Berühren statt klicken

Arno Puder

Langsam nimmt die NFC-Technik Fahrt auf: Android-Smartphones haben passende Hardware, Google liefert die APIs und eine Anwendung. Zeit also für einen Blick auf die Technik und ihre praktische Nutzung.

Oberflächlich betrachtet ist NFC (Near Field Communication) ein weiterer Standard zum drahtlosen Übertragen von Daten. Im Unterschied zu WiFi oder Bluetooth stand bei seiner Einführung jedoch der einfache Umgang aus Benutzersicht im Vordergrund. Während WiFi und Bluetooth zunächst einer mehr oder weniger aufwendigen Konfiguration bedürfen, funktioniert NFC sofort.

Die Technik gibt es bereits seit mehr als 15 Jahren, sie steckt unter anderem in Kreditkarten, Reisepässen und Zugangskarten. In allen Anwendungsszenarien ist das Ziel die Unterstützung von 0-Klick-Interaktionen: Der Benutzer muss nicht mehr umständlich eine Aktion mit Mausklicks oder Tastenkommandos auslösen. Allein durch das Annähern eines NFC-Geräts an ein anderes drückt er seine Absicht aus. Sei es das kontaktlose Bezahlen in einem Geschäft, wo er ein Smartphone an ein Bezahlterminal hält, oder das Einmalticket mit integriertem NFC-Chip, das ein Terminal beim Annähern entwertet. Die Funktechnik soll eine Brücke zwischen der physischen und der virtuellen Welt schlagen: Das physische Heranhalten eines NFC-Tags löst eine virtuelle Reaktion aus.

Aus dieser Motivation leiten sich die Eigenschaften der Technik ab. Anders als bei RFID ist dabei die Kommunikation nur über sehr kurze Distanzen bis maximal 10 cm möglich, typisch sind 1 bis 4 cm. Die Übertragungsgeschwindigkeit liegt mit bis zu 848 Kbps pro Sekunde verhältnismäßig niedrig. Dafür ist ein großer Vorteil von NFC der reibungslose Verbindungsaufbau. Im Unterschied zu Bluetooth, bei dem man Sender und Empfänger zunächst paaren muss, reicht hier das bloße Annähern.

