IoE

Warum IoE?
erhöhen der Lebensqualität
... aber: ubiquitäres Monitoring, sammeln & auswerten von Daten
Industrie 4.0 (Industrial IoE)
"Automation der Automation
Monitoring, Wartung

Allerdings Kommunikation von Lichtschalter & Smartphone über "Cloud", keine E2E Kommunikation
Cloud speichert Daten
Verarbeitung? Transparenz? Sicherheit? Abhängigkeit?
Möglichkeit für Hersteller, teure Geräte zu verkaufen, da die Kommunikation über die Cloud zwingend ist, und nur diese als autorisiert eingestellt werden können

Things attack the internet
DDOS Angriff durch Hack von IOE Devices

Anwendung:
Gewächshaus: Temperatursensor → Lüftung / Heizung
Lichtsensor → Jalousine
Feuchtigkeitssensor → Fenster / Bewässerung
Weinkeller
Great Duck Island } Erforschung der Tierwelt
Zebras

Drähte schwer, teuer, im Weg, obstruierend
=> Drahtlos
HW: Microcontroller
Speicher
Kommunikationsschnittstelle
Sensoren / Aktoren
Energieversorgung
} Teilweise antiproportional zueinander
=> Tradeoffs!

Meistens Funk, neben Display (falls vorhanden) größter Strombedarf

Drahtlose Sensor-Aktor-Netze
Menge von Geräten, verbunden drahtlos
Messen Daten und senden sie periodisch an eine Senke (evtl. mit Internet-Zugang)

Annahmen:
Energieversorgung durch Batterie
Keine Unterstützende Infrastruktur (Katastrophengebiete, Militär)
Aufbau unterstützender " evtl. zu teuer
WSAN soll unauffällig integriert werden
Hohe Stückzahl erwünscht
Geringe Sendereichweite → Multi-Hop-Kommunikation
Dezentral, selbstorganisierend, limitiert in Ressourcen, störanfälliger Kommunikationskanal, unsicher (security)
Infrastruktur ist nicht ständig erreichbar, kann nicht vor Ort gewartet werden, schwer zugänglich
Rechenleistung, Energie- und Datenspeicher stark limitiert
ebenso Kommunikationsmöglichkeit
Fehlerbehaftetetes Kommunikationsmedium
Knoten können ausfallen, zerstört werden oder im Nachhinein hinzugefügt. Krypto zu "teuer", Kanal abhörbar

Eigene Netze für IoT?
→ Sigfox / Swiss Com Low Power Network
Oder nicht?
→ Narrow Band B IoT
Telekom Smart Parking

Hinter einem Gateway verbirgt sich das Internet
IoT ↔ IoT
↳ Backend-Server, Datenanalyse, Monitoring

Gerätekassen & Anwendungen

Kriterium: Leistungsfähigkeit

C0	Data size: << 10KB	Code Size << 100 kB	unsichere Kommunikation
C1	~ 10KB	~ 100 KB	opt. IP & UDP, kein TLS
C2	~ 50KB	~ 250 KB	opt. Netzwerkstacks

Kriterium: Energiebeschränkung

E0	Beschränkung auf ~~Ereignisses~~ Ereignisses	Energy Harvesting
E1	" auf Zeitperiode	Batterie austauschen / aufladen
E2	Lebenszeitbeschränkung	Batterie untauschbar
E9	keine Beschränkung	Steckdose

Tablets
Smartphones
SBC
Schwarmroboter
Smart Home
Smartmetering
Sensornetze
Submetering
Smart Dust
Nanonetze
Hardware
— Größe, Gewicht →
← Spezialisierungsgrad, Knotendichte —

Nanonetze
C0, E0
Anwendung: Militär, Biomedizin, Industrie, Chemie
1 Node kann nur eine Aufgabe übernehmen: Speichern, Rechnen, Messen, Manipulieren
Molekulare Kommunikation statt elektromagnetische
Walkway, Flow, Diffusion

Smart Dust
C0 E0 - E1
Hoch spezialisierte Hardware, energy Harvesting
Problematische Entsorgung
Stacked Die
Topologiekontrolle problematisch

Sensornetze Meist C1 E0 - E2

Datenmessung
Drahtlose Kommunikation zur Senke
Batterie zur Energieversorgung → kein Kabel → mobil, aber zeitlich Beschränkt
Einige Gramm schwer, wenige cm³ groß
Maßgeschneidert auf Anwendung
Geringe Leistungsaufnahme
8-bit CPU, ~5kB RAM
Energy harvesting möglich
Teilweise austauschbare Sensor / Aktor Module
Typische Komm. Schnittstelle: Funk: IEEE 802.15.4 / WLAN / ZigBee / BT
Kabel: I²C, SPI, UART, RS323

Physical & Embedded Computing C2 E1 - E3

Smart Metering
Bastlerprojekte
Home-Automation (Gewächshaussteuerung)
Mehrere Sensoren & Aktoren, Leuchtdioden, Displays, Kameras, ...
Wahlweise Betrieb an der Steckdose
Modularer Aufbau, steckbare Zusatzboards

Smart- und Submetering C2 E3

Zeitnahe und periodische ~~Erfassung~~ Erfassung von Energieverbrauch
beispielsweise an häuslichen Geräten (Heizung, ...)
Bestandteil der Smart grid vision
Zähler ↔ Gateway RS485, wireless M-Bus
Gateway ↔ ~~HAN~~ WLAN / Ethernet
Gateway ↔ WAN GPRS / DSL

Smart Home C0 - C2 E0-E3

Hausautomation und -monitoring durch (drahtlose) Sensornetze
Funk ermöglicht Flexibilität + einfache Erweiterbarkeit
Steuerung vor Ort / über das Internet
(Herstellerspezifische) Gateways
Einfache Konfiguration vs. Sicherheit
Zugriffsschutz, Benutzerverwaltung?
Robustheit & Energieeffizienz?
Skalierung?
Zeitkritische Aufgaben?
Hersteller sind im Wohnzimmer und telefonieren nach Hause!
Kernbereich persönlicher & privater Lebensgestaltung prinzipiell
ungeschützt / öffentlich
Abhängigkeit: Türsteuerungsserver nicht erreichbar → Tür zu.

Drohnen & Roboter C2 E1

Einsatz in kritischen, lebensfeindlichen oder unzugänglichem Gebiet
Unterstützung / Hilfestellung für den Menschen
Abstandssensoren, GPS, Accelerometer
Antriebsmittel, LEDs, Greifer
Infrarot, ZigBee, ~~WLAN~~

Wearables C1 - C2 E1

Datenverarbeitung in Körpernähe
Unterstützung in Alltagssituationen durch bsp. Gestenerkennung
Life-Logging durch Armbanduhren
Health-Status wird aufgezeichnet, Löschen nicht vorgesehen
Welche Daten wo hin gehen unklar

Smart Phones C2 E1
Schnittstelle IoE ↔ Mensch
Apps zur Kontrolle von Everything
32bit / 64bit Hardware
Komplexe Betriebssysteme & Programmierung
Kommerziell betrieben
WLAN, BT, NFC
GSM, GPRS, LTE, UMTS
USB

Single Board Computer (SMB) >C2 E9
Prototypingumgebung
Vielseitige Betriebssysteme
Günstig durch hohe Herstellungsstückzahl
Überschaubare HW & Treiber
Sensoren über GPIO & SPI anschließbar
Ethernet

Industrie 4.0 / Industrial IoE C0 - C2 E1 - E9
Leistungsfähige, zuverlässige, robuste und langlebige Steuer- und Regelsysteme

Datenmodell
Betrachtung von Einzelgeräten im IoE schwierig
Ganzes System / Anwendungsmodell besser geeignet
Cloudanbindung
Trend zur Anbindung von Kleinstgeräten an die Cloud zur Datenhaltung
" " Remote-Steuerung per Smartphone/Tablet

Privatsphäre

Privatsphäre

Im IoE ist Privatsphäre stärker gefährdet, als im Internet
- Lifelogging am Körper, Gewohnheiten, Gesundheit, Hygiene...
- Sensoren messen 24/7 sensible Daten: Puls, Bewegung,...
- Verkettung dieser Daten erlaubt detailliertes Nutzerprofil

Angreifermodell in IoE
- Angreifer korrumpiert Menge von Geräten durch bspw. Ausnutzen einer Konfigurationslücke im Gerät
- β = Anteil von korrumpierten Geräten
- Kommunikationsprotokolle müssen bspw. $\beta = 10\%$ aushalten und sicher funktionieren
- IoE Geräte befinden sich häufig in Reichweite an öffentlichen Orten
- Angreifer kann Geräte zerstören oder klauen und zerlegen

Personenbezogene Daten landen oft in der Cloud
- ⇒ Intransparent: welche Daten werden wie & wo gespeichert?
- Angreifer legt Cloud lahm ⇒ ich komme nicht mehr rein :o

Smart Traffic benötigt Positionsdaten, ansonsten sinnlos
- Google Live Traffic ortet Telefone, anonymisiert per gleichbleibende ID ... aber dadurch, dass ich nach Hause fahre, wissen die, welche ID wo wohnt ⇒ ↯ smart traffic idee

Smart Metering
- Wo, Wie, Wann, Wie viel Strom verbraucht wird, wird erfasst
- Intelligente Stromzähler erlauben Einblick in Privatleben
- MDL (Messdienstleiter) kann Daten verkaufen ⇒ Schutzbedarf!

Schutzziele

Confidentiality (Vertraulichkeit)

Ein System bewahrt Vertraulichkeit, wenn es keine unautorisierte Informationsgewinnung ermöglicht.

