur Verfügung. [33, p. 16-19]

Außerdem ist das Binding ein weiterer wichtiger Bestandteil des CoRE-Frameworks. Binding bezeichnet Links und Informationsaustausch zwischen Ressource verschiedener Hosts. Jeder Host besitzt eine Binding-Tabelle um Beziehungen zu fremden Ressource zu verwalten. Eine Beziehung besitzt Eigenschaften wie eine eindeutige IDs, Angaben zu Synchronisationsperioden, Ursprungs- und Zielhost der Ressourceninformation und REST-Methoden für den Austausch [34, p. 6-9].

Das Link Format und deren Attribute sind im RFC 6690 spezifiziert. Die Attribute dienen dazu, die Eigenschaften einer Ressource zu beschreiben. Das Core Link Format stellt eine Erweiterung des HTTP Link Formats (RFC5988) dar. Es wurde um drei Attribute erweitert:

- Das Resource Type ’rt’ Attribut (1) ermöglicht es, Ressourcen zusätzlich anhand anwendungsspezifischer Semantik zu beschreiben. Beispielsweise möchte man bei einem Knoten mit mehreren Temperatursensoren eindeutig zwischen Außen- und Innentemperatur unterscheiden können. Der String “Außentemperatur “er-

möglicht es, nach Ressource zu suchen, die ausschließlich die Außentemperatur liefern [31, p. 9].
- Das Maximum Size Estimate 'sz' Attribute (2) gibt Aufschluss über die Größe der Repräsentation einer Resource. Große Ressource die nicht innerhalb einer Maximum Transmission Unit (MTU) übertragen werden können, sollen gekennzeichnet werden, sodass Clients in ressourcebeschränkten Umgebungen vorab entscheiden können ob genügend Rechenkapazitäten vorhanden sind, diese zu verarbeiten [31, p. 10].
- Das Interface Description 'if' Attribut (3) ermöglicht es für einzelnen Ressourcen flexible REST Schnittstellen zu definieren. Diese beschreiben eindeutig wie mit dieser Resource zu kommunizieren ist [31, p. 10].

CoRE definiert dabei unterschiedliche Interfaces [34, p. 10-15]:

- Link List: Abfrage für Ressource eines Hosts
- Batch: Manipulation von Ressource
- Sensor: Spezielle Abfrage von Sensordaten (Repräsentation mehrerer Messungen möglich)
- Parameter: Lesen und Schreiben von einzelner Link Format Attribute
- Binding: Manipulation der Binding-Tabelle

2.2.7 Observer

Bisher wurde davon ausgegangen, dass der Abfragen ausschließlich durch Clients initiiert werden. Um den Clients stets den aktuellen Informationsstand zu gewährleisten, müssen diese GET-Operationen in bestimmten Perioden durchführen (Polling). Dieses Pull-Modell ist in Umgebungen wie CoRE und unter Berücksichtigung von möglichen Schlafzuständen von Knoten meist nicht praktikabel. Das CoRE-Framework sieht vor, Clients die Möglichkeit zu geben Änderungen zu Ressourceninformationen zu abbonieren. Dies entspricht einem Publish-Subscribe-Muster. Wenn der Server (Publisher) den Client als Subscriber akzeptiert, übermittelt der Server bei Änderungen der Ressource unmittelbar die aktuellen Informationen an den Client. CoRE bezeichnet diese Rolle, die der Client in diesem Fall einnimmt als Observer. [5, p. 4].

Die Observerfunktion ermöglicht asynchrone Datenübertragung innerhalb von CoRE-Umgebungen. [5, p. 4-5]. Das REST-Paradigma wird dabei jedoch verletzt. Zum einen würde die Observerfunktion einen Push-basierten Nachrichtenaustausch bedeutet. Außerdem speichert der Server Informationen bezüglich des Observers ab, somit kann nicht mehr von Zustandslosigkeit gesprochen werden.

