Freie Universität Berlin

Network stack refactoring and its model

Martine Lenders

FU Berlin
Institut für Informatik
RIOT

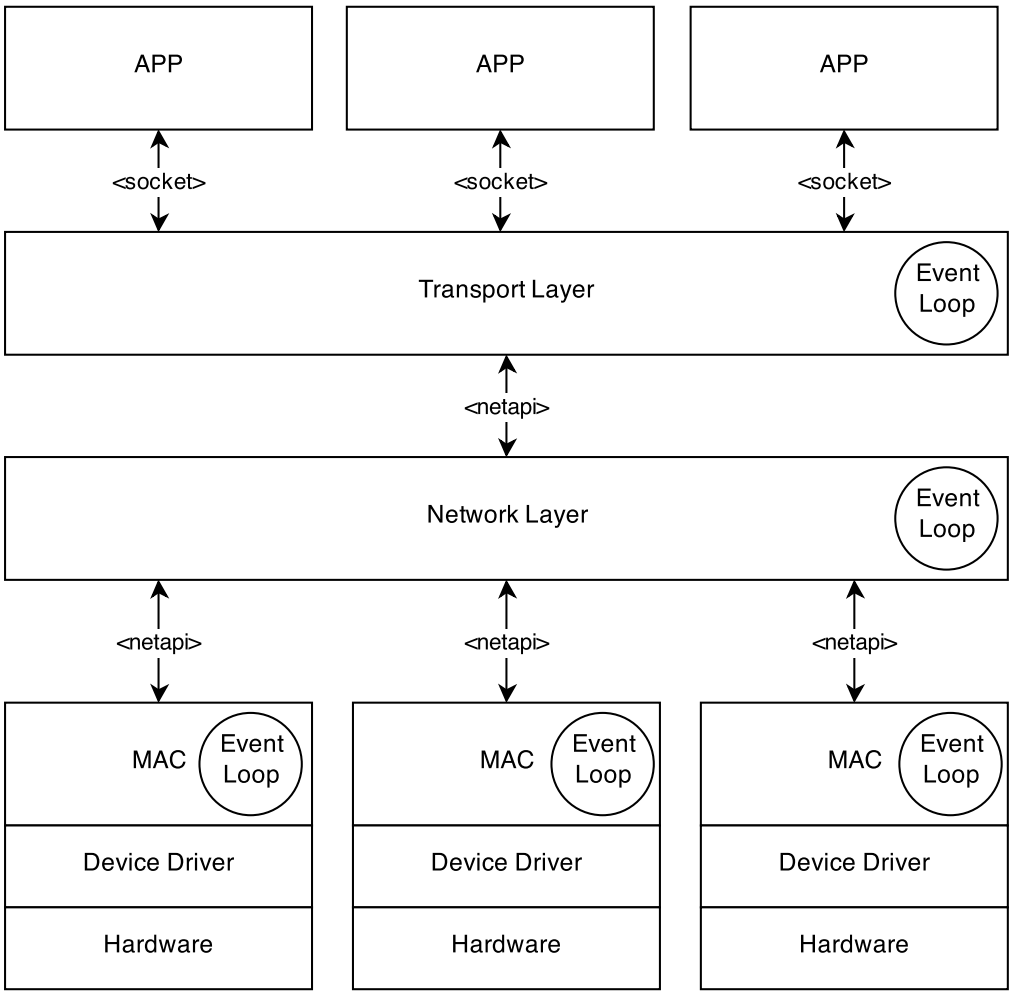
January 6, 2015

Motivation

Problems with the old network stack

- Too big (buffers everywhere, multiple reimplementations of same stuff)
- Too inconsistant (every protocol has its own set of APIs)
- Too monolithic
  - originally designed for just 6LoWPAN over cc110x
  - IEEE 802.15.4 support patched in with advent of at85rf231/cc2420 support
  - every new device type requires heavy patching
  - IPv6 without 6LoWPAN currently impossible
- Transceiver API does not scale (for every new device new `#ifdef` branch)
- Context (thread) of functions calls not always clear