Integrity
 stark: unautorisierte Manipulationen der Daten unmöglich
 schwach: " " " " fällt auf
Availability (Verfügbarkeit)
 keine Unautorisierten Eingriffe beeinflussen die Funktionalität
Gerade in IoT schwierig, da
- oft keine Hardware (Energie, CPU, RAM) dafür vorhanden ist
- oft keine zentrale Infrastruktur vorhanden ist (Vertrauensanker)
- oft das Vertrauensmodell unklar ist

Privatsphäre ist ein grundlegendes Menschenrecht, wichtiger individueller Wert, essentielles Element für die Funktionsweise der Demokratie
In der digitalen Welt herrschen massive Machtunterschiede zwischen datenverarbeitenden Unternehmen/Individuen und denen, die sie erzeugen
BDSG fordert Schutz für personenbezogene Daten
Weitreichender: privacy-relevant Data
Daten, die zwar einer Person nicht zugeordnet sind, können für die Privatsphäre einer Gruppe relevant sein
⇒ Daten, die ermöglichen, eine Person zu identifizieren, nachdem sie mit anderen Daten in Verbindung gebracht wurden
(Positionssamples & ID der Position ⇒ ID zuordenbar)
Metadaten können privacyrelevant sein
Unverkettbarkeit
Ein System bewahrt Unverkettbarkeit, wenn personenbezogene Daten aus zwei Kontexten für einen Angreifer nicht miteinander in Bezug gesetzt werden können

präventive Verfahren: Datenvermeidung, Separation von Domänen, Pseudonymisierung / Anonymisierung

Transparent

ein System ist transparent, wenn die Verarbeitung pers. Daten nachvollziehbar und überprüfbar ist, zu jedem Zeitpunkt

präventive Verfahren: Logdaten erzeugen, Dokumentation, source code, privacy policy

Intervenierbarkeit

Ein System bietet Intervenierbarkeit, wenn Nutzer über die Art der Erfassung & Verarbeitung ihrer pers. Daten selber bestimmen können.

präventive Verfahren: Schaffung von Wahlmöglichkeiten, manuelles Überschreiben von atom. Einstellungen

Nicht-Identifizierbarkeit

Unentdeckbarkeit

Abstreitbarkeit

Regulierung des Datenschutzes

durch Regierung: BDSG, DSGVO, auch auf Landesebene, ...

durch Organisationen: TÜV, TRUSTe, ...

durch eines selber per PETs (Privacy Enhancing Technologies)

Privacy by Design

meist nicht primäre Anforderung, kann im Konflikt mit Anderen stehen benötigt Risikoanalyse, Vertrauensmodell, Angreifermodell, Analyse des Systems

Daten-orientierte Strategie, um Privacy by Design zu gewährleisten:

Minimize

beschränke Menge an pers. Daten auf das Wesentliche / Minimum

Auswählen vor Sammeln / Senden

Anonymisieren / Pseudonymisieren wenn möglich

Hide
Kommunikationsmuster (spatial, temporal) verschleiern
Pseudonymisieren
Mix-Netzwerke
Verschlüsselung
Seperate
pers. Daten verteilt aufbewahren, dann dort Lokal auswerten
weg von der Cloud! ⇒ hin zu distributed cloud
Aggregate
aggregiere pers. Daten so weit möglich & sinnvoll
über die Zeit oder über Samples
K-Anonymität
Ansätze generell anwendungsspezifisch, keine Universelle Lösung

Device Democracy
Warum das IoT einen Neustart benötigt
- Kosten der Konnektivität
- Fehlendes Vertrauen
- Nicht zukunftssicher
- Fehlender funktioneller Wert, Wertzeugung
- Kaputte Geschäftsmodell
- Teure Serverfarmen als Cloud
Anforderung an dezentrale Cloud
Trustless Transactions
Technische Garantien statt Vertrauen
Gewährleistung von Privatsphäre, Integrität, ...
Robuste & skalierbare Koordination von Geräten
Konsistenz
keine zentralen Vertrauensanker dürfen benötigt werden
⇒ Blockchain?

Smart-Metering

Periodisches Senden von Verbrauchswerten liefert Einblick in Privatsphäre (Urlaub, Arbeitszeiten, ...)

Pseudonymisierung aufwändig benötigt Vertrauensanker, bietet Verkettbarkeit, hilft nicht gegen Lokalisierbarkeit über IP Adresse

Modifikation des Verbrauchs umgehen hilft nicht gegen Kochen

Verbauen von Akkumulatoren im Keller teuer, was ist, wenn er leer/voll ist? Hohe Betriebskosten

Daten Aggregieren, bevor sie gesendet werden

heutige Praxis: ablesen jedes Jahr -> Aggregation über Zeit

Aggregation über Haushalte schwierig:

Aggregation muss woanders als beim MDL passieren, oder jeder übermittelt verfälschte Werte, Summe ist aber korrekt -> Kooperation der Stromzähler

Topologie? Protokoll? Vertraulichkeit? Ausfälle?

Homomorphe Verschlüsselung teuer

MDL kann korrumpierte Zähler platzieren

SMART-ER

Einteilung von Haushalten in Gruppen, durch MDL

Größe Variabel

Innerhalb der Gruppen werden maskierte, korrekte Werte aggregiert und an MDL gesendet

Pro Messintervall tauschen die Haushalte in den Gruppen Zufallswerte aus. Diesen Zufallswert rechnen die Haushalte auf ihren Messwert drauf, und die anderen Haushalte ziehen sie wieder ab $\Rightarrow \sum$ korrekt

Wählt MDL $n-1$ korrumpierte Haushalte, so ist ein Angriff möglich

Gegenmaßnahme: dezentrale Gruppenbildung, Speed Dating

Elderberry-Tree mit P2P overlay
Skaliert gut, allerdings komplexer als Speed-Dating
Smart-Traffic
Zu Verkehrssicherheit: Car 2 Car Comm / Car 2 X Comm
Zur Verkehrsaufkommensoptimierung: Fahrzeugnavigation
Statisch / adaptiv / koordiniert
akt. Verkehrslage ↑ akt. Routen anderer Autos
Samples müssen Position enthalten
Selbst mit Fahrzeugen als Dienstleister (hier wäre die Position der Datenquelle zwar offiziell, aber nicht pers. bezogen)
löst nicht das Problem, dass Empfängerposition benötigt wird.
Implizite Positionsbestimmung über Signalstärke des nächsten Netzmasten und dessen Position (Nutzer kann nicht Präzision bestimmen)
Explizit: Nutzung von GPS (Nutzer kann verschleiern o.Ä.)
⇒ Rückschlüsse auf Privatleben / Gewohnheiten
selbst wenn nicht explizit zuordenbar, kann Kontextwissen (Wohnort / Arbeitsplatz) identifizierend sein
Verwendung von oft wechselnden Pseudonymen, nicht während Bewegung / alleine in der Umgebung
Verschleierung / Mix von Pseudonymen & random Stille nach Pseudonymwechsel
Funkstille in Mix-Zone @ hohem Verkehrsaufkommen & niedriger Geschwindigkeit
Zeitliche: oder spatiale Verschleierung, Präzision senken, Dummy-Werte
Virtual Trip Lines: Position nur teilen, wenn Trip Line passiert wurde, bspw. auf Autobahn (pers. nicht bezogen)
Alles außer Trip Line ist Mix Zone
Aber: Fahrzeuge fahren i.d.R auf Straßen, auf Autobahnen

wird nicht geparkt ⇒ Kontextwissen vermindert # spatiale Verschleierung, zu große spatiale Verschleierung mindert Qualität d. Serv

Blockchain

Schwarzes Brett, nur schreiben, nicht löschen
Einträge haben Zeitstempel
P2P-Netz, Jeder Client kennt die gesamte Blockchain
Neue Transaktionen werden geflutet
Blocks sind Bestandteil der Blockchain, enthalten mehrere Transaktionen & Referenz auf den vorherigen Block
Mining ≙ Erstellung eines gültigen Blocks
der einzige Nachfolgerblock
Widersprüchliche Blöcke werden bei der Erstellung ausgeschlossen ⇒ Blockchain = längste Sequenz von validen Blöcken

Bitcoin

Dezentrales Zahlungssystem, Konten entsprechen Schlüsselpaaren
Überweisungen ≙ Transaktionen
Bitcoin ≙ Block
Nicht # ganz anonym, getätigte Zahlungen sind öffentlich
Enttarnung möglich über Kontextwissen, regelmäßige Einkäufe
Mixen von Transaktionen kann Anonymität gewährleisten

Kommunikation

Topologiekontrolle

Dichte # hoch ⇒ Komplexität der Topologie hoch
Ziel: Komplexität beherrschbar halten ⇒ niedrig halten, ohne Konnektivität zu verlieren
Flaches Netz, gleichberechtigte Systeme, Sendeleistung regulieren, Zahl der Nachbarn regulieren
Hierarchisches Netz, Systeme haben unterschiedliche

Aufgaben & Fähigkeiten, Backbone bilden, Cluster bilden

Flache Netze werden erreicht durch reduzierte Sendeleistung, oder durch Verzicht auf Verbindungen
Sendeleistung einstellbar individuell oder für gesamtes System
Problem: "richtige" Sendeleistung = ?
Tradeoff zwischen Konnektivität, Durchsatz, Energiebedarf & Komplexität
Minimale Maximale Sendeleistung (MMS)
Ziel: Sendeleistung der Einzelgeräte minimieren
Gegeben: Position der Systeme:
Greedy: Sendeleistung der am nächsten beisammenliegenden Systeme erhöhen, bis Konnektivität hergestellt ist, anschließend alle Verbindungen, die Konnektivität nicht beeinflussen, abschalten

Hierarchische Netze
Backbone Netz: 2 Klassen 1. Dominating Set D, zusammenhängend
2. Alle Systeme außerhalb von D, dürfen nicht direkt miteinander kommunizieren, haben mindestens eine direkte Verbindung nach D
Cluster: Mehrere Gruppen von Geräten (C_i), die miteinander kommunizieren. Edge-Geräte verbinden Cluster miteinander
Vertreter eines Clusters, genannt Cluster-head.
Cluster heads dürfen nicht miteinander verbunden sein
Gesucht: maximale unabhängige Menge (NP-vollständig)
Cluster head wird bspw. gewählt nach lokal ausgetauschtem Attribut

Nach der Bildung der Mengen C_i => Gateways finden, Cluster verbinden

Jeder Cluster head braucht Konnektivität zu allen anderen Cluster heads, max Hops: 3 => Backbone Konnektivität

Formal (~~der~~ vereinfachtes) Steierbaum-Problem, NP-vollständig

Cluster heads werden viel belastet => fallen schneller aus

LEACH (Low Energy clustering Hierarchy)

Regelmäße Umverteilung der Rollen => hoffentlich ausgeglichenerer Energiebedarf der Knoten

Prinzip p% der Knoten sind Cluster-Heads

Verteilter Aufbau der Cluster-hierarchy

Annahme: Cluster-Heads kommunizieren mit Basisstation (hohe Sendeleistung)

Knoten bestimmen mit W'keit p, dass sie jetzt Cluster-Head sind und teilen dies mit. Empfänger reihen sich ein

=> Advertisemant-Phase, jeder Knoten wird 1mal Head in 1/P Runden

=> Cluster-Set-up-Phase: Knoten reihen sich neuem Head ein, lassen dies den Cluster head wissen.