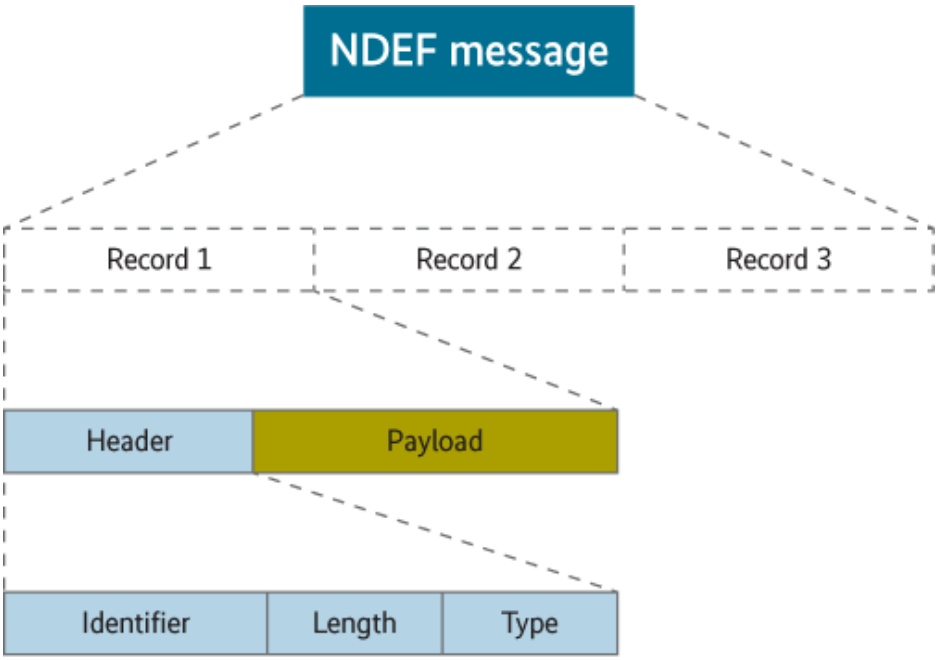
Aktive und passive Teilnehmer

NFC unterscheidet zwischen Kartenlesern und sogenannten Tags. Ein Kartenleser ist aktiv in dem Sinne, dass er ein RF-Feld (Radio Frequency) für die Datenübertragung aufbaut. Ein NFC-Tag hingegen ist passiv und verfügt typischerweise nicht über eine externe Energiequelle, sondern nutzt das Induktionsfeld des Kartenlesers. Deshalb ist die Antenne passiver Tags ähnlich einer Spule gewickelt, da sie nicht nur kommuniziert, sondern auch die notwendige Spannung erzeugt. Wegen ihrer kompakten Bauweise lassen sich passive Tags nahezu überall integrieren.

Die Standardisierung geht auf eine Initiative von Nokia, Sony und Philips (heute NXP) im Jahr 2004 zurück. Sie schlossen sich zum NFC-Forum zusammen, dem mittlerweile über 170 Firmen angehören – Finanzinstitute neben Hardwareherstellern und Produzenten von Consumer-Elektronik. Das Ziel des NFC-Forums ist die Standardisierung von Verfahren und Datenformaten. Bei passiven Tags unterscheidet es vier Typen, die hinsichtlich der Frequenzcharakteristika, der Übertragungsprotokolle und der Handhabung von Kollisionen differieren. Prinzipiell sind alle Tag-Typen beschreibbar. Kommandos können Tags der Typen 1 und 2 jederzeit auf Nur-Lesen umstellen, während dies bei den Typen 3 und 4 lediglich der Hersteller ab Werk vermag. Außerdem gibt es nicht standardisierte proprietäre Tag-Typen.

Aus Entwicklersicht ist einer der wesentlichen Standards des NFC-Forums das NDEF (NFC Data Exchange Format). Es beschreibt die während eines Kommunikationsvorgangs übertragene PDU (Protocol Data Unit). NDEF-Nachrichten erleichtern die Zusammenarbeit verschiedener Geräte.

Austausch zwischen Gleichberechtigten



NFC-Nachrichten bestehen aus Records, deren Typ bei Tags die für die Verarbeitung zuständige Anwendung bestimmt (Abb. 1).

Eine NDEF-Nachricht besteht aus einem oder mehreren Records (s. Abb. 1). Jeder davon verfügt über einige Standardelemente wie den Typ, einen Identifier und die eigentlichen Daten. Der Typ eines NDEF-Records beschreibt den Inhalt der Nutzlast. Der Typ *URI* etwa legt fest, dass die Nutzlast aus einer absoluten URI nach RFC 3986 besteht. In der Regel wählt der Typ des NDEF-Records die Anwendung aus, die auf die Nachricht eines Tags reagiert. Besteht sie aus mehreren Records, entscheidet normalerweise der Typ des ersten über die zuständige Anwendung. Herstellerspezifische MIME-Typen erlauben eine eindeutige Zuordnung zwischen Tags und der Applikation.

NFC unterstützt drei unterschiedliche Betriebsmodi für den Datenaustausch. Zunächst kann ein Kartenleser einen Tag lesen und in manchen Fällen auch Daten darauf schreiben. Der Abstand zwischen Tag und Leser muss so kurz sein, dass sie eine Verbindung aufbauen können. Ein typisches Szenario ist ein Smart-Poster mit einem integrierten NFC-Tag. Er enthält eine URI für weiterführende Informationen. Hält man ein geeignetes Smartphone an den Tag, liest es die URI aus und lädt sie im Browser.

Ein weiterer Betriebsmodus ist der sogenannte Peer-to-Peer-Betrieb, bei dem zwei Kartenleser Daten austauschen. Ein typisches Anwendungsszenario dafür ist der Austausch von Kontaktdaten. Hält man beispielsweise zwei Android-Geräte aneinander, während man auf dem einen einen Eintrag in der Adressverwaltung betrachtet, werden automatisch diese Kontaktdaten als VCard übertragen.

