Supporting Radio Clocks in OpenBSD

Marc Balmer <mbalmer@openbsd.org>

micro systems/OpenBSD

BSDConTr, Istanbul, Turkey

Agenda

Trivia

- Every computer has a clock device, but they drift
- Computer clocks can be sychronized using NTP
- GPS and Time signal stations can provide an external reference time, e.g. where there is no net to reach any NTP servers.
- OpenNTPD can be used to serve the received time on a local net.

Time Signal Stations

- Transmit time information using longwave radio
- Different coding schemes, but similar principles
- We currently support DCF77 (.de) and HBG (.ch)
- MSF (.uk), will be added
- JJY (.jp), will be added on my next trip to .jp (sponsors?)

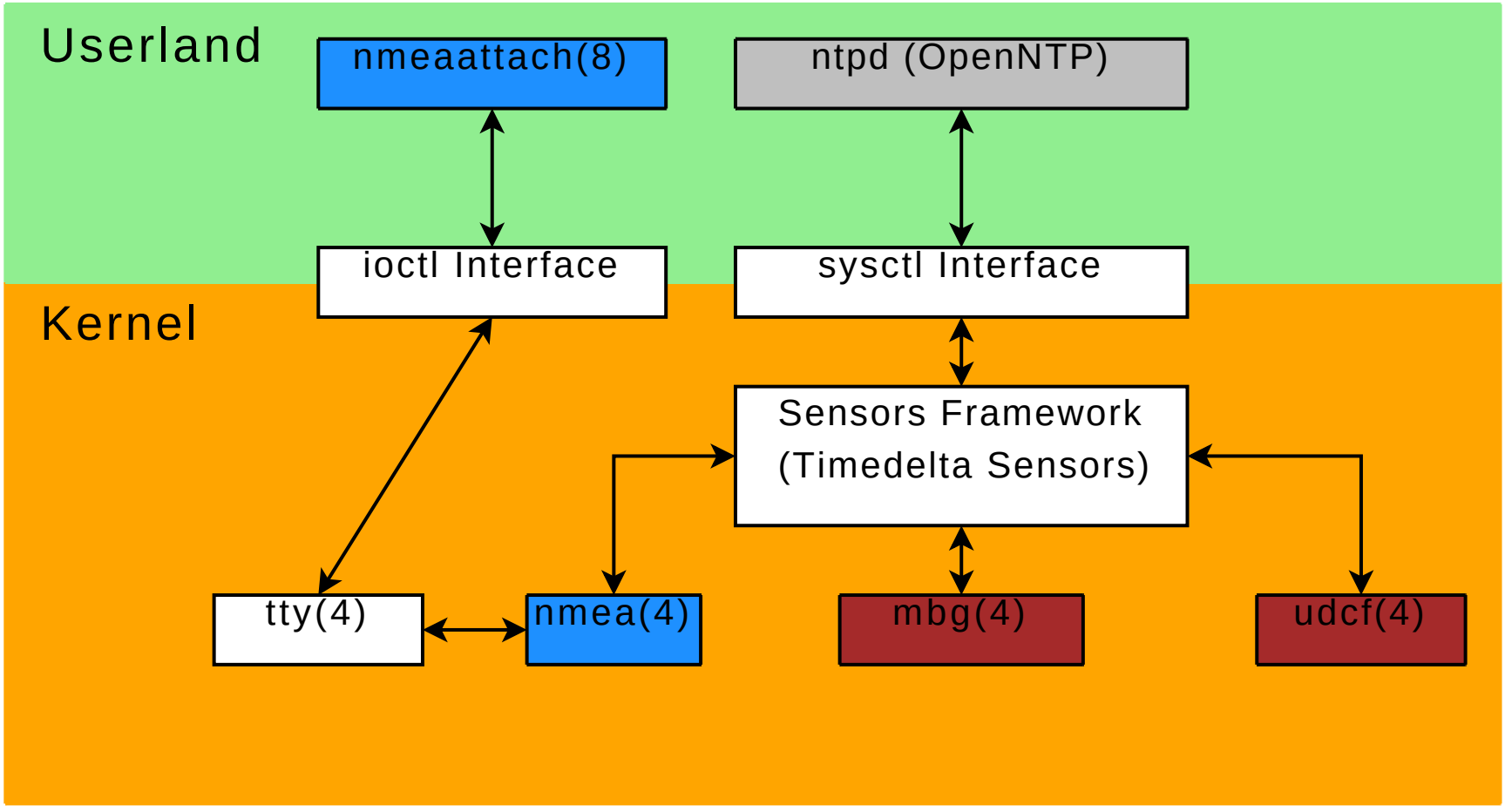
Global Positioning System GPS

- 3D Localization based on calculation with precise time information
- Time information can be extracted to get precise time
- We are only interested in time information, not the position information