Cluster head sammelt Mitteilungen von Nodes und erstellt TDMA-Zeitplan, den er allen Knoten in seinem Cluster bekannt gibt

Steady State => Cluster Head muss aktiv bleiben. Normaler Betrieb

Nach Zeitintervall d beginnt Runde r+1

Eventuelles ~~Ea~~ Verbessern des LEACH Algorithmus: ~~der~~ p wählen basierend auf restlichen verbleibenden Akkukapazität

MAC

Netzzugriff via Funk

unzuverlässig, hohe Fehlerraten, geteiltes Medium

Viele Kollisionen

im IoT besonders:

selbstorganisierend, lange Lebenszeit erwünscht, Robustheit gegen Ausfälle, Topologieänderung (durch bspw. schlafende Geräte), Skalierbarkeit, Sicherheit & Schutz der Privatsphäre

Niedrige Datenraten, Verbindungsabbrüche, schwankende Qualität & Verzögerungen, Luftmedium bietet alle Angriffsflächen, Regulierung der Frequenzbereiche, unidirektionale Links

Signalausbreitung

nimmt quadratisch mit Distanz ab, von Frequenz abhängig, kann durch Hindernisse blockiert werden, idealerweise radial

Warum kein CSMA/CD?

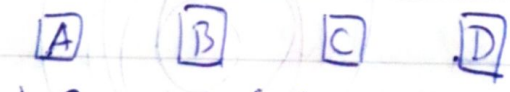
Versteckte Systeme

System A ist für C versteckt

A sendet an B

C hört A nicht und sendet an B, dort Kollision

Ausgelieferte Systeme



C ist B ausgeliefert

B sendet an A

C will zu D senden, muss warten

Signale mit größerer Stärke übertönen schwächere

Medienzuteilung

Verwendung von Zeitmultiplex

- schlafen legen
- Kollisionsvermeidung
- Kein unnötiges Lauschen (Idle Listening)
- Kein unnötiges Mithören (Overlistening)

} Akku sparen

Problem: keine Duplexverbindung ... !!

~~Kollisionsvermeidung~~ Kollisionsvermeidung:

Out-of-band: auf anderer Frequenz In-band: RTS/CTS

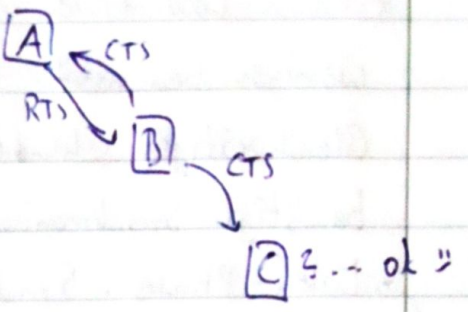
RTS / CTS - Handshake

Vermeide versteckte Systeme

A und C wollen zu B senden

A sendet zuerst RTS, was B mit CTS beantwortet

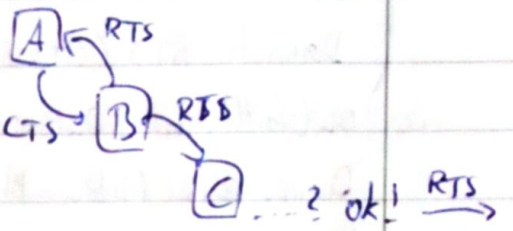
C wartet, da plötzlich ein CTS kommt



Vermeide ausgelieferte Systeme

B will zu A, C irgendwo hin senden

C wartet nicht, wenn nach dem RTS von B kein CTS empfangen wird



Duty-Cycling

Idee: Abschalten des Funk-Interfaces, um Akku zu sparen

Latenz der Übertragung steigt.

Funk Interface bei Bedarf / nach Zeit einschalten

Synchron: Geräte "erwachen" koordiniert

Asynchron: Geräte erwachen irgendwann, senden dann so lange, bis ein Empfänger aufwacht und Paket empfangen kann

WLAN hat Kollisionen, Overlistening & idle Listening

Bluetooth ist langsam, nutzt Frequenz-Hopping, Synchronisation aufwändig & bedarf Zertifizierung zum Einsatz im Kommerz

Bluetooth Smart ⊃ Bluetooth Low Energy

optimiert für IoE-Anwendungen (kleine Dateneinheiten, Reichweite 150m, Duty-Cycling, geringe Latenz, kein Beacon-Spam & Gerätesuchen, built in Krypto, Lizenzfrei

Konkurrent zu ZigBee!

S-MAC (Sensor-MAC)

Ziele: Low Power → kein idle Listening, overlistening, CA

weniger beachtet: Fairness, Latenz

Gleichzeitiges schlafen vermeidet idle Listening

benötigt Synchronisation

- Listen-Phase: Synchronisation + Datenaustausch anstoßen
- Sleep-Phase: schlafen, oder Datenaustausch

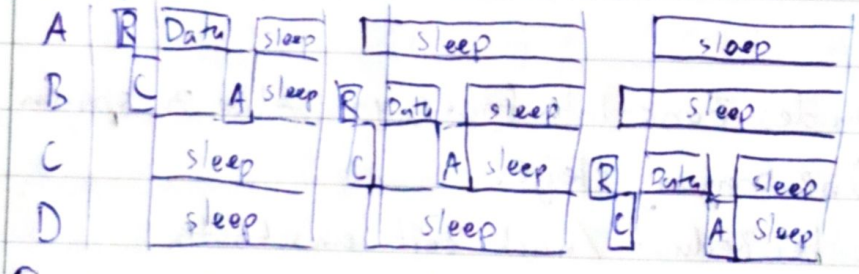
enthält SYNC + Zeit-Information bis zur nächsten Listen-Phase

Danach RTS/CTS-Phase, zufälligen Schlitz zum Senden wählen

entfällt, falls Dateneinheit gesendet/empfangen werden kann

Dauer der Listen-Phase nicht frei wählbar. Ergibt sich aus den Parametern MAC + PHY + Datenrate + ... etc

Beispiel A → B → C → D



Synchronisation durch lernen anderen Zeitpläne. Falls keine empfangen werden, wähle eigenen Zeitplan

⇒ Insel von Zeitplänen.

Geräte an Grenze muss beide Zeitpläne können ⇒ mehr Energieverbrauch.

Zu Beginn: warten auf SYNC von jemand anderen.

Falls Timeout eintritt: wähle Eigenen

Erweiterung Adaptive-Listening

Neue Phase für weiteren Datenaustausch

System hört CTS ohne RTS ⇒ Jemand in Reichweite empfängt etwas, das er vielleicht an/über mich weiterleiten möchte. ⇒ Schlafe nur so lange, wie in CTS als

Datenübertragungsdauer angegeben ist.
⇒ erwache, und lausche auf RTS an mich, wenn es kommt, empfange Paket nach CTS-Quittung
S-MAC: niedriger Energiebedarf, aber geringer Durchsatz

B-MAC
Zielsetzung: Kollisionsvermeidung, effiziente Kanalnutzung bei hohen & niedrigen Datenraten
Skalierbarkeit mit X Systeme
Toleranz gegenüber sich verändernden Funkbedingungen
Rekonfigurierbarkeit durch Vermittlungsschicht
Einfach zu implementieren, kleiner Code, niedriger Stromverbrauch
Besonderheiten: Periodisches Prüfen des Kanals statt zeitl. Synchronisation
Behandelt keine versteckten Systeme
Keine Fragmentierung
Prinzip: Geräte erwachen sehr kurz, prüfen Kanal auf Daten (Clear Channel Assessment (CCA))
Wach bleiben, falls Daten zu senden sind
Dann prüfen, ob Kanal frei ist } Collision Avoidance
übertrage Präambel }
Intervall zwischen Aufwachen & Präambellänge muss stimmen
Je weniger Knoten um den Kanal konkurrieren, desto höher der Durchsatz, dann z.T. 4,5x so hoch wie bei S-MAC
Erweiterungen: ACKs, CTS/RTS
Hinsichtlich Energie: Bei S-MAC wird höherer Durchsatz durch längere Duty-Cycle erreicht ⇒ Linearer Energieverbrauchzuwachs
Bei B-MAC wird die dadurch kürzere CCA-Intervalle erreicht ⇒ Sublinearer Energieverbrauchzuwachs
End-to-End Latency bei B-MAC linear, bei S-MAC auch.

hat allerdings bei adaptive-listening einen großen offset

BMAC: CCA + Low Power Listening (LPL)

+ Keine Synchronisation notwendig
+ einfaches Protokoll, kompakte Implementierung
+ geringe E2E-Latenz
- erfordert Präambel (Dauersendung)
- keine Behandlung von versteckten Geräten

IEEE 802.15.4

Standard, definiert Phys und MAC
Genutzt von ZigBee & 6LoWPAN
Ziele: Übertragung von kleinen - mittleren Datenraten
Moderate Verzögerung
Geringer Energiebedarf (Batterie für Monate / Jahre)
Geringe Komplexität
Zielanwendungen: Sensoren, Gebäudeautomatisierung, Spielzeug, Etiketten, Fernbedienungen
Frequenzbänder: 868 MHz / 914 MHz / 2450 MHz.
max 250 kb/s, 10m Reichweite
max 127 B / Segment
Systeme können unterschiedliche Rollen annehmen
Basiert auf CSMA/CA
Kombiniert Zeitplan & Konkurrenz-Verfahren
2 Klassen der Systeme
FFD (Full Function Devices)
RFD (Reducted Funktion Devices)
Pro Netz ein FFD, Koordinator, kommuniziert zu RFDs
mit Beacons: synchronisiert, fordert auf zum Senden, Identifikation des Netzes