Der letzte Betriebsmodus ist die Card-Emulation: Ein Kartenleser emuliert einen NFC-Tag, sodass ein ein anderer Leser darauf zugreifen kann. Googles elektronisches Bezahlsystem Wallet arbeitet nach diesem Prinzip. Läuft die Wallet-App auf einem Android-Gerät, emuliert es einen Tag, auf den ein POS-Terminal (Point of Sale) in einem Geschäft zugreift. Dafür wäre zwar auch der Peer-to-Peer-Modus geeignet, aber bei der Card-Emulation kann dasselbe POS-Terminal ebenfalls eine Kreditkarte mit integriertem NFC-Tag als Zahlungsmittel akzeptieren.

Als ein Beispiel für einen passiven Tag sei im Folgenden der MIFARE Ultralight vorgestellt. Er stammt von NXP Semiconductors, ist vom Typ 2 und erfüllt die Anforderungen der ISO 14443-A. Der Tag lässt sich kostengünstig herstellen und findet häufig für Einmaltickets Verwendung. Auf ihm lassen sich 64 Bytes speichern, die in 16 Seiten zu je 4 Bytes segmentiert sind. Daten überträgt er mit 106 Kbps, was eine typische Transaktion unter 35 ms hält. Das EEPROM hält die Daten bis zu 5 Jahren und erlaubt bis zu 10 000 Schreibzyklen.

Speicheraufteilung

Page address		Byte number			
Decimal	Hex	0	1	2	3
0	00h	serial number			
1	01h	serial number			
2	02h	serial number	internal	lock bytes	lock bytes
3	03h	OTP	OTP	OTP	OTP
4 to 15	04h to 0Fh	user memory			

Speicheraufteilung

48 der 64 Bytes sind frei verfügbar, der Rest ist für bestimmte Funktionen reserviert (s. Abb. 2). Von den ersten 9 Bytes stehen 7 für die Seriennummer gemäß ISO 14443-3 zur Verfügung, zwei weitere enthalten eine Prüfsumme. Die zwei Lock-Bytes regeln den Schreibzugriff. Jedes ihrer Bits repräsentiert eine der 16 Speicherseiten. Ist eins gesetzt, lässt sich der korrespondierende Speicherbereich nur noch lesen. Ein einmal gesetztes Lock-Bit kann nicht mehr gelöscht werden. Somit ist es möglich, den Speicher ganz oder teilweise für das Schreiben zu blockieren.

APIs für Java, Android, Web

Neben den Lock-Bytes bietet die MIFARE Ultralight einen 4 Byte großen OTP-Zähler (One Time Programmable). Im Auslieferungszustand ist er auf 0 gesetzt. Wurde eins seiner Bits einmal gesetzt, ist es ähnlich wie bei den Lock-Bits nicht zu ändern. Der Wert des OTP-Zählers kann sich somit maximal 31-mal ändern. Über das Setzen von Lock-Bits oder das Ändern des OTP-Zählers entscheidet die Anwendung auf der Seite des Kartenlesers. Mithilfe der Lock-Bytes und des OTP-Zählers kann sie einen Tag nach einmaliger Benutzung entwerten.

Für den Zugriff auf NFC-Geräte existieren verschiedene APIs. Bereits 2006 erschien eine Java-API für Mobilgeräte als JSR 257. Eine Programmierschnittstelle für C und C++ gibt es vom *libnfc*-Projekt. Das W3C wiederum ist momentan damit beschäftigt, eine HTML5-kompatible NFC-API zu standardisieren (s. „Alle Links“). Momentan dürfte die Android-API am weitesten verbreitet sein. Googles Mobilbetriebssystem führte den Umgang mit NFC-Geräten bereits mit Version 2 („Gingerbread“) ein. Die Schnittstelle unterstützt das Lesen und Schreiben sowie den Peer-to-Peer Modus, nicht jedoch die Card-Emulation. Wallet ist als Google-eigene Anwendung eine Ausnahme. Im App-Store Google Play gibt es unter anderem Apps von NXP Semiconductors, deren Chips in vielen Android-Smartphones stecken. Die frei verfügbaren Anwendungen erlauben das Lesen, Schreiben und Analysieren passiver Tags.