Time Signal Stations vs. GPS

- GPS
  - Available worldwide
  - Good receiving conditions (skyview) needed
  - No rule w/o exception: Some Swiss made receivers work inside building (using signal reflections)
- Time Signal Stations
  - Work inside buildings, even in the basement
  - Not available everywere

Overall Architecture



Representing Time as the Error of the Local Clock

- Not the absolute time received, but the difference between the system time and the received time is provided (i.e. the error of the local clock)
- The purpose of the driver is to calculate this offset
- This timedelta can be accessed through the sensors framework

Sensors

- Sensors provide operational status of devices like temperatures, fan speeds, etc.
- Sensors are read-only values that can be queried using the sysctl mechanism: sysctl hw.sensors
- Sensors have a name, description, value, a state, and a timestamp
- **Timedelta sensors indicate the error of the system time in nanoseconds**

Line Disciplines

- Are a set of functions that are called from tty code
- Can receive every character received by the tty
- Can look at and manipulate the data stream
- Are set using an ioctl(2) on the device
- A well known line discipline besides TERMIOS is SLIP

The Mother Tongue of GPS Devices: NMEA 0183

- National Marine Electronics Association
- An ASCII protocol to transmit navigation data
  - Date, time, position, speed, course, etc.
- Spilled out by most GPS units
- Easy to parse record format
  - Each record starts with '$' and extends to the CR/LF
  - No longer than 82 characters
  - Comma separated values
- Many GPS units also speak proprietary binary formats

nmea(4), a Line Discipline to Decode NMEA 0183

- A tty line discipline that provides a timedelta sensor
- Decode the GPRMC message to get time information
- Indicates GPS warning status in the sensor status
- Can make use of PPS signals through tty timestamping
- For highest possible precision, use PPS

nmea(4), Decoding Time from NMEA Sentences

- Look at the received NMEA stream and decode the GPRMC message
- All NMEA devices support the GPRMC message
- Calculate the offset between the local time and the time received
- **But: When was the time just received valid?**

GPS devices with PPS signal

- PPS = Pulse Per Second, a higly precise signal triggered at the start of each second, typically 1us precision
- The GPRMC message following the PPS indicates time at the moment the PPS occured
- **tty timestamping has been added for correlate the GPRMC message to a point in time**

tty Timestamps Tell When the Received Time was Valid

- PPS signal has to be wired to a serial port's CTS or DCD line
- When the PPS even occurs, the current system time is immediately copied to a buffer by the tty device
- nmea(4) copies the timestamp at the next possible occasion
- Timestamping can be done on the raising or falling edge
- Keep in mind: It is precise, but OpenBSD is not a real-time operating system

Timestamps Without a PPS signal

- Cheap GPS device or USB attached ones don't provide a PPS signal
- Soft timestamp is taken on the first '$' of the first message at the beginning of a second:

$GPABC,...	take timstamp at '$' character
$GPDEF,...	more messages
$GPRMC,...	decode time here
...	more messages & a longer gap

- Beginning of a second is determined by measuring the time between consecutive sentences (called "the gap" in the source)

Something completely different: udcf(4)

- Device driver that provides a timedelta sensor
- Decode the DCF77 and HBG timecode
- Indicate the timecode (DCF77 or HBG) in the sensor description and the quality of the timedelta in the sensor status

Details of the DCF77 Timecode (used by HBG, too)

Bits	0-14	Weather Information (encrypted)
Bit	15	Call bit
Bit	16	Announcement of DST change
Bit	17-18	Indication of DST
Bit	19	Announcement of a leap second
Bit	20	Start of encoded time information
Bits	21-27	Minute
Bit	28	Parity
Bits	29-34	Hour
Bit	35	Parity
Bits	36-41	Day of month
Bits	42-44	Day of week
Bits	45-49	Month
Bits	50-57	Year
Bit	58	Parity

Receiving the Timecode

- Bits are transmitted by reducing the amplitude of the
- signal to about 1/3 at the start of each second
- Convert length of reduction to 1's and 0's
  logical 0 low-level pulse of 100 ms duration
  logical 1 low-level pulse of 200 ms duration

Basic Time Decoding Algorithm

- Synchronize: Wait for a delay between two bits > 1.5 sec
- Next bit is bit 0
- Pick up all 59 bits
- Decode bits
- Make time valid at next start of a second

udcf(4) Implementation Details

- On attach power up the receiver
- Set up timeouts to decode the bit stream
- Start of a second is detected in a fast loop (defines overall "precision")
- On detach stop all timeouts