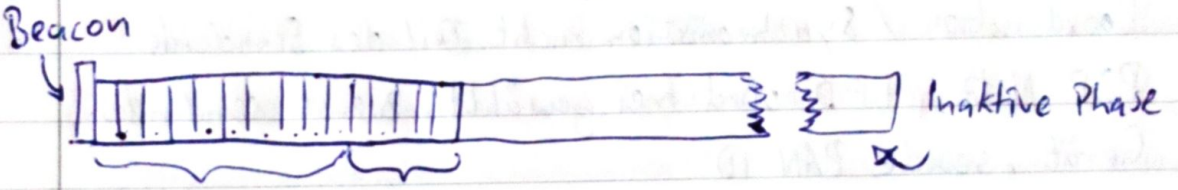
⇒ 2 mögliche Topologien P2P-Netz (Mesh) oder Stern

Beacon-Modus

Stern Netz mit PAN (Personal-Area-Network) Identifier

Koordinator: vergibt Adressen an seine RFD
sendet regelmäßig Beacons
bearbeitet Anforderungen für garantierte Zeitschlitze
vermittelt zwischen Systemen und Peer-Koordinatoren

Grundlegende Annahme: für den Koordinator ist Energie nicht limitiert, sie übernehmen mehr Aufgaben als RFD



Konkurrierender Zugriff → CAP (Contention Access Period) Zugriff via slotted CSMA/CA

Garantierte Zeitschlitze
↳ müssen beim FFD angefragt werden, dann reserviert

CSMA/CA: Warte zufällige # Zeitschlitze (exp. Backoff)
Prüfe ob Medium frei ist. Frei? → Senden!
Nicht frei? Backoff *=2; Warte nächsten Beacon ab
Maximale # Neustarts erreicht? → Abbruch !!

GTS: ACK vom FFD bei Empfang von GTS Anfrage.
GTS-Vergeben werden im nächsten Beacon bekannt gegeben
Reservierte Zeitschlitze bleiben bis explizite Freigabe erfolgt belegt. Koordinator überprüft Nutzung und kann diese daraufhin eventuell wieder freigeben.

Dateiübertragung
RFD → FFD
if (IFS + Payload + ACK ≤ GTS) sende über GTS
über CAP mit CSMA/CA sonst.

FFD → RFD
if (IFS + Payload + ACK < GTS) then sende über GTS
Sonst: nutze Beacon, um Datenwunsch (sende) an RFD zu senden
RFD antwortet mit ACK in CAP
FFD sendet Daten in CAP
RFD quittiert Daten in CAP mit ACK
bei jeglichem Fehler (durch Kollisionen o.Ä.): wiederhole!

Non-Beacon-Modus
Keine Rahmenstruktur
Zugriff via unslotted CSMA/CA, da keine Synchronisation
Koordination / Synchronisation nicht Teil des Standards
P2P-Netz, FFD wird frei gewählt, zuerst sendendes Gerät, sende PAN ID
Ziel: Einsatz in Industrie 4.0
Höhere Zuverlässigkeit als im Beacon-Modus
Deterministisches Zeitverhalten

Alternative Betriebsmodi:
TSCH (Time-Slotted Channel Hopping)
effizienter, planbarer und robuster Medienzugriff
vermeidet idle-listening
DSME (Deterministic & synchronous Multi-channel Extension)
flexibel, insbesondere bei Mesh-Netzen
Anpassbar: Zeitverhalten, Datenrate, Energiebedarf
Beacon-Modus, aber Reservierungen kanalspezifisch
→ Synchrones Zeitmultiplexing
Einteilung des Mediums in feste Zeitschlitze, Anzahl fix
Synchrone Wach- und Schlafzeiten
Wechsel der Frequenz für jede Dateneinheit
⇒ Sendewiederholung auf anderer Frequenz

Vorteile: + Planbare zeitliche Belegung des Mediums
+ Bandbreitenzuteilung möglich
+ Frequenzwechsel verbessert Robustheit, Interferenzen / Störungen betreffen selten alle Kanäle

Slotframe - Struktur
Menge von sich wiederholenden Slots
Zeitplan gibt vor, was welches Gerät in welchem Slot tun darf (Senden / empfangen / schlafen)
enthält Information, welcher Nachbar wach ist und auf welchem Kanal gesendet werden soll
Zeitschlitze lang genug für DATA+ACK + Sendewiederholung (~10ms)

802.15.4e beschreibt, wie Layer 2 einem Zeitplan folgt.
Zeitplan beschreibt Kommunikation aller Systeme auf einem Link. Mobilität? Variierende Datenraten?
Erfordert dezentrales oder zentrales Scheduling

Abgrenzung zu DSME
Slots für paarweise Kommunikation reserviert & definiert durch Zeitpunkt & Kanal
Aushandlung ohne FFD, RTS/CTS-ähnlicher Handshake mit Veto-Möglichkeit

Senden von größeren Datenmengen führt nicht zwangsweise zu höherem Energieverbrauch
Bei IEEE 802.15.4 Unterschied minimal, Sender länger auf Transmit, Empfänger kürzer auf Receive
S-MAC Transceiver länger auf Transmit, kürzer im sleep
=> höherer Energiebedarf
BoX-MAC-2 Transceiver für lange Zeit nur auf Transmit, kürzer in Receive. Transmit ist "günstiger", daher Energieeinsparung!

Routing

Fluten

Nutze MAC-Broadcast Adresse zum Versenden von Nachrichten

Alle Empfänger erwidern mit Broadcast

+ Dezentral, keine Wartung

+ keine Routingtabellen

- hohe Netzbelastung, Implosion

- terminiert nicht

- unzuverlässig

Erkennung von Duplikaten benötigt Speicher, Dateneinheiten müssten eindeutig identifizierbar sein

Wenn überhaupt, dann Verwendung probabilistischer Verfahren

Begrenzte Reichweite (TTL)

+ Geringere Netzbelastung

- immer noch keine Zuverlässigkeit

Gossiping

Wähle W'keit p ($\in [0{,}65 ; 0{,}75]$) für Weiterleitung

+ Keine Implosionen

+ Geringerer Overhead als Fluten

- evtl. lange Übertragungszeit bei „unsinnigem" Pfad

- keine Zuverlässigkeit

Kombination von Fluten & Gossiping

$k \hat{=}$ # Hops Fluten

$p \hat{=}$ W'keit dann für das Gossiping

$(\#1, k) \Rightarrow$ Fluten

$(p, 0) \Rightarrow$ Gossiping

Erfahrung / Tests: 1000 Knoten $\Rightarrow k = 4$, $p = 72$ ist gut :)

Verbesserung: variierendes p, basierend auf Distanz zum Ziel.

Voraussetzung: Jeder Knoten kennt Distanz zum Ziel

Directed Diffusion

Inhaltsbasiert, dezentral, findet Pfade von Quelle zu Senke

Unterstützt Aggregation

Besonders geeignet bei einer Datenanfrage, regelmäßige Antworten

Senke äußert Interesse via Fluten

Knoten speichern Richtung, von wo das Interesse empfangen wurde => Gradient mit Zeitstempel, enthält Datenrate

durch Fluten & wahrscheinlich bidirektionale Gradienten, aber durch Zeitstempel nicht symmetrisch

Daten werden auf Rückwärtspfaden, in Richtung Gradienten gesendet, durch bidirektionale Gradienten erstmal = Fluten

Zu Beginn etablieren sich mehrere schwache Gradienten, quasi willkürlich

Senke startet Reinforcement-Phase, sobald Interesse ankommt

Gradient und Datenrate wird verstärkt in Richtung Quelle

Knoten geben dies weiter in Richtung Quelle

=> etablieren von wenigen guten Pfaden die ausschließlich zur Datenübertragung dienen. Timestamp spielt hierbei zentrale Rolle.

Andere Metriken, bspw. # Nachbarn, # Empfangene Pakete, ... möglich. => Anpassung an Topologieänderungen

Vergleichen zu traditionellem Routing:

Reaktiv vs. Proaktiv (Reaktion auf Interessenbekundung)

2-Phasig, wobei immer Daten übertragen werden (traditionell: nach Aufbau des Pfads)

Duplikatenerkennung anhand Zeitstempel

Aggregation, Interpolation möglich

Nur lokale Kommunikation (traditionell: E2E)

Rumor Routing

ähnlich wie Directed Diffusion, hier aber werden auch Daten anfangs geflutet ⇒ 2 Pfade entwickeln sich unabhängig:
Interessenpfad, Quellenpfad
Knoten werden bei aufgezeichnetem Ereignis mit W'keit p ~~Daten~~ Agenten senden und fluten ihre mit Pfadinformation durch das Netz. Nutzen dabei TTL-Feld
Knoten benötigen Information über ihre Nachbarn ⇒ periodische Hello-Dateien
Benötigen Ereignistabelle mit Weiterleitungsinformation zu bekannten Ereignissen, Updates durch ankommende Ereignis-Agenten
Einträge sollten limitierte Lebensdauer haben
Werden per random walk weitergeleitet,
treffen sich 2 Pfade
Fluten oder Rumor Routing?
Query Flooding bei wenigen Knoten ⇒ kürzesten Weg ein paar duplizierte Pakete
Danach Rumor Routing ⇒ wenige Pakete, aber auch nicht
Event Flooding bei sehr vielen Knoten. Gradient zum kürzesten Pfad
Dann Event-Anfragen günstig Ereignis aufbauen

Daten- vs. Adressbasiertes Routing

Routingansatz	Adresse	Inhalt
Voraussetzung	unique Adress	vordefinierte Semantik
Routingverfahren	pro- oder reaktiv	Fluten, Reverse Routing
Vorteile	niedrige Latenz	keine Adressinformationen
	niedrigen Overhead	Redundanz
Nachteile	unique addresses needed	hoher Overhead für einzelne Verbindung

Lokationsbasierte Verfahren

Manchmal ist Interesse über ein Gebiet bei Koordinaten XY gegeben, oder "mehr als X m weit weg", "am nächsten dran"
Bei Abbildung von Systemnamen auf Position ⇒ Routingtabelle entfällt
Falls Position = Adressabbildung: implizite Ordnung auf Adressen, geeignet für Greedy-Ansätze
Strategie: next hop always closer
+ Schleifenfrei
- nicht die minimale #hops garantiert
- Pfade am Rand der Reichweite des Senders bevorzugt ⇒ instabil
Verbesserung:
Minimaler Abstand zur Luftlinie } nicht garantiert Schleifenfrei
Minimaler Winkel zur Luftlinie }
- Laufen womöglich in Sackgassen