2.2.8 Mirror Server

Ressourcebeschränkte Geräte wie CoRE Hosts verfügen über einen Schlafzustand, um Energie zu sparen. Dabei trennt der Host die Datenverbindung zum Netzwerk. Das herkömmliche Client-/Server-Modell ist nicht für Umgebungen wie CoRE nicht geeignet. Die Observerfunktion trägt zwar zum energieeffizienten Nachrichtenaustausch positiv bei. Server in der CoRE-Umgebung sind jedoch aufgrund ihrer beschränkten Rechenkapazitäten und limitierten Datenraten nicht imstande, beliebig viele Observer zu beliefern [38, p. 2]. Das CoRE-Framework stellt deswegen einen Mirror Server zur Verfügung. Der Mirror-Server bietet Funktionen und Schnittstellen für Hosts, von Repräsentationen von Ressource zu speichern [38, p. 4]. Im Unterschied dazu speichert RD nur Web Links und Eigenschaften bezüglich der Ressource ab. Der Mirror Server agiert als Mailbox zwischen Client und dem Server im Schlafzustand. [38, p. 4-5]

Der Mirror Server ist als zentrale Instanz innerhalb eines CoRE Netzwerks implementiert. Ein zentralisierter Ansatz bietet mehr Stabilität, da die Hosts nicht darauf angewiesen sind ständig das Ressource Discovery durchzuführen [38, p. 5-6].

2.3 Sicherheitsmechanismen in CoRE

In diesem Abschnitt werden Sicherheitsmaßnahmen innerhalb CoRE beleuchtet. Darüber hinaus werden Probleme vorgestellt, die bisher noch nicht in der Spezifikation gelöst wurden.

2.3.1 CoRE Security Bootstrapping

Deshalb präsentiert die CoRE Arbeitsgruppe einen Entwurf zur initialen, automatischen und sicheren Konfiguration (Security Bootstrapping) von Netzwerken mit ressourcebeschränkten Knoten [29, p. 1]. Allgemein beinhaltet Bootstrapping jeden Prozess der nötig ist um ein Netzwerk betriebsfähig zu machen. Dieser Prozess ist aufwendig, da vorauszusetzen ist, dass Knoten in einem Netzwerk initial keinerlei Informationen voneinander haben. Trotzdem soll gewährleistet sein, dass nur autorisierte Knoten Zugriff zum Netzwerk erhalten. Darüber hinaus besitzen ressourcebeschränkte Knoten meist keine modernen Benutzerschnittstellen. [29, p. 4]

Transport Layer Security (TLS) wird verwendet um Authentizität, Integrität und Vertraulichkeit während des Datenaustausches zwischen zwei Teilnehmern im Internet zu ermöglichen [10, 10]. Datagramm Transport Layer Security (DTLS) ermöglicht TLS in verbindungslosen Datenverkehr und findet deshalb Anwendung in CoRE. CoRE verwendet eine neue Art von TLS-Zertifikaten, die es ermöglicht, Raw Public Keys auszutauschen. Bisher ermöglicht TLS nur eine Authentifizierung der Teilnehmer via PKI (X.509 basierte Public-Key-Infrastruktur) oder OpenPGP Zertifikate [16]. Die Übertragung von Raw Public Keys, veringert das Datenaufkommen im Vergleich zu vollständigen Zertifikaten. Außerdem ist das Parsen und die Verarbeitung dieser Keys weniger rechenaufwändig. [40, p. 6]

Um das Datenverkehrsaufkommen in CoRE-Umgebungen zu reduzieren, sind die CoRE-Knoten mit der Adresse des CoAP-Servers und dessen Public Key bereits vorkonfiguriert [40, p. 6]. Der Austausch der Public Keys, wie ihn PKI versieht, entfällt somit. Außerdem müssen die ressourcenbeschränkten Knoten keine Echtzeit Uhr enthalten, um PKI Expiration Checks durchführen zu können.

Grafik 3 zeigt die Hierarchie der Bootstrapping Architektur von CoRE. Beginnend mit dem Root-Knoten, in 6LoWPAN Border Router (6LBR) mit den größten Ressourcenkapazitäten. Eine Ebene tiefer arbeiten Interior Router die RPL benutzen um miteinader und mit dem 6LBR zu Routing-Informationen auszutauschen. Auf der untersten Ebene befinden sich die Endknoten die in 6LoWPAN als Hosts bezeichnet werden [29, p. 3-4].