Decoding using timeout(9) (reset at the start of every second)

T1	150 ms	Bit detection
T2	250 ms	Clock type detection
T3	800 ms	Restart fast polling
T4	1500 ms	Detect the minute gap
T5	3000 ms	Detect signal loss

Update of the Sensor Value

- Time information received is decoded and veryfied in the T4 (minute gap) timeout routine
- The sensor value is updated at the start of the next second

Limitations udcf(4)

- Polling over USB severely impacts precision
- Longwave transmission results in a small position dependend offset, this offset changes during the day
- The propagation delay is a non-issues for us since the supported receivers have a precision of 30ms - 70ms
- This is a best effort driver, precision is around 100ms at best
- **Still enough precision for most applications**

mbg(4), PCI attached Industrial Receivers

- PCI based receiver cards for the DCF77 time signal station and GPS
- Very high precision
- Supports the following cards:
  - PCI32, 5V DCF77 receiver
  - PCI511, 3.3V/5V DCF77 reveiver
  - GPS170PCI, 3.3V/5V 6-channel GPS receiver

splhigh();
getnanotime();
write command to card $\leftarrow$ *card takes timestamp*
splx();
later: read timestamp from card

OpenNTPD and Timedelta Sensors

- ntpd was extended to support timedelta sensors
- The new keyword is **sensor** <**name**> or simply **sensor ***
- ntpd recognizes new sensors at runtime and uses them

Digression: Setting the System Time

- No problem when done at system startup
- Serious problems may occur when changing the time at runtime
- Programs might rely on time intervals or on absolute time

Changing the Time at Runtime

- Time as a continuum: Compress or stretch the time axis but never loose a time unit

  adjtime(2)

- Time as a sequence of time units with fixed length: Insert ot delete time units but don't stretch or compress the time axis

  settimeofday(2)

Use adjtime(2) or settimeofday(2) ?

- Consider the type of the software that is running
- cron(8): Needs all time units to start programs
- Other software might rely on accuracy of the tickrate
- If both types run: Changing the time yields unpredictable results. The system time should better not be adjusted at runtime
- But keep in mind: OpenBSD is not a realtime OS

Summary, New Concepts

- Novel way in getting at the time and making it available in the operating systems
- Timedelta sensors as an abstraction of the radio-clock and a uniform way to get at precise time, independent of the device being used
- Soft- and hard (PPS) timestamping for GPS
- Received time is validated before it is used

Summary, Clock Device Support

- udcf(4)
  - Almost no hardware dependency (other than having USB)
  - Limited precision (as polling over USB is needed)
  - Support for MSF (.uk) is planned
- nmea(4)
  - Works with almost all GPS receivers, serial-, USB-, CF-, and PCMCIA-attached
  - Highest possible precision with PPS capable devices, requires special wiring
- mbg(4)
  - High precision
  - Industrial grade receivers

Acknowledgments

- Meinberg Funkuhren donated several PCI based GPS and time signal station receiver cards for the development of mbg(4)
- Gude ADS donated several Expert Mouse CLOCK devices for the development of the udcf(4) driver
- The concept of timedelta sensors was an idea of Theo de Raadt who also did the implementation of the tty timestamping code.
- Several OpenBSD users donated radio clocks of any kind to help with time related development, which was much appreciated.
- Many OpenBSD developers helped in various ways, be it by testing the code or by pushing me in the right direction.

Availability

nmea(4) and udcf(4) are included in OpenBSD since the 4.0 release. The newer mbg(4) driver is included in OpenBSD since the 4.1 release.

`http://www.openbsd.org/`

About the Author

After working for Atari Corp. in Switzerland where he was responsible for Unix and Transputer systems, Marc Balmer founded his company micro systems in 1990 which first specialised in real-time operating systems and later Unix. During his studies at the University of Basel, he worked as a part time Unix system administrator.
He led the IT-research department of a large Swiss insurance company and he was a lecturor and member of the board of Hyperwerk, an Institute of the Basel University of Applied Sciences.
Today he fully concentrates on micro systems, which provides custom programming and IT outsourcing services mostly in the Unix environment.
Marc Balmer is an active OpenBSD developer; he was chair of the 2005 EuroBSDCon conference that was held at the University of Basel.

In his spare time he likes to travel, to photograph and to take rides on his motorbike. He is a licensed radioamateur with the call sign HB9SSB.

Contact Information

Marc Balmer
micro systems
Wiesendamm 2a, Postfach
CH-4019 Basel
Switzerland

E-mail	marc@msys.ch, mbalmer@openbsd.org
Company web	http://www.msys.ch/
Personal web	http://www.vnode.ch/