Greedy Perimeter Stateless Routing

Rechte-Hand-Regel (um Labyrinth zu entkommen)
wahsle von Greedy in Perimeter-Modus, falls der nächste Greedy-Schritt eine Verschlechterung birgt
Merke bis dato beste Distanz zum Ziel in der Dateneinheit. Beginne mit Rechte-Hand-Regel, Face zu umrunden, das von der Luftlinie zum Ziel als erste geschnitten wird ⇒ zwangsweises Verlassen des Perimeter-Modus nach halber Umrundung des Faces ⇒ Greedy übernimmt
Erfordert planaren Graphen, oft nicht gegeben ⇒ Graph ausdünnen
Bei vollständigem Graphen: Greedy optimal

Distanz-Vektor-basiertes Verfahren: RPL (Ripple)

Aufbau von DODAGs (Destination-oriented-distributed Acyclic Graphs)
Erlaubt neben Definition von Metrik eine Auswahl von Constraints

Verkehr in Richtung Wurzel, Internet: „up" concast
" " " Kinder, Knoten: „down" multicast
P2P: nutzt beide Wege, up & down
Konstruktion des DODAG: Start an der Wurzel, Knoten empfangen Nachricht von Wurzel und entscheiden über Beitritt. Falls sie beitreten und weitere Knoten in Richtung down haben, leite Beitritt an sie weiter, verkünde „Weg zur Wurzel"
DODAG-Wald kann mehrere Wurzeln enthalten, diese können logisch zusammengefasst werden - Allerdings kann ein Knoten maximal einen Wurzelknoten im DODAG haben
Bei Topologieänderung kann DODAG nicht mehr optimal sein/ werden
Node-Rank als relative Position zur Wurzel, wächst monoton „down"
Kann Metriken & Constraints enthalten, definiert wie viele Elternknoten gewählt werden sollen
Nachrichten: DIO (DAG Information Object)
ermöglicht Entdecken einer RPL Instanz, Lernen der Metriken und DODAG Eltern zu wählen
DAG Information Solicitation (DIS)
Anforderung eines DIOs
DAO (Destination Advertisement Object)
zur Kommunikation von Routingzielen entlang des DODAGs
Storing / non-storing Nodes
Ziel: Etablierung von Routinginformationen in „down" Richtung
Problem: Speicheraufwand
non-storing: Source-route wird bei Ankunft an der Wurzel in Dateneinheit hinzugefügt
storing-nodes:
Knoten merken sich Pfade „unter" ihnen
Sicherheit: teuer, selten auf Resourcenarmen Geräten möglich

RPL - Optionen
Unsecured: keine zusätzlichen Maßnahmen
Pre-Installed: Vorinstallierte Geheimnisse auf den Knoten
Authentifiziert: Beitritt der Blätter nur mit vorinstalliertem Key
Als Router muss Key von einer Autorität erworben werden
Schutzziele: CIA, Schutz vor Wiedereinspielen, Verzögerungen, ...
DODAG-Konstruktion: Knoten schicken periodisch link-lokale DIO-Nachrichten
Andere Knoten nutzen diese Info, um DODAG beizutreten oder diesen zu verwalten (Anforderung von DIO via DIS)
Ableitung des Node-Ranks aus DIOs
Node-Rank reicht aus, um DODAG Beitritt zu vollziehen, Wurzel selber muss nicht bekannt sein
Knoten dürfen ihren Node-Rank nur verkleinern.
Node-Rank leitet sich ab aus bspw. Distanz + Metrik + Constraint + Runden (Aufrunden)
Ein höherer Node-Rank kann vorteilhaft sein: mehr Eltern zur Auswahl (Robustheit), weniger Weiterleitungstraffic
Count-to-Infinity-Problem, wenn Knoten abwechselnd sich entscheiden, einander unterzuordnen -> max_depth-rule!
Durch ständige Topologieänderung ist Schleifenfreiheit nicht zu garantieren, oder sehr teuer
Erreicht ein Knoten einen Node-Rank > max_depth, so löst er sich aus dem DODAG und sucht sich einen neuen Elter
Schleifen im Datenpfad immer noch möglich bei Knotenausfall
RPL: Piggypacking von Routinginformationen in Dateneinheiten
Bspw. durch setzen von Flags im Datenkopf (IPv6 Header, ...)
=> Datapath Validation: markiere Richtung „up" / „down" im Paket zusammen mit Node-Rank des Absenders

Erkennung von veirrten Datenpaketen möglich
=> DODAG Reparatur: Neuwahl des Elternknotens dort, wo Inkonsistenz erkannt wurde, dann Reparatur / aktualisieren der Node-Ranks der Kinderknoten
Poison & wait: falls kein neuer Elternknoten gewählt werden kann: setze Node-Rank auf +INF, warte random-lange dann rejoin
Globaler Rebuild als Notfallstrategie => hinterher bessere Pfade (Theor.!)
Alternative: Neuberechnung der Node-Ranks bei gleichen Links
MP2P: gut! P2MP selten, wird nicht betrachtet, kaum möglich
P2P: nie. Wenn, dann Kommunikation über Wurzel
Fragmentierung bei Verwendung von 802.15.4 auf Layer 2
RPL ausgelegt für IPv6 (1280B +)
Source-Routing bedarf ebenfalls Platz im Payload
Verlust eines Fragments => Verlust des gesamten Pakets!!!
Aggregation von Adressen bedarf gewisser Form der Adressen.
bei häufigem Elternwechsel kaum denkbar => storing-mode hat schnell volle Routingtabellen
Eltern können nur noch redundant gewählt werden, wenn ein gemeinsames Präfix vorhanden ist.

TRANSPORT

Besonderheiten wie immer im IoE:
wenig Batterie, unzuverlässiges Medium, MP2P, Concast, wenig Speicher, keine Verschlüsselung, Datenaggregationen
Teilweise können angefragte Daten zeitnah an Wert verlieren
Typische Kommunikationsmuster: Unicast (Sensor → Aktor)
Multi-/Broadcast: Senke → Knoten (Config/Software-Update)
Concast: Knoten → Senke (Messdaten, Status) Achtung! Stau!

Herausforderung: Zuverlässigkeit
Alle Datenpakete kommen an, unverändert, reihenfolgetreu, duplikatfrei und ohne Phantom-Paketen
Mechanismen von TCP dazu: Zeitgeber, ACKs, Sequenznummern, Sendewiederholungen (Automatic Repeat Request), FEC
Warum kein TCP im IoE?
TCP kennt die Ursache für Paketverlust nicht, nimmt Stau an
Liefert 100% Zuverlässigkeit, was teilweise gar nicht gebraucht wird => hoher Energiebedarf
Verhindert Datenaggregation, erzeugt head-of-line Blocking
3-Way Handshake, nur für Unicast ausgelegt
Probabilistische Zuverlässigkeit oft ausreichend/gewünscht
Nur ein Prozentsatz der Daten erreicht Senke (W'keit p)
Bei Messen der Raumtemperatur ausreichend, Teile der Information können durch Aggregation kompensiert werden
Hohe Latenzen kommen durch schlechte Funkverbindung zustande
Sendewiederholungen? Wo? Pro-Hop? Am Start der Route?
ACKs sinnvoll, NACKs nicht, da Empfänger nicht weiß, dass ihn ein Paket hätte erreichen sollen (keine offene Verbindung)
bei periodischer Übertragung einsetzbar
Quittungen Hop-by-Hop => geringere Latenz, aber Zwischensysteme müssen Zustand halten, viele Quittungen, teuer
Quittungen beim Empfängersystem, E2E => hohe Latenz, wann sollte ein Timeout gestartet werden? Gesamter Pfad wiederholt Dateneinheit
Bei niedriger Fehlerrate ($p < 0{,}001$) wähle E2E ACKs
Ansonsten Multi-Hop ACKs. Je mehr Wiederholungen pro Hop erlaubt werden, desto später explodiert der Energiebedarf einer Übertragung. ∞ Wiederholungen => Linearer Energiebedarf.

Pro-Hop ACK: Energiebedarf wird im Netz verteilt

HHR (Hop-by-Hop Reliability)

Ziel: höhere Zuverlässigkeit ohne ACKs

Sende Dateneinheit immer k-mal

Wähle k so, dass bei h Hops das Paket mit W'keit r vom Ziel korrekt empfangen wird: $r = 1 - p^k$

Wenn Fehlerrate p bekannt ist, gilt:

$k_{HHR,i} = \frac{\log(1-r_i)}{\log(p_i)}$ Overhead $O_{HHR} = k_{HHR} \sum_{i=0}^{h-1} r^{i/h} = \frac{\log(1-r^{1/h})(1-r}{\log(p)(1-r^{1/h})}$

HHR + ACKs (HHRA)

Sende bis zu $k_{HHRA,i}$-mal, warte aber auf ACKs

Aufhören zu senden bei erstem ACK

$k_{HHRA,i} = 1 + \sum_{i=1}^{k_{HHR}} p^{i-1} = 1 + \frac{1-p^{k_{HHR}}}{1-p_{n-1}} = 1 + \frac{r^{1/h}}{1-p}$

Overhead: $O_{HHRA} = k_{HHRA} \sum_{i=0}^{h} r^{i/h} = \left(1 + \frac{r^{1/h}}{1-p}\right) \frac{(1-r)}{(1-r^{1/h})}$

HHRA lohnt sich erst ab einer hohen Paketfehlerrate ($p \approx 0{,}25$)

HHR + Wenige Pakete durch ausbleiben der Quittungen

\+ insgesamt weniger Datenpakete bei niedriger Fehlerrate

\- hört bei Empfang des " nicht auf zu senden

HHRA + keine weiteren Übertragungen bei ACK

\- unverhältnismäßiger Overhead bei niedriger Fehlerrate

Multicast

Senke sendet Anfrage an Sensoren
Senke sendet Aufforderung an Aktoren
Senke sendet Code-Update an alle
} 1 => n - Muster

Bedarf Zuverlässigkeit

Pump Slowly Fetch Quickly (PSFQ)

Senke sendet 2 Pakete ins Netz

Knoten speichern Dateneinheit, falls sie neu ist

Weiterleitung nur, falls Sequenznummern konsistent

Falls eine fehlt, sende NACK an Vorgänger

Verzögerung zwischen Pump-Operationen groß
Zeit für bis zu ~5 fetch-Operationen
W'keit klein, dass neue Dateneinheit gepumpt wurde, während die alten noch nicht zuverlässig zugestellt sind
Warte zufälliges Zeitintervall $t \in [t_{min}; t_{max}]$ und zähle, wie viele Nachbarn Dateneinheit weitergeleitet haben.
≤ 3 ? => Weiterleiten. > 3 ? dann nicht => Nachbarabhängiges fluten. Fehlende Dateneinheiten währenddessen fetchen
NACKs sind Broadcasts, da Nachbarn u.U. die Dateneinheit auch nicht kennen
∀(fehlende Dateneinheit im NACK) do
  if (Dateneinheit bekannt)
    wait random & listen for Dateneinheit;
    if (Dateneinheit gehört)
      return;
    else
      send Dateneinheit;
    end if
  end if
end for

so lange fetchen / NACKs senden, bis alle fehlenden Dateneinheiten vorliegen, oder obere Schranke für #NACKs überschritten.