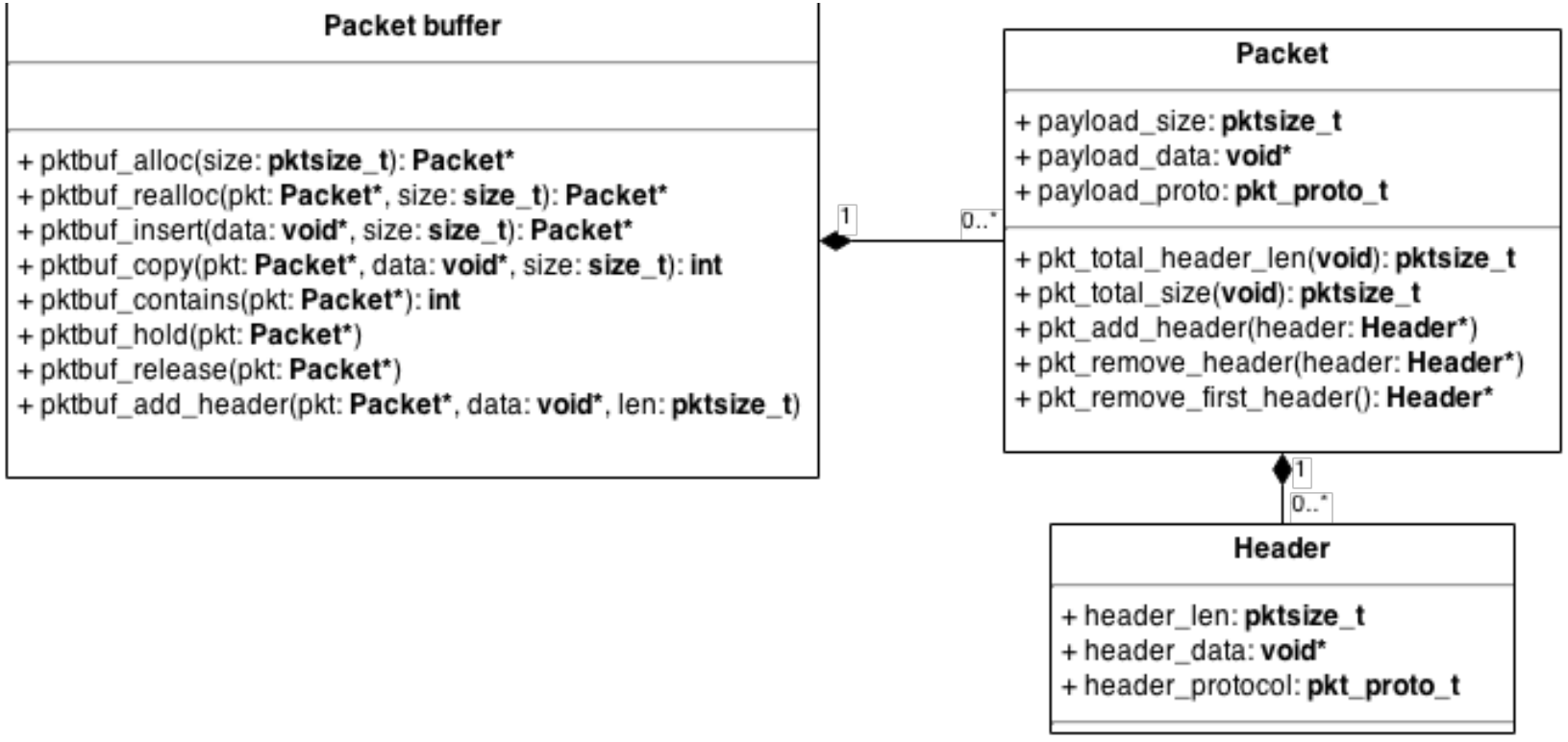
Basic idea


APP
APP
APP
<socket>
<socket>
<socket>
Transport Layer
Event Loop
<netapi>
Network Layer
Event Loop
<netapi>
<netapi>
<netapi>
MAC
Event Loop
MAC
Event Loop
MAC
Event Loop
Device Driver
Device Driver
Device Driver
Hardware
Hardware
Hardware