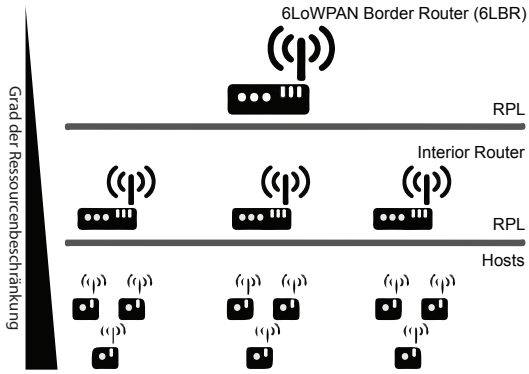

Abbildung 3: CoRE Security Architektur [6, p. 2].

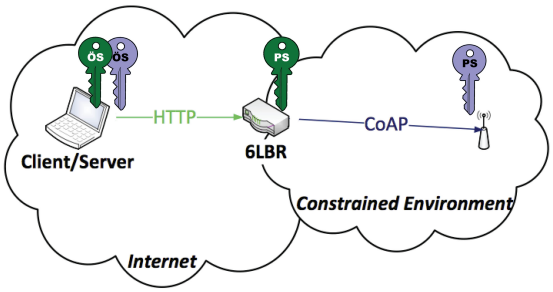

Abbildung 4: Mögliches Szenario für eine Ende-zu-Ende-Verschlüsselung mit ITLS. Client/Server verschlüsselt hierbei als Sender die Daten, mit den beiden Öffentlichen Schlüssel, um diese an einen Knoten innerhalb der CoRE Umgebung zu senden. [6, p. 2] (Angepasst durch Verfasser)

Aktuelle Konzeptentwürfe sehen einen für die CoRE-Umgebung angepassten EAP-(D)TLS Authentifizierungsprozess vor [29, p. 4-7]. Somit findet nach der automatischen Anmeldung, mittels Zertifikat, der Knoten im Netz, die Kommunikation zwischen den Knoten und des CoAP-Servers verschlüsselt statt. Neue Router werden anfangs als einfacher Host im Netzwerk aufgenommen. Sie durchlaufen den Bootstrapping-Prozess um anschließend einen Master Session Key (MSK) zu generieren. Anschließend werden Session Keys mit dessen Nachbarn durch RPL ausgehandelt um Up- und Downstream zu spezifischen Eltern- und Kindknoten aufbauen zu können [39, p. 113-118]. In jeder Hierarchieebene existieren verschiedene Mechanismen für den Bootstrapping-Prozess, die sicherstellen, dass sich Knoten auf der vorgesehenen Hierarchieebene befinden.

Wichtige Konzeptentscheidungen, ob der 6LBR selbst als Authentifizierungsserver agiert oder dieser als eine separate Instanz außerhalb des Netzes anzusiedeln ist sind noch nicht getroffen[[13, p. 5]. Im Vergleich zum ZigBee Standard fungiert der Coordinator (gleichzusetzen mit 6LBR) als Trust Center. Ein zentralisierter Ansatz erlaubt eine zentrale Verwaltung Geräten und deren Schlüssel und vereinfacht Backupmechanismen für Schlüssel eines bestimmten Netzes. Nachteile sind, dass beim Schlüsselaustausch zweier beliebiger Knoten, beide eine Verbindung zum zentralen Authentifizierungsknoten innerhalb des Netzwerks haben müssen. Außerdem repräsentiert der zentrale Knoten einen Single-Point-of-Failure in diesem Netzwerk.