Concast
n Sensoren / Aktoren senden zur Senke (1 Senke)
Herausforderungen: Zuverlässigkeit / Stau / Energie-Effizienz
ESRT (Event-To-Sink-Reliable-Transport)
Sensoren senden periodisch Daten an Senke, diese braucht eine bestimmte Anzahl, egal von welchem Knoten
Ziel: in jeder Periode i mit Länge τ sollen ca. R Datenpakete ankommen
Anpassen der Senderate f_i zum Zeitpunkt t_i
Senke misst in i r_i empfangener Datenpakete

Senke berechnet Zuverlässigkeit $\eta_i = r_i / R$

$\Rightarrow$ soll gelten $\eta_i \in [1-\varepsilon, 1+\varepsilon]$

Senke berechnet f_{i+1} basierend auf f_i und η_i

Versende f_{i+1} dann per Broadcast (Annahme: Senke kann durch hohe Sendeleistung alle Geräte erreichen)

Finde OOR (optimal operation region)

f_i zu niedrig: kein Stau, keine Zuverlässigkeit $\xrightarrow{f_{i++}}$ OOR $\xrightarrow{f_{i++}}$ kein Stau, zu viele Daten / mehr als benötigt $\xrightarrow{f_{i++}}$ Stau, zu viele Daten $\xrightarrow{f_{i++}}$ Stau, zu wenige Daten

NC, LR $\rightarrow$ NC, HR $\rightarrow$ C, HR $\rightarrow$ C, LR $\curvearrowleft$ η_i / k

$f_{i+1} =$ f_i / η_i^g $\nearrow f_i$ OOR $\nwarrow f_i / 2 \cdot (1 + 1/\eta_i)$ f_i / η_i f_i

$\eta_i < 1$, wenn zu wenige Daten ankommen

$\eta_i > 1$, wenn zu viele Daten ankommen

k: # Perioden, in der das System in C, LR verweilt

Senke führt Fallunterscheidung durch Vergleich von η_i mit 1 und f_i mit f_{max} durch

Wie groß sollte R sein? So groß, dass Senke immer genügend Informationen für die Systemanforderung hat

Paketfehlerrate? Knotenausfälle? Nicht untersucht :/

Aggregation

Einsetzbar bei Concast $\Rightarrow$ Daten verrechnen, fusionieren, weniger einzelne Datenpakete übertragen (Datenaggregation)

Paketaggregation: Pakete sammeln und dann in einem Größeren weitersenden, Daten werden nicht „verfälscht"!

Beides spart Übertragungen, Größe variiert aber, das muss je nach Protokoll beachtet werden

Datenvolumen hat geringen Einfluss auf Energieverbrauch, eher die Zeit, in der der Transceiver online ist

Aggregation kann Starvitationen verhindern

Systeme

6LoWPAN (IPv6 over Low-Power Wireless Personal Area Network)

Systeme haben global eindeutige IPv6 Adressen

Nutzung von bestehenden Management & Automationsmechanismen

E2E - Möglich, UDP nutzbar

Vorgesehenes Schichtenmodell:

CoAP
UDP
RPL
6LoWPAN
802.15.04

Ziel: Herstellen von IP Konnektivität

Adressierung: IPv6

Fragmentierung, Reassemblierung & Header-Compression

Soll einfach zu konfigurieren sein, Selbstheilung unterstützen, einfaches Bootstrapping

Verbindung zwischen Stub-Inseln: 1 Router mit Ein-Ausgang

gemeinsames IP-Präfix pro Insel

2 Gerätekassen: Router, Host

Edge Router können mit Router Nodes im 6LoWPAN sprechen, um Internet-Konnektivität herzustellen

Edge-Router Protokollstack

IPv6	IPv6
Ethernet MAC	LowPAN Adaption
	IEEE 802.15.4 MAC
Ethernet Physical	IEEE 802.15.4 PHY

Aufgabe: Abbildung von IPv6 Adressen auf 802.15.4 Adressen

Anycast: Dateneinheit wird an ein beliebiges (meist: nächstes) Endgerät mit Adresse aus Menge von Adressaten ausgeliefert

Stateless Address Autoconfiguration: Berechnung der Adresse aus Präfix des LowPANs und Interface ID des Geräts

Edge Router hat bspw. Adresse 2001:300a:1::/64

Dann hat Device 1 im LowPAN Netz die Adresse: 2001:300a:1::1

802.15.4 Frame bietet Platz für 127 Bytes und bietet keine Fragmentierung. => Im worst case 102 Bytes für Nutzdaten,

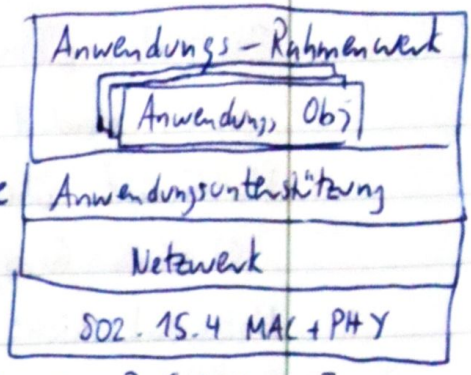
da IPv6 Kopf 40 Bytes und UDP-Header 8 Bytes einnimmt
=> max. 32 Bytes Nutzdaten in worst-case
6LoWPAN Header für Einbettung von IPv6 in 802.15.4
AB = 00 => kein LowPAN Header
01 => Dispatch Header (Komprimierung)
10 => Mesh (Layer 2 Routing)
11 => Fragmentierung
} kann verkettet werden
Fragmentierung:
erstes Feld nach "11": tag zur Unterscheidung, hierbei Kooperation mit Sender/Empfänger-Adresse von 802.15.4
danach: size, dann offset
Header Compression
Nach "01": dispatch Header-Präfix, 6 bit um nachfolgenden Paketkopftyp zu identifizieren (ob UDP oder IPv6 komprimiert wurde)
IPv6: was kann komprimiert werden? Version immer 6
Interface ID kann aus 802.15.4 abgeleitet werden
Payload length eventuell aus L2 Payload length oder L3 data_size abgeleitet werden
Traffic class oft 0
Flow Label oft 0
Next Header nur UDP, ICMP oder TCP
=> Kann auf 2 Byte Kopf reduziert werden
(Plus 1 Byte für das Hop-Limit Feld)
Setze für jede Komprimierung im IPv6 Header eine 1 im Dispatch-Byte von 6LoWPAN
UDP Komprimierung: Ports: nicht 2^{16} notwendig
Payload length aus L3 u.U. berechenbar
8 Bytes => 6 Bytes komprimiert

CoAP (Constrained Application Protocol)
L5, Alternative zu HTTP, setzt auf UDP auf, best-effort
REST (Representational State Transfer)
GET, PUT, POST, DELETE (Aufruf, Speichern, Übermitteln, Löschen)
Interworking mit HTTP, Proxy-Umsetzung
Unterstützung von DTLS (Datagram Transport Layer Security)
Requests / Responses enthalten Methodenaufruf auf Objekt
Message-Schicht
Mechanismen zur zuverlässigen Nachrichtenübertragung
4 Typen: NON, CON, RST, ACK
Client-Server Modell (allerdings rotierende Rollen)
Client sendet Request an Server, um Aktion aus zu führen
Request besteht aus: Auszuführenden Methode auf Objekt
Identifiziert durch Token: Objekidentifikator, Nutzdaten, Optional: Metadaten zum Request
Server antwortet mit Response Code:
2 Success, 4 Client Error, 5 Server Error
Können im ACK Piggy-Packed sein, oder als separate Nachricht
Eindeutige IDs der Anfragen / Nachrichten
NON: Non-Confirmable (unzuverlässig)
CON: Confirmable (zuverlässig, exponential Backoff), dann ACK
RST: Reset bei Fehlern
CoAP nutzt URIs, um Resourcen und deren Lokation zu identifizieren:
coap://example.net:61616/.well-known/core
ACK kann Payload enthalten. Gerade bei GET Anfrage
Bei getrenntem Übertragen: MID eindeutig zuordenbar

MQTT (Message Queue Telemetry Transport)
Wie CoAP, aber Fokus auf Many-to-Many-Kommunikation
Nutzt TCP, UDP in MQTT-SN für Sensornetze
Publish / Subscribe Paradigma
Zuordnung von Inhalten / Nachrichten zu Topics
Nachrichtenspeicherung und -weiterleitung durch zentralen Broker
Sicherheit über TLS / DTLS
Ereignisorientiert, Weiterleitung über Broker
publish (Home/Fridge/Temp, 7°C) → Broker
Broker → publish (")
Nodes (Clients) subscribe (Home/Fridge/Temp) → Broker
Letzter Wille: Dateneinheit von Client an Broker, die Broker sendet, wenn Verbindung zum Client abbricht
QoS enabled, Austausch des QoS-Levels in Sub/pub Nachrichten
Bei Level > 0: Paketen enthalten ID (bestimmt Sendezeitpunkt)
Level 1: Mögliche Duplikate: wenn PUBACK von Broker an publizierenden Client verloren geht
Level 2: explizite Release-Nachrichten. PID wird erst freigegeben, wenn alle Subscriber PUBREL gesendet haben
Keine E2E-Kommunikation wie bei CoAP, wegen Broker
Broker kann Datenquellen entlasten, Informationen müssen nur 1mal pub'd
Wenn Ereignisrate >> Bedarf und wenige Senken vorhanden, Senken (Broker) überlastet
Zuverlässigkeit MQTT Level 2 ≙ CoAP CON