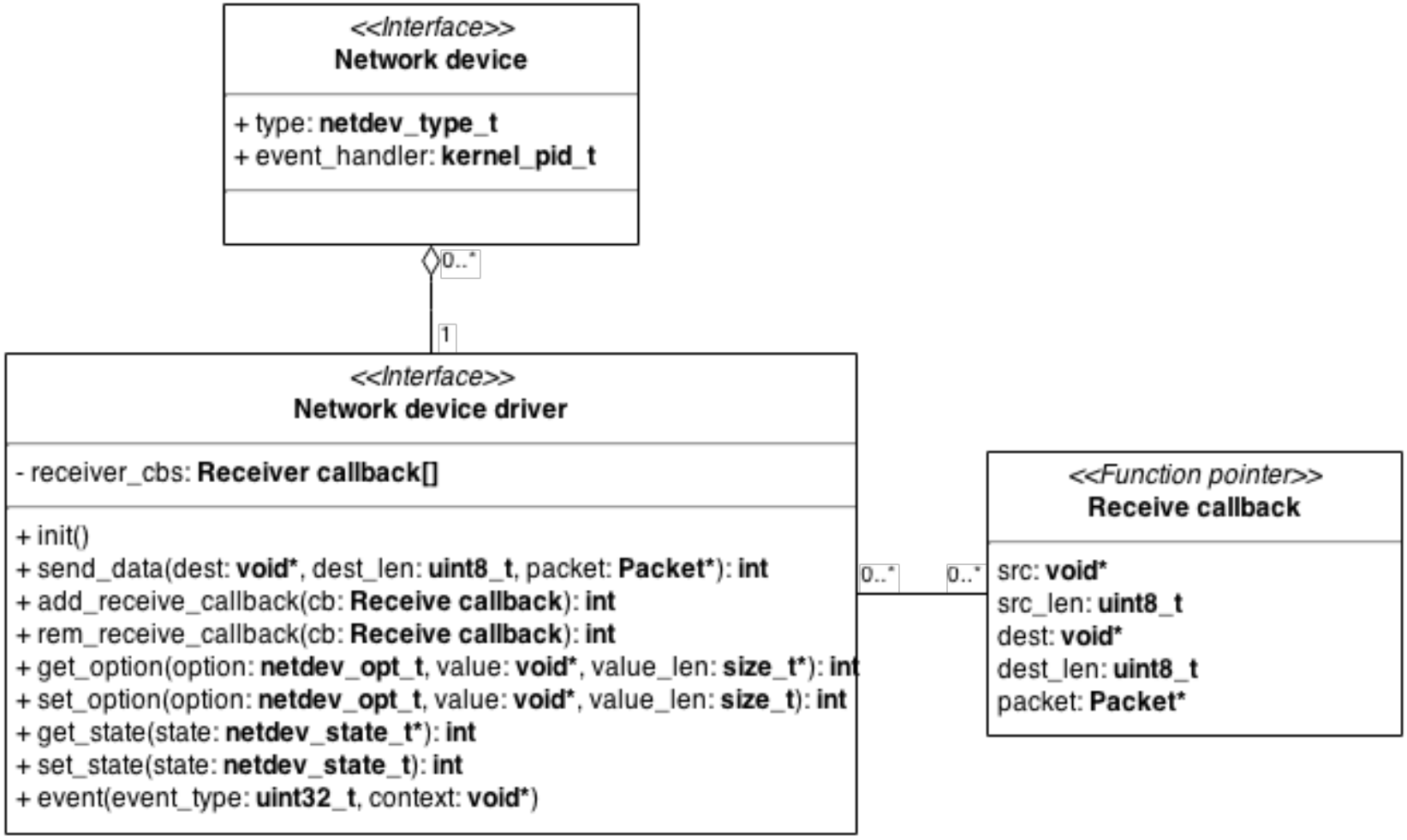
Prelimanary reasoning

- Use RIOT's scheduler to priorities network layers $\Rightarrow$ netapi as IPC API
- Device and MAC layer need low-latency communication (e.g. IEEE 802.15.4) $\Rightarrow$ MAC and device driver can't communicate via netapi $\Rightarrow$ netdev (more or less netapi in function-based)
- Central packet buffer $\Rightarrow$ pktbuf (get thread-safe packet buffers from central array)
- Reduce numbers of memory movings $\Rightarrow$ protocol headers as linked list in pktbuf