Android bindet NFC mit seinem Intent-System ein: Erkennt der Kernel einen Tag, schickt er einen Intent, den Anwendungen verarbeiten können. Aus Sicherheitsgründen passiert dies nur, wenn der Bildschirm eines Android-Geräts eingeschaltet und entsperrt ist. Das Auslesen eines NFC-Tags unter Android gelingt mit wenigen Programmierkenntnissen, wie das folgende Beispiel zum Lesen des Speichers im MIFARE Ultralight zeigt. Zunächst muss eine Android-Anwendung in der Datei *AndroidManifest.xml* den Benutzer um die Erlaubnis bitten, auf den NFC-Chip zugreifen zu dürfen:

```
<manifest ...>
   <uses-permission android:name="android.permission.NFC" />
 </manifest>
```

In der Datei *res/xml/tech_list.xml* ist die unterstützte NFC-Hardware aufzuführen:

```
<resources>
   <tech-list>
        <tech>android.nfc.tech.MifareUltralight</tech>
    </tech-list>
 </resources>
```

Liefert Android einen NFC-Intent an die App aus, kann sie den Speicher des Tags auslesen (s. Listing). Danach enthält das Array *tagData* die Daten, die man mit der Klasse *MifareUltralight* ebenso wieder

schreiben könnte, wobei sich die Seriennummer und zuvor gesetzte Bits der Lock-Bytes und des OTP-Zählers nicht ändern würden.

Insbesondere in Bezug auf die Privatsphäre und die Sicherheit werfen die interessanten Anwendungsmöglichkeiten kritische Fragen auf. Befürwortern der Technik gelten die geringen Kommunikationsdistanzen als ein wichtiges Argument für eine geschützte Privatsphäre, da anders als bei RFID Angriffe nur aus kurzem Abstand möglich seien. Android schaltet zudem den NFC-Chip zusammen mit dem Bildschirm aus, und Anwendungen Dritter dürfen auf der Plattform den Chip nur benutzen, wenn sie im Vordergrund laufen. Die Kommunikation zwischen NFC-Geräten ist allerdings nicht verschlüsselt, sodass eine Anwendung eigene Sicherheitsmechanismen verwenden sollte. Neuere Chips bieten einige kryptografische Operationen an, um dies zu vereinfachen.

Ausblick

Obwohl NFC in einigen Anwendungen wie Einmaltickets bereits erfolgreich funktioniert, war die Technik bisher kaum verbreitet. Nokia verbaute 2006 als erster Handy-Hersteller NFC-Chips in seinen Geräten. Doch erst mit dem Siegeszug der Smartphones scheint die Zeit reif für eine weitere Verbreitung zu sein. Google forciert die Funktechnik in Android und versucht gerade mit seinem Wallet ein neues Bezahlsystem zu etablieren.

Überhaupt könnte das kontaktlose Bezahlen eine wichtige Anwendung für die Near Field Communication zu sein. ISIS, ein Konsortium aus AT&T, T-Mobile USA und Verizon Wireless, versucht allerdings bislang erfolglos ein mobiles NFC-Bezahlsystem in Nordamerika zu etablieren. T-Mobile Polen bietet seit letztem Jahr ein solches Bezahlverfahren an, und in Deutschland will die Telekom in diesem Jahr nachziehen. Apple hingegen wartet erst einmal ab. Trotz Spekulationen vor dem Verkaufsstart hat das aktuelle iPhone 5 keinen NFC-Chip. Es scheint jedoch klar, dass Smartphones eine ideale Plattform für NFC-Anwendungen bieten. (ck)

Arno Puder
lehrt an der San Francisco State University und hilft Mozilla bei der Integration von NFC in Firefox OS.

Alle Links: www.ix.de/ix1303126

Listing: Auslesen des Speichers eines NFC-Tags

```
class MyActivity extends Activity {
   // ...
  @Override
  public void onResume() {
    super.onResume();
    Intent intent = getIntent();
    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
      Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
      MifareUltralight mifare = MifareUltralight.get(tag);
      byte[] tagData = new byte[64];
      int k = 0;
      for (int p = 0; p < 4; p++) {
        byte[] payload = mifare.readPages(p * 4);
        for (int i = 0; i < payload.length; i++) {
          tagData[k++] = payload[i];
        }
      }
      // tagData enthält die 64 Bytes des Ultralight tags
    }
  }
}
```