ZigBee
Ziel: Standards zur Realisierung von Anwendungen mit Überwachungs- und Steuerungsaufgaben. Einheitliche, hersteller übergreifende Schnittstellen. Ziele: Geringer Energiebedarf
Annahme: niedriges Datenaufkommen

ZigBee



Anwendungs-Rahmenwerk
Teil der Anwendungsschicht
Unterteilung der Anwendungen in 240 Objekte
IDs definierbar. Vorgesehene Standard IDs können vom Hersteller überschrieben werden
Anwendungsprofile bestehen aus {Geräten}, {Clustern}, {Attributen}, {Befehlen}, Beschreibung, welche Cluster von welchen Geräten benötigt werden, Funktionale Beschreibung für jedes Gerät.
Client-Server Modell: Attribute sind (typischer Weise) Servern zugeordnet
Client: Lichtschalter Command: toggle Server: Lampe
Cluster: {Befehle, Attribute}, Jeweils eine Rolle den Clients/Servern zugeordnet
Gerät: Enthält Funktional-Beschreibung über bereitgestellte Funktion
Diese ist über die Cluster ID identifiziert
Endpunkt: Analog zum Port im „normalen" Internet. 255 max.
Anwendungsunterstützung (Schnittstelle zur Netzwerkschicht)
Zuverlässiger Transportdienst auf 802.15.4
Verbindungslos, Fragmentierung (dann ACKs), Duplikaterkennung
L2 übernimmt: Fehlererkennung, keine Reihenfolgetreue, keine Staukontrolle, keine Flusskontrolle
ACKs bei Fragmentierung ⇒ Flusskontrolle, Reihenfolgetreue
Kein Piggy-Packing
Sendewiederholung: bei Timeout (max. 3)
bei Empfang „unerwarteter" Quittung
Fragmentierung: Quittungen pro Fenster (1-8 Fragmente), per Bitmaske kann jedes Fragment separat ge-ACK-t werden
ACK wird gesendet: bei vollständigem Empfand des Fensters
Ende der Fragmentierung
Timeout (dann Bitmaske 0 → NACK)
Binding: Verknüpfung zweier Endpunkte, gespeichert in

Zentralem Binding-Cache-Server
⇒ EP sendet ohne Adresse kennen zu brauchen
Multicast über Zuordnung eines EPs an mehrere Gruppen in einer Gruppentabelle. Gruppen ID → EP-Zuordnung
Netzwerkschicht
Übernimmt Routing, Adressierung (16 bit-Adressen), Rollenzuweisung, Netzwerkmanagement
Rollen: Koordinator (FFD), max. 1 pro Netz, wählt Kanal, beeinflusst Topologie, Verwaltet Netz
Router (FFD): kann Dateneinheiten weiterleiten
Endgerät (FFD oder RFD)
Unterstützte Topologien: Stern, Baum, Mesh (ohne Routing)
Baum: Jeder Router hat eigenen Adressbereich
Parameter: max_depth, max_childs, max_child_routers
Adressen werden direkt zum Routing genutzt
Router kennen Distanz zum Koordinator & Parameter des Adressbaums
Ziel in Teilbaum? Weiterleitung „down", ansonsten „up"
+ keine Routingtabellen
+ keine Zustandshaltung,
+ keine Pfadsuche
- Ausfallanfällig
- suboptimale Pfade
- kein Router mit freien Child-slots in Reichweite ⇒ kein Zutritt
- Bottleneck Koordinator
Weitere Eigenschaften: 802.15.4 Beacon-Mode unterstützt
Mesh: Routing notwendig, Adresskollision müssen vermieden werden
Nutzung von Ad-hoc on-demand distance Vector Routing (AODV)
Fluten von Routenanfragen, Pfad zur Quelle wird etabliert
Endgeräte nicht darum beteiligt

+ Robust
+ Selbstheilend
+ bessere Pfade
- kein Beacon Modus möglich ⇒ permanenter Betrieb erforderlich
- Zustandshaltung

Für P2MP-Kommunikation nutzbar, da günstige als n-fache Pfadaufbau

Multicast Routing für Gruppenverwaltung

Sender ist Mitglied der Zielgruppe

Zieladresse ist Broadcastadresse

Max-hop Limit, nicht-Gruppenmitglieder leiten Nachricht weiter

Sender ist nicht Mitglied der Zielgruppe

Finde Route zu einem Mitglied (Anycast), Routing über flutenden Router

dann Broadcast wie oben

ZigBee IP : ZigBee SE 2.0 → HTTP/TLS → TCP → RPL → 6LoWPAN → 802.15.4

Security: Single-Mission key auf Netzwerkschicht

Auf Anwendungsschicht: nur für E2E Kommunikation (Unicast)

Schlüssel werden im Klartext einmalig übertragen !!

6TiSCH

IPv6 E2E Konnektivität für Industrial IoE

Determinismus, Zuverlässigkeit, Energieeffizienz

Medienzugriff: 802.15.4e (time slotted channel hopping (TSCH))

Upper layers also 6LoWPAN

6top (6TiSCH operation sublayer) besteht aus

scheduling-Funktionen

6top Protocol (6p)

Nachbar-Erkennung, Zeitpläne erstellen & verwalten

Multi-hop-Pfade von RPL auf Zeitpläne abbilden

Auf Topologieänderungen reagieren

6TiSCH-Netz: drahtgebunden, schneller Backbone

Hohe Anforderung an Synchronisation

6P

Implementiert 6TiSCH Kommunikation

erstellen, löschen, verschieben von Zellen im TSCH Schedule

6P-Transaktion: Aushandlung der Befehle

Wichtig: Konsistenz!

TSN (Time sensitive Networking)

6TiSCH ermöglicht Nutzung robuster & planbarer Medienzugriffsmechanismen (802.15.4e TSCH) für WPANs

Konzept: keine Verluste von Paketen

niedrige Latenz

explizite Reservierung von Ressourcen

⇒ keine Überlast ⇒ keine Puffer nötig ⇒ niedrige Latenz

Leistungsgarantie für reservierte Datenströme

Abgeschlossenes Teilnetz

Anderer Datenverkehr nur mit Best-effort-Garantie

⇒ Stream-Konzept (continuous flow of data, low latency, jitter, ...)

„Talker" sendet Talker Advertisement Message, enthält Stream Deskriptor (Bandbreite, Akkumulierte Latenz)

Router addieren ihre Latenz hinzu, leiten über Links weiter, die noch genügend Kapazität frei haben

⇒ Geräte bei denen der Stream verfügbar ist, sehen das Advertisement & Latenz

Zusätzliche Nachrichten für Fehler, Beenden o.Ä.

Ingress Policy

Verwerfen von nicht-reserviertem Traffic

Traffic wird auf #Frames oder #Bytes / pro Zeit reserviert

⇒ Bursting möglich

Rahmen werden bei Verletzung der Vereinbarung verworfen

Traffic Shaping (Priorisieren & formen von Echtzeittraffic)
- Viele Warteschlangen pro Egressport im Switch
- Priorisiert ↑ Credit-Based, Burst ⇒ Stream
- Time-aware shaper sendet immer zu deterministischen Zeiten
- Frame Preemption: Pausieren von Best-Effort Rahmen

DetNeT
- L3 Mechanismus für verbundene, heterogene Geräte
- Unterstützung von "more deterministic Flows"
- Höhere Ausfall~~rate~~sicherheit / Robustheit / Zuverlässigkeit durch Replikation (→ dann Elimination) über redundante Pfade
- Keine Verluste durch Staukontrolle
- Zeitsynchronisation im Netz
- Absolute Garantie für min/max E2E Latenz, Jitter
- Sehr geringe Paketverlustrate (auch nicht durch Staukontrolle oder Verkehrsdrosselung)
- Flows können durch Datenrate oder wiederkehrende Schedules gekennzeichnet sein
- Verfahren zur Ressourcenreservierung nötig
- Service Layer: Redundante Pfade
- Transport Layer: Garantien für Rate & Jitter
- Flows: TSN-Streams nutzen mindestens eine der beiden Layer
- Nodes: Transit-Node: DetNet ↔ DetNet
  - Relay-Node: Transport / Service-Layer (Pakete erzeugen / eliminieren)
  - Edge-Node: TSN ↔ DetNet

LoRa & LoRaWAN
- Netzarchitektur für dedizierte IoE-Netze
- SemTech LoRa: Phys. Schicht von LoRaWAN
- Topologie: Stern aus Sternen
- LoRa-Funkverbindung zwischen Endgerät und Gateway

Gateway leitet Daten an Network Server weiter
Network Server: Schicht 2 Endpunkt der Verbindung zum Endgerät
Konfiguriert Endgeräte
Logik auf nachgelagerten Application Server
Schnittstelle zwischen Netzinfrastruktur & Application Server
MAC-Protokoll ermöglicht Abwägung zwischen Energiebedarf & Latenz
Class A: Endsystem initiiert jede Kommunikation, unsynchronisiert & unkontrolliert, zufällige Kanalwahl. Gateway hat dann möglichkeit, Daten auszutauschen (pending-Bit), muss auf allen Kanälen permanent lauschen
Bei Gatewayausfall ~~dürfen~~ sollten die Endgeräte nicht die gleiche Retry-Strategie verfolgen. Randomisierte Sendezeiten statt exponentiellem Backoff
Class B: zusätzliche Zeitfenster für Kommunikation zum Endgerät
sog. Ping-Slots
Network-Server sendet Beacon zur Zeitsynchronisation
Zuweisung von Ping-Slots vorkonfiguriert, zeitliche Versatz zwischen Slots random errechnet aus Geräteadresse & Beacon-Zeitstempel => Kollisionen weitestgehend vermieden
Class C: dauerhafte Verbindung
Mobile Geräte müssen Positionsänderung bekannt geben, wenn dadurch das Gateway gewechselt wird