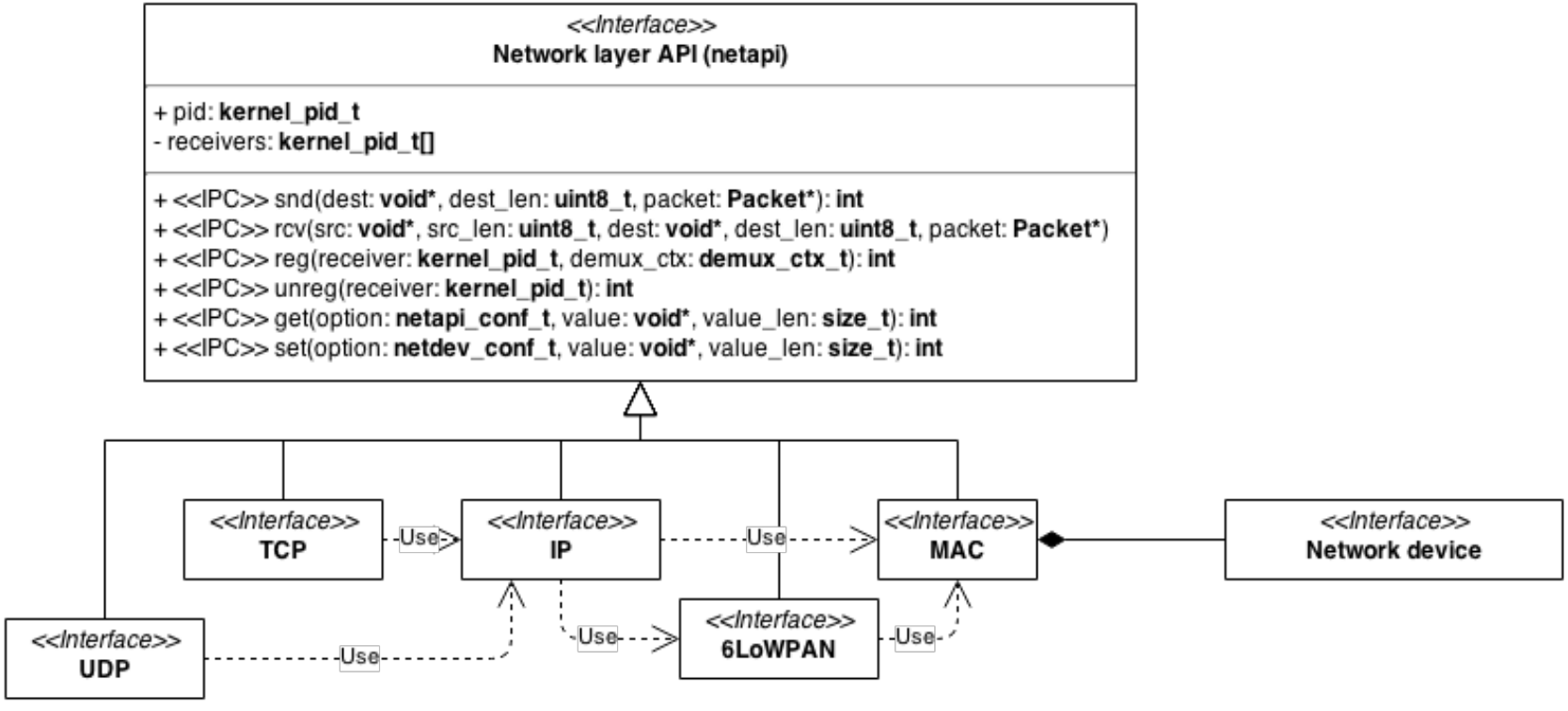
Class diagram: pktbuf



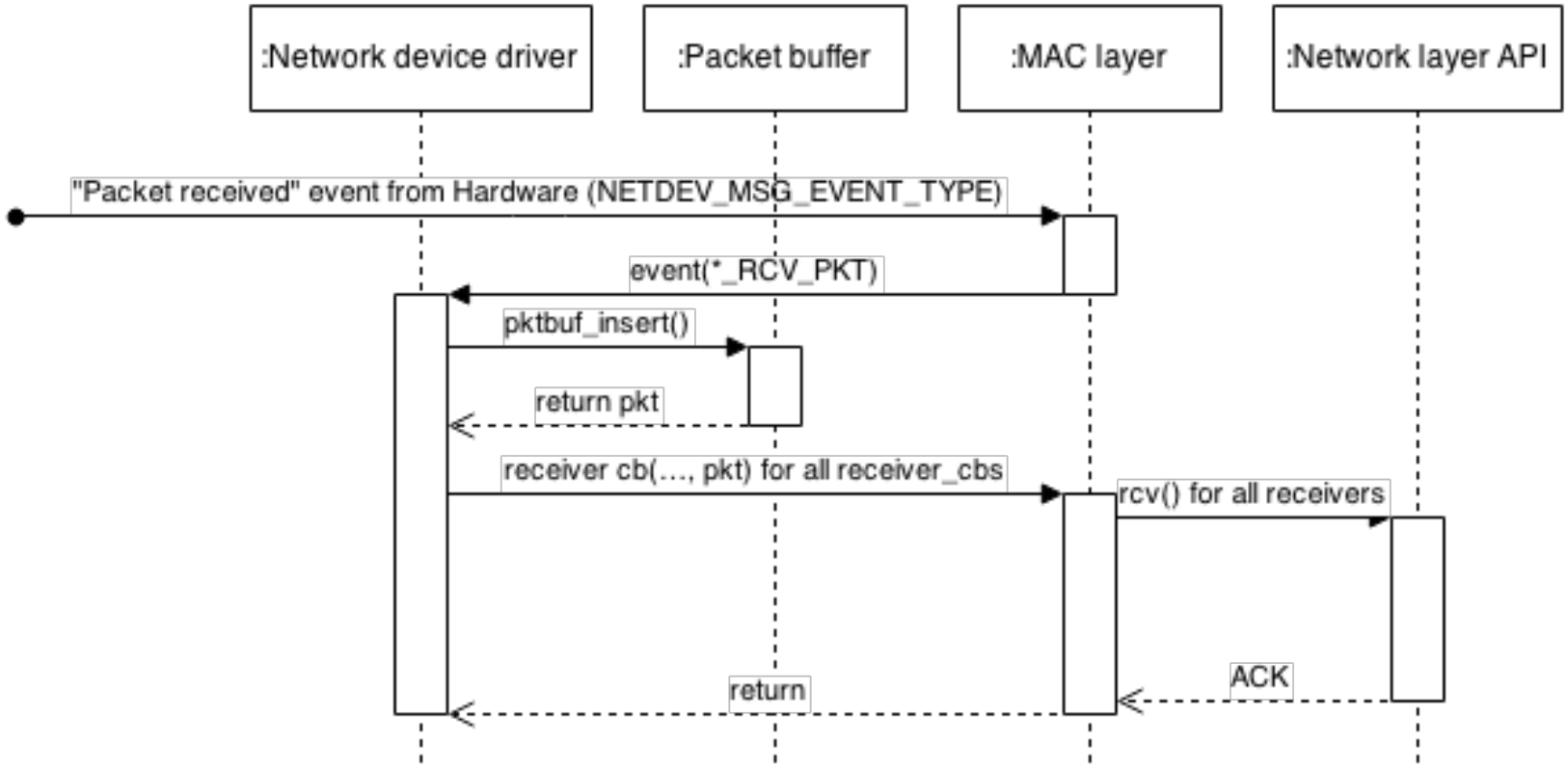
Class diagram: netdev



Class diagram: netapi



Sequence: Getting receive context out of ISR



Preliminary status report (aka PRs exists or already in master)

- pktbuf ✓
- netdev ✓ (ports exists for all devices in master)
- netapi
  - MAC (✓) (nomac, as simple forwarding MAC layer, more complex MAC adaptions welcome)
  - 6LoWPAN ✓
  - IPv6 (nearly done, though on halt due to pktbuf changes)
  - Transport layer (TCP/UDP) (not porting efforts as of yet)