Innerhalb des CoRE-Frameworks gilt es noch zwei wichtige Sicherheitsprobleme zu lösen. Erstens, CoAP/HTTP-Mapping unterstützt keine Ende-zu-Ende-Verschlüsselung (E2EE) per DTLS/TLS. Zweitens, da TLS nur geringfügig angepasst wurde um auch verbindungslosen Datenverkehr über UDP zu ermöglichen unterstützt DTLS kein Multicasting [6, 2-3]. Ersteres könnte gelöst werden durch TLS-DTLS-Tunneling oder durch Integrated Transport Layer Security (ITLS). Grafik 4 zeigt, dass mit ITLS der Sender die Pakte mit zwei Schlüssel verschlüsselt. Der 6LBR besitzt den ersten Schlüssel, entschlüsselt das Paket und reicht es an den Empfänger mit dem zweiten Schlüssel weiter. Als Nachteil ergibt sich ein größerer Overhead.

Beide Probleme können gelöst werden indem IPsec ESP (Encapsulation Security Payload) anstatt DTLS verwendet wird. Die CoRE Arbeitsgruppe empfiehlt dies jedoch nicht, da nicht alle IP Stacks unterstützt werden [6, p. 3].

2.3.2 Sicherheitsmaßnahmen im CoRE Link Format

Obwohl das CoRE Link Format als einziges Dokument der CoRE Arbeitsgruppe den RFC-Status “Standard“besitzt, sind noch einige Fragen bezüglich dessen Sicherheit offen. CoAP-Server stellen den Ressource Discovery Service Clients bezüglich deren Zugriffsberechtigungen unterschiedliche Linklisten zur Verfügung. Clients, die Ausschließlich leseberechtigt sind, erhalten keine Links die Schreibrechte voraussetzen. Für Angreifer besteht weiterhin die Möglichkeit sich URIs zu erschließen oder durch Probieren herauszufinden [31, p. 15].

Ein weiteres Problem stellt die einfache Möglichkeit von Denial of Service (DoS) Angriffen durch Multicastanfragen dar. Multicastanfragen auf well-known Link-Format-Ressource ”coap://[IPv6]/well-known/core”werden akzeptiert und innerhalb des Netzwerks an den Zielknoten weitergereicht. Erst der Zielknoten prüft das Datenpaket auf Authentizität [31, p. 16].

Ein weitere Sicherheitslücke ist innerhalb der CoRE Link Format Parser zu finden. Parser müssen zyklische Link- Descriptions erkennen. Solche können direkt (Link-zeigt auf sich selbst) oder indirekt (Referenzierte Link-Ressource auf andere Ressource Discovery Services) sein [31, p. 16].

3. DPWS IM VERGLEICH

Neben CoRE existieren derzeit noch ein weiterer bekannter Ansatz um M2M-Anwendungen basierend auf embedded Systemen wie Smart Objects zu realisieren. Dieser Abschnitt gibt Aufschluss über den SOA-basierten Ansatz von Device Profile for Web Services (DWPS) und stellt ihn CoRE gegenüber.

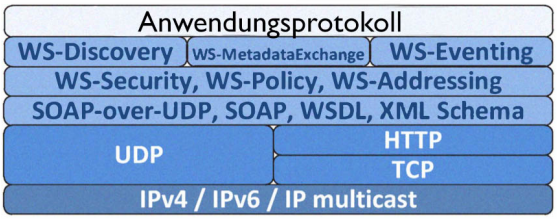

Abbildung 5: DPWS Stack [36, p. 3].

Ein zentraler Ansatz neben REST sind Service Orientierte Architekturen (SOA). Im Allgemeinen ist Service Orientierte Architekturen (SOA) ein Softwareparadigma zum Design loser gekoppelte Softwarearchitekturen [22, p. 99]. Innerhalb solcher Architekturen werden angebotene Web Services in einem Service Directory (Universal Description Discovery and Integration (UDDI)) hinterlegt. Web Service Registrieren sich bei der UDDI und definieren dabei mit Hilfe der Web Service Description Language (WDSL) ihre Eigenschaften. Clients stellen daraufhin Anfragen an das UDDI um anschließend von diesen Web Services Gebrauch zu machen. SOAP (ursprünglich Simple Object Access Protocol) ist ein aus XML-RPC entstandener Webstandard des Word Wide Web Consortiums (W3C) [15]. Es regelt den XML-basierten Austausch von Daten zwischen Geräten und kann dabei auf verschiedene Anwendungsprotokolle, wie HTTP, FTP aber auch CoAP aufsetzen [25, p. 3].