Sicherheit

Kryptografische Bausteine
Verschlüsselung (symmetrisch oder asymmetrisch)
Integritätssicherung (Hashfunktion, Digitale Signatur)
Schlüsselaustausch

Einsatz von Kryptografie ist teuer
Schlüsselmaterial, Krypto-Code, Rechnungen...
Symmetrische Kryptografie „günstiger" als Asymmetrische
Hardware-offloading meist effizienter, wenn Krypto-
verfahren komplex ist. Ansonsten überwiegt die Initialisierung,
Kommunikation und die Konfiguration des Koprozessors
Schlüsselaustausch bei Ressourcenknappheit, selbstorganisierten
und ändernden Netztopologie
Zentrale Certificate Authorities oft nicht gegeben/erreichbar
Diffie-Hellman zu teuer (> 2 Sekunden)
Single-Mission-Key
Symmetrischer Schlüssel, in jedem System vor Ausbringung fest
verankert, wird genutzt für jegliche Kommunikation
Problem: 1 korrumpierter Knoten ⇒ alles unsicher
Paarweise Schlüssel ⇒ Problem mit Speicherbedarf
⇒ Problem mit neu hinzugekommenen Knoten
⇒ Problem mit Schlüsselverteilung
Zufallsverteilte Schlüssel (EGLI)
Pool von Schlüsseln P
Systeme bekommen zufälligen Ausschnitt R von P
⇒ Systeme haben wahrscheinlich gleiche Schlüssel ⇒ gut
falls nicht, baue Schlüsselpfad über Key-Ring
Erkennung gleicher Schlüssel: senden von Klartext-Liste UNSICHER!
senden von zufälligen Klartext + Chiffrat
⇒ Empfänger kodiert empfangenen Klartext und vergleicht
Ergebnis mit empfangenem Chiffrat ⇒ TEUER!
Angreifermodell: Angreifer kann Schlüsselliste auslesen
mit bereits 1%-5% der Systeme kann er >50% der
Verbindungen abhören. Traffic von „sicheren" Knoten geht

Über unsicheren Pfad ⇒ auch unsicher!
⇒ Schlüsselaustausch, mehrere Pfade, mehrere Verbindungen mit gleichem Schlüssel ... UNSICHER!
Aber: keine weitere Infrastruktur nötig, Austauschwkeit zwischen Knoten hoch, Trade-off zwischen Sicherheit & Speicher

Key-Infection
Bei Abwurf aus dem Flugzeug anwendbar, da Angreifer höchstwahrscheinlich (noch) nicht präsent
Keine Infrastruktur nötig (Keyserver)
Knoten ziehen zufälligen Schlüssel und broadcasten ihn.
Empfangene Keys werden zum Sitzungsschlüssel verrechnet und gemeinsam mit eigener Knoten ID zurückgesendet
Optional: Multihop-Key-Exchange, Secrecy Amplification (Mehrwege-Austausch)
Vorteile: nur genutzte Schlüssel werden gespeichert
pro Schlüsselaustausch 2 Nachrichten
Nachteile: Schlüsselaustäusche meist gleichzeitig ⇒ Kollisionen
Links müssen symmetrisch sein
Annahme eines abgeschwächten Angreifermodells

Sicherheit in IEEE 802.15.4-2015
Optionaler Schutz gegen Wiedereinspielen, Schutz von Integrität & Vertraulichkeit. Anwendbar auf alle Dateneinheiten außer L2 ACKs
Verwendet symmetrische Verfahren, Key Exchange nicht spezifiziert
Bis zu 255 Schlüssel, ausgewählt per Adresse oder expliziter Indizierung.
Verwendung für Unicast oder Gruppenkommunikation
AES-CCM: Stromchiffre
Chiffrat = Schlüsselstrom XOR Klartext, Schlüsselstrom = Nonce AES Key

Probleme bei der Schlüsselverwaltung & -verwendung

Nonce & Schlüssel dürfen nicht sequenzgenau wiederholt werden

Nonce besteht aus Zähler => kann sich wiederholen

Zähler pro Schlüssel => begrenzt Lebensdauer eines Schlüssels

Zähler resetten sich bei Stromausfall ...

TLS / DTLS

E2E Verschlüsselung, bewährt im Internet

Konfiguration & Kontrolle auf Anwendungsebene

Komplexer & umfangreicher Standard => teuer!

DTLS: besser geeignet für drahtlose Netze, im IoE

DICE: DTLS In Constrained Environments

TLS: Authentifizierung, Schlüsselaustausch, Aushandlungen, Umschalten auf kryp. gesicherte Kommunikation, Fehler-Handling, Durchreichen der Anwendungsdaten, Schutz von Vertraulichkeit, Integrität & der Daten

Warum DTLS: TLS setzt TCP voraus, im IoE aber öf UDP in Betrieb

DTLS = TLS + Paketverlust handling im Handshake-Protokoll + Fragmentierung + Reihenfolgetreue + Erkennen von Paketverlusten

Keine Verwendung von Stromchiffren vorgesehen (da wäre die Reihenfolge kritisch notwendig!)

DICE: Zuschneiden von DTLS auf IoE / IoT * gängige Szenarien:

Authentifikation zweier Kommunikationsendpunkte

Integrität & Authentizität der Daten

Vertraulichkeit

Implementierbarkeit auf ressourcenarmen Geräten

Einige wenige effiziente kryp. Algorithmen unterstützt

Einschränkung der zu unterstützenden Authentifizierungsmechanismen

" der Protokollerweiterungen & Optionen (bspw. Sitzungswiederaufnahme)

DICE nutzt AES, SHA-256, asymmetrische Kryptografie auf Basis von elliptischen Kurven
effizienter implementierbar, geringere Schlüssellänge erforderlich
Authentifizierung:
via vorverteilten Schlüsseln (symmetrisch): geringer Rechen- und Kommunikationsaufwand
Vorverteilung von Krypto-Hardware (TPM) oder in Firmware
Raw Public Keys
Jedes System besitzt asymmetrisches Schlüsselpaar
Nicht Teil des Zertifikates, Authentifikation nicht Teil des Schlüsselaustauschs. Vorverteilung in kryp. Hardware oder in der Firmware
Zertifikate
Nutzung von Diffie Hellmann auf Basis von elliptischen Kurven
Minimale X.509 Zertifikate auf Basis von ECDSA
Zertifikatvalidierung bzw. Prüfung auf Widerruf kann ausgelagert werden
Security by Delegation
Selbst mit DICE: asymmetrische Kryptographie ist teuer
Gebe Sicherheitsfunktionalität an vertrauenswürdige Basisstation weiter
+ Einsatz starker Kryptographie
- Zusätzliche Kommunikation
- Basisstation muss vertraut werden
Delegated Authentication & Authorization Framework (DCAF)
Delegationsmechanismus für CoAP
Client Authentication Manager (CAM)
Server Authentication Manager (SAM)
Endgeräte mit SAM/CAM verbunden
CAM $\xleftrightarrow{\text{TLS}}$ SAM Verbindung

Endgeräte sind mit SAM/CAM via DTLS/DICE verbunden
Authentifizierung dort via symmetrischen Verfahren
CAM ↔ SAM Authentifizierung via TLS
Client → Server stellt CoAP Anfrage GET bat/status
erhält 401 unauthorized zurück
Client stellt Autorisierungsanfrage an CAM
CAM leitet Anfrage weiter (kann das auch ablehnen) an SAM
SAM erstellt Ticket, das Server verlangt und sendet es an CAM
CAM sendet Client das Ticket
Client sendet Ticket an Server
Server sieht sein Ticket, prüft es und sendet verlangten Wert an Client
Ticket enthält verifier & face zum Prüfen
Face enthält SAI (Gewährte Zugriffsberechtigungen)
Nonce, Timestamp => Schutz gegen Replay
Methode der Kryptographie (hmac_256,...)
Einen symmetrischen Schlüssel für die Client → Server Verbindung

Generell: Angreifer hat mehr Ressourcen
Angriffe: Eavesdropping: Abhören (Passiv)
Aktiv: Injection (Einbringen eigener Dateneinheiten)
Replay (Wiederholen von gesendeten)
Wormhole (Weiterleiten von Signalen)
Intrusion (Einbringen eigener Geräte)
Sybil (Vortäuschen von Geräten)
DoS: Flooding, Jamming (Stören des Mediums)
Overflow (Spamming, absichtlicher Pufferüberlauf)
Routing: Sinkhole (selektives Unterdrücken/Weiterleiten von Routing-Informationen)
Manipulation von Metriken und Topologien

Angriffe auf die Hardware

Passiv: Extraktion von Schlüsselmaterial erreichbar durch:

- Auswerten von elektromagnetischen Signalen
- Abhören von akustischen Signalen.
- Aufzeichnen des Stromverbrauchs
- Beobachten des Laufzeitverhaltens

Aktiv: Klonen von Hardware

- Reprogrammierung
- Auslesen von Speicher
- Vorspielen von falschen Sensor-/Eingabedaten

BMW Connected Drive

Nutzt Modem zum Senden/Empfangen Daten von/zum GPRS/EDGE Mast- AES und HMAC_SHA256 verwendet => was aus dem Modem rauskommt, ist sicher. Aber nicht, was rein geht. Angriff zwischen Hauptprozessor und Modem. Einsicht: Schlüsselmaterial im Modem fest einprogrammiert, leicht zu extrahieren und für alle Fahrzeuge gleich => Single Mission Key

Angreifer kann nun Befehle an das Auto senden: unlock();

Auto öffnet die Türverriegelung

BMW Connected Drive kann deaktiviert werden, und per BMW Connected Drive wieder eingeschaltet werden

Keine Authentifizierung in der Connected Drive Cloud

Mit Zugriff auf die Hardware leicht angreifbar

Philips Hue

ZigBee - Verschlüsselung

1 Netzwerkschlüssel für alle Geräte im PAN

Fehler in Protokollimplementierung erlaubt zurücksetzen aller Geräte in Reichweite => Neubeitritt in Netzwerk & Schlüsselaustausch