NFC on Linux

Samuel Ortiz
Intel Open Source Technology Center

November 5th, 2012
Agenda

- NFC basics
- NFC open source stacks
- The Linux NFC stack
- One example
NFC basics
Near Field Communication

- A short range (< 5cm) wireless technology.
- Low throughput (< 500 kbps).
- Low cost.
- Not Bluetooth, not RFID.
- Partly standardized by the NFC Forum.
- “Tap-to-share” NDEFs.
- NFC tags and NFC devices.
Three NFC modes

- **Reader**
  - One device reads a tag.

- **Peer to peer**
  - Two devices talk to each others

- **Card emulation**
  - One device pretends to be a tag
Use cases

- Very wide...
- Data exchange.
  - Playlists, URLs, business cards...
- Connection Handover.
  - Simplified Bluetooth pairing
- Payments, loyalty cards.
- Ticketing.
- Security, access control.
  - Key-less rental cars
NFC Open Source Stacks
• The Android bounty.
• Android as the single supported platform.
• No kernel support for NFC.
• No standard Linux distribution support.
Two stacks, same issues

- Two Android stacks.
  - libnfc-nxp, opennfc.
- 100% userspace, ad hoc kernel interface.
- Exclusive HW support.
  - NXP pn544, INSIDE microread: HCI only.
- No community, no source code repositories.
- Exclusive support, no visibility.
  - Google, INSIDE.
Other stacks

• nfcpy
  • Nice implementation, 100% python.
  • Sony sponsored.
  • No HCI or NCI support.

• libnfc
  • Academic project, LGPL licensed.
  • Only USB and UART devices supported.
  • Missing features.
  • SVN repository, community.
The Linux NFC stack
Yet another stack?

- HW independence.
- NFC for non Android platforms.
- POSIX NFC APIs.
- Kernel/User space split.
- Consistent behavior and APIs.
- Open development process.
The Linux NFC stack

- The official NFC Linux kernel stack.
- Maintained by Intel.
- Hosted on git.kernel.org.
- GPLv2 licensed.
- 1.5 year old.
- Split between kernel and user spaces.
- Open development.
The overall picture

Kernel space

User space

NFC Hardware

NFC Drivers

NFC Core

AF_NFC Sockets

NFC Netlink

neard

D-Bus API

application
Kernel Architecture

- Netlink socket
- PROTO_LLCP
- PROTO_RAW
- LLCP
- AF_NFC
- NFC Core
- NCI
- HCI
- SHDLC
- pn533
- Wilink
- µread
- pn544
The NFC daemon

- Tag specific handling (R/W).
- Transport protocols on top of LLCP.
- Adapter and targets management.
- NDEF parsing.
- Handover.
- D-Bus APIs.
- Plugin based.
- GLib and libnl dependency.
## Hardware and Features Support

<table>
<thead>
<tr>
<th></th>
<th>Interfaces</th>
<th>Tag R/W</th>
<th>LLCP</th>
<th>Handover</th>
<th>Card Emulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linux</strong></td>
<td>HCI, NCI, USB</td>
<td>Yes</td>
<td>SNEP, NPP</td>
<td>Bluetooth</td>
<td>No</td>
</tr>
<tr>
<td><strong>Android</strong></td>
<td>HCI</td>
<td>Yes</td>
<td>SNEP, NPP</td>
<td>Bluetooth</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Inside Secure</strong></td>
<td>HCI</td>
<td>Yes</td>
<td>SNEP</td>
<td>Bluetooth, WiFi</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>libnfc</strong></td>
<td>USB, UART</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>nfcpy</strong></td>
<td>USB</td>
<td>Yes</td>
<td>SNEP</td>
<td>Bluetooth</td>
<td>No</td>
</tr>
</tbody>
</table>

1 PN532 not supported yet
Plans

• Short term
  • Secure Element and card emulation netlink API.
  • Improve MIFARE support.
  • Inside Secure microread support.

• Long term
  • Wi-Fi Handover
  • OBEX and IP over NFC.
  • Personal Health Device Communication.
  • libneard.
One Example - PHDC
Personal Health Device Communication

- Medical and fitness devices.
- IEEE 11073 APDUs.
- NFC as a carrier.
- LLCP
  - APDUs over LLCP.
- Reader/Writer
  - PHD NDEFs.
PHDC over LLCP with neard

Implemented as plugins/phdc.c

```c
struct near_p2p_driver phdc_driver = {
    .name = "PHDC",
    .service_name = "urn:nfc:sn:phds",
    .read = phdc_read,
    .push = NULL,
    .close = phdc_close,
};

near_p2p_register(&phdc_driver);

```
PHDC raw implementation: Device

```c
struct sockaddr_nfc_llcp addr;

fd = socket(AF_NFC, SOCK_STREAM, NFC_SOCKPROTO_LLCP);

addr.sa_family = AF_NFC;
addr.dev_idx = adapter_idx;
addr.nfc_protocol = NFC_PROTO_NFC_DEP;
addr.service_name = "urn:nfc:sn:phds"

bind(fd, (struct sockaddr *) &addr, sizeof(addr));
```
PHDC raw implementation: Manager

```c
struct sockaddr_nfc_llcp addr;

fd = socket(AF_NFC, SOCK_STREAM, NFC_SOCKPROTO_LLCP);

addr.sa_family = AF_NFC;
addr.dev_idx = adapter_idx;
addr.target_idx = target_idx;
addr.nfc_protocol = NFC_PROTO_NFC_DEP;
addr.service_name = "urn:nfc:sn:phds"

connect(fd, (struct sockaddr *) &addr, sizeof(addr));
```
Questions ?

- NFC daemon
  http://git.kernel.org/?p=network/nfc/neard.git;a=summary
- NFC kernel
  http://git.kernel.org/pub/scm/linux/kernel/git/sameo/nfc-3.0.git
- Web site
  • https://www.01.org/linux-nfc
- Mailing list
  https://lists.01.org/mailman/listinfo/linux-nfc
- sameo@linux.intel.com