Neben SOAP existieren weitere Protokolle. Diese sind zusammengefasst als WS-* Spezifikationen. WS-Addressing stellt Mechanismen zur Adressierung von WS zur Verfügung. Es entkoppelt SOAP von niedrigeren Protokollschichten. Das Multicastprotokoll WS-Discovery ermöglicht Plug-and-Play angeschlossener neuer Geräte im Netzwerk. Darüber hinaus definiert WS-Policy Eigenschaften von Webservices und stellt Bedingungen an die Clients. Zusätzlich definiert WS-Security Regeln und Sicherheitsmaßnahmen für die Verbindung mit dem Webservice. Darüber hinaus ermöglicht WS-Eventing asynchrone Kommunikatoin zwischen Web Services [35, p. 2]. Grafik 5 zeigt den DPWS-Stack und liefert einen Überblick zu SOAP sowie WS-* Spezifikationen.

Die Datenrepräsentation durch SOAP ist jedoch aufgrund des zu Großen Overheads nicht für WSNs geeignet [25, p. 6]. Einen Vergleich zwischen SOAP und CoRE ist in der Grafik 6 dargestellt.

Devices Profile for Web Services (DPWS) beinhaltet eine Reihe bekannter Web Service Protokolle wie HTTP, XML, SOAP und WS-*. Das Ziel dabei ist, SOA-basierende Web Services direkt auf eingebetteten Geräten zu implementieren [28, p. 1].

DPWS definiert dabei vier wichtige Spezifikationen [26, p. 5]:

- Sicherer Nachrichtenaustausch zu/von Webservices
- Dynamisches finden von Webservices
- Beschreibung von Webservices

	CoRE	SOAP
Technologieart	REST als Architekturstil	Protokollframework
Sichtweise des Webs	Zugriff auf Ressourcen	Transport von Nachrichten
Kommunikation	direkt (Client-Server)	indirekt (über SOAP Intermediär), direkt (Client-Server)
Transport	UDP	UDP, TCP
Anwendungsprotokoll	CoAP	Protokoll unabhängig
Naming	konsistenter Namingmechanismus für Ressourcen	kein standardisierter Namingmechanismus vorhanden
Nutzdatenrepräsentation	XML, JSON, EXI	XML, EXI
Automatische Discovery	JA (Ressource Discovery)	JA (WS-Service Discovery)
Zugriffsrichtlinien	JA (Ressource Discovery)	JA (WS-Policy)
Zielplattform	Ressourcenbeschränkte Geräte	Herkömmliche Rechner

Abbildung 6: Vergleich CoRE und SOAP [4, p. 27].

- Webservice-Events abonnieren und empfangen

DPWS nutzt dafür SOAP und WS-* Spezifikationen [35, p. 4].

DPWS besitzt eine große, zum Teil aktive Entwicklergemeinschaft. Die Initiative Web Service for Devices (WS4D), eine Kooperation zwischen Wissenschaft und Industrie entwickelten eine Reihe von Werkzeugen für DWPS ((C/C++), WS4D-Axis2 (Java), WS4D-JMEDS (Java) and WS4D-uDPWS (C)). Ein ähnliches Projekt ist Service Oriented Architectures for Devices (SOA4D), indem unter anderem Microsoft mitarbeitete. [36, 2-3]

Vorteile gegenüber CoRE:

- Etabliertes Service Orientiertes Anwendungs Design
- Ausgereifte Zero-Configuration-Mechanismen (Plug&Play)
- Ausgereifte Sicherheitsmechanismen
- Web Services laufen direkt auf Knoten
- Große Entwicklergemeinde (Wissenschaft und Industrie)
- SOAP-over-CoAP Binding möglich

Nachteile gegenüber CoRE:

- Nicht konzipiert für ressourcenbeschränkte Geräte
- Weitaus größerer Overhead der übertragenen Daten

4. ZUSAMMENFASSUNG UND AUSBLICK

Diese Arbeit liefert einen Überblick über das von IETF erarbeitete CoRE-Framework. CoRE ermöglicht Restful Web Services in ressourcebeschränkten Umgebungen. Trends wie IoT und WoT zeigen, dass in Zukunft Smart Objects die In-

ternetlandschaft verändern werden. Sukzessive Integration moderner M2M-Anwendungen in vielfältigen Anwendungsgebieten, wie Heimautomatisierung, Smart Energy, Verpackungsindustrie, Logistik und weitere wird möglich sein.

Ressourcebeschränkte Geräte und Netzwerke, Entwickler sowie das Ideal WoT stellen Anforderungen an CoRE. Aufbauend auf dessen Protokollstapel werden diese Anforderungen unter der Berücksichtigung von Zielplattformen durch die CoRE-Architektur berücksichtigt und adressiert.

Durch die Verwendung von IEEE 802.15.4 Standards, sind Grundlagen dafür geschaffen eine hohe Anzahl sogar stark ressourcebeschränkter Geräte in bestehende Infrastrukturen zu integrieren. Durch die freie Verfügbarkeit von CoRE und dessen Anwendungsprotokoll CoAP werden Voraussetzungen für den Einsatz in heterogenen Gerätelandschaften geschaffen.

Jedoch verfügt derzeit lediglich das CoRE Link Format über eine genaue Spezifikation in RFC 6690. Neben den Verstoß gegen Konformitäten des REST Architekturstils, lassen das ungenaue CoRE-Architekturdesign sowie ungelöste Sicherheitsprobleme derzeit keinen sinnvollen Einsatz zu.

5. LITERATUR

[1] A. Ahmed, H. Shi, and Y. Shang. A survey on network protocols for wireless sensor networks. In *Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on*, pages 301–305, 2003.

[2] Arbeitsgruppe 2 M2M Initiative Deutschland. Machine-to-machine-kommunikation - eine chance fuer die deutsche industrie. In *Nationaler IT Gipfel - Essen 2012*.

[3] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[4] A. E. Ayadi. Webservices: Rest vs. soap. Master's thesis, Hamburg University of Applied Sciences, 2008.

[5] C. Bormann and Universitaet Bremen TZI. *CoRE Roadmap and Implementation Guide*. Internet Engineering Task Force, 2013.

[6] M. Brachmann, O. Garcia-Morchon, and M. Kirsche. Security for practical coap application issues and solution approaches. Master's thesis, Technische Universitottbus, 2011.

[7] A. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi. Web services for the internet of things through coap and exi. In *Communications Workshops (ICC), 2011 IEEE International Conference on*, pages 1–6, 2011.

[8] W. Colitti, K. Steenhaut, and N. De Caro. Integration wirelsess sensor networks with the web. *Vrije Universitrussel - ETRO*.

[9] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota. Evaluation of constrained application protocol for wireless sensor networks. In *Local Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, pages 1–6, 2011.

[10] M. Conrad. *Verfahren und Protokolle fuer sicheren Rechtsverkehr auf dezentralen und spontanen elektronischen Maerkten (German Edition)*. KIT Scientific Publishing, 2010.

[11] N. Freed, Innosoft, and N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, 1996. RFC 2045.

[12] Gartner Inc. Key technologies of the internet of things. 2012.

[13] J. Gilger and H. Tschofenig. Report from the 'smart object security workshop', march 23, 2012, paris, france. Technical report, Internet Engineering Task Force, 2012.

[14] D. Gourley, B. Totty, M. Sayer, A. Aggarwal, and S. Reddy. *HTTP: The Definitive Guide*. O'Reilly Media, 2009.

[15] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen, A. Karmarkar, and Y. Lafon. Soap version 1.2 part 1: Messaging framework (second edition), 2007. zuletzt zugegriffen 22.06.2013.

[16] R. Housley, SPYRUS, W. Ford, VeriSign, W. Polk, NIST, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. Internet Engineering Tastk Force, 1999. RFC 2459.

[17] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy. Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, 14(1):44–51, 2010.

[18] M. Kovatsch. Firm firmware and apps for the internet of things. In *Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications*, SESENA '11, pages 61–62, New York, NY, USA, 2011. ACM.

[19] N. Kushalnagar, G. Montenegro, C. Schumacher, Microsoft Corp, and Intel Corp. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. Internet Engeneering Task Force. RFC 4919.

[20] M. Laine. Restful web services for the internet of things. Master's thesis, Aalto University School of Science Department of Media Technology.

[21] P. Levis, Stanford University, T. Clausen, LIX Ecole Polytechnique, J. Hui, Arch Rock Corporation, O. Gnawali, J. Ko, and Johns Hopkins University. *The Trickle Algorithm*, 2011.

[22] D. Masak. *Soa?:*. Springer London, Limited, 2007.

[23] J. Matthews. *Computer Networking: Internet Protocols in Action*. Wiley, 2005.

[24] G. Montenegro, Microsoft Corporation, N. Kushalnagar, Intel Corp, J. Hui, D. Culler, and Arch Rock Corp. Transmission of ipv6 packets over ieee 802.15.4 networks, 2007. RFC 4944.

[25] G. Moritz, F. Golatowski, and D. Timmermann. A lightweight soap over coap transport binding for resource constraint networks. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 861–866, 2011.

[26] T. Nixon and A. Regnier. Devices profile for web services version 1.1, 2009. OASIS Standard.

[27] L. Richardson and S. Ruby. *Restful Web Services*. O'Reilly Media, 2007.

[28] I. Samaras, G. Hassapis, and J. Gialelis. A modified dpws protocol stack for 6lowpan-based wireless sensor

networks. *Industrial Informatics, IEEE Transactions on*, 9(1):209–217, 2013.

[29] B. Sarikaya and Huawei USA. *Security Bootstrapping Solution for Resource-Constrained Devices*. Internet Engineering Task Force, 2013.

[30] Z. Shelby. Embedded web services. *Wireless Communications, IEEE*, 17(6):52–57, 2010.

[31] Z. Shelby and Sensinode. *Constrained RESTful Environments (CoRE) Link Format*. Internet Engineering Task Force, 2012.

[32] Z. Shelby, Sensinode, K. Hartke, C. Bormann, and Universiaet Bremen TZI. *Constrained Application Protocol (CoAP)*, 2013.

[33] Z. Shelby, Sensinode, S. Krco, Ericsson, C. Bormann, and Universitaet Bremen TZI. *CoRE Resource Directory*. Internet Engineering Task Force, 2013.

[34] Z. Shelby, Sensinode, M. Vial, and Schneider-Electric. *CoRE Interfaces*. Internet Engineering Task Force, 2013.

[35] A. Sleman and R. Moeller. Integration of wireless sensor network services into other home and industrial networks; using device profile for web services (dpws). In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–5, 2008.

[36] S. Sucic, B. Bony, and L. Guise. Standards-compliant event-driven soa for semantic-enabled smart grid automation: Evaluating iec 61850 and dpws integration. In *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pages 403–408, 2012.

[37] P. Thubert and Cisco Systems. *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*, 2012. RFC 6552.

[38] M. Vial and Schneider-Electric. *CoRE Mirror Server*. Internet Engineering Task Force, 2013.

[39] T. Winter, P. Thubert, Cisco Systems, A. Brandt, Sigma Designs, J. Hui, R. Kelsey, Stanford University, Dust Networks, Cooper Power Systems, Cisco Systems, and R. Struik. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, 2012. RFC 6550.

[40] P. Wouters, Xelerance, J. Gilmore, S. Weiler, AuthenTec, and Nokia Siemens Networks. *TLS out-of-band public key validation*. Internet Engineering Task Force, 2011.

[41] D. Yazar and A. Dunkels. Efficient application integration in ip-based sensor networks. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, pages 43–48, New York, NY, USA, 2009. ACM.

doi: 10.2313/NET-2013-08-1_17