TDMA for Long Distance Wireless Networks

Sam Leffler
sam@errno.com
sleffler@google.com

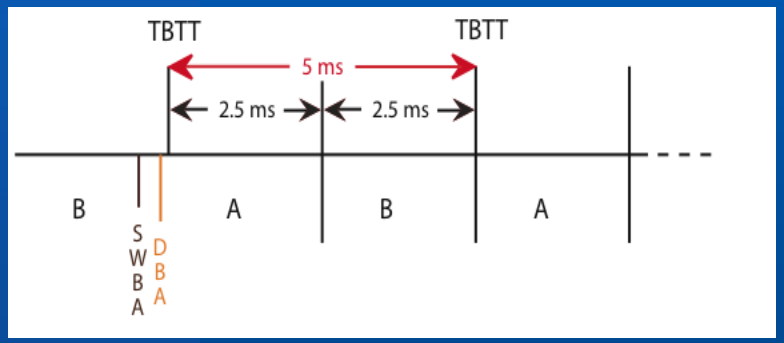
Project Background

- UC Berkeley TIER Project
  - WilDNet
  - TDMA MAC
- Rural Connectivity Platform (RCP)
  - Intel Research Lab in Berkeley, CA
  - FreeBSD

TDMA Motivation

- Distributed Coordination Function (DCF):
  - CSMA/CD (with CCA)
  - 802.11 standard
- ACK frames and “hidden node problem”
- Time Division Multiplexed Access (TDMA):
  - Each station gets a time slice (“slot”)
  - N stations ⇒'s (N) TDMA slots
  - Superframe = N * slot-length

TDMA Configuration



- 2 stations: A, B
- 2.5 ms slot length/time
- 5 ms superframe
- TBTT = Target Beacon Transmit Time

TDMA for Wireless Networks

- Old idea
- Many proprietary solutions, e.g. SkyPilot
- Cellular, WiMAX, HIPERLAN
- We are different: leverage commodity 802.11 parts and do all the hard bits in hardware

FreeBSD Usage/Operation

- Implemented as new VAP type:

```
ifconfig wlan create wlandev ath0 wlanmode tdma
```

- Parameters settable through the CLI:

```
ifconfig wlan0 tdmaslot 0
```

- Works as a p2p bridge out of the box:

```
ifconfig wlan create wlandev ath0 wlanmode tdma \
    ssid freebsd-tdma tdmaslot 0 up
ifconfig bridge create addm wlan0 addm em0 up
```

FreeBSD TDMA Support

- Software structured as two components
- Device-independent 802.11 layer support:
  - Scanning and rendezvous
  - TDMA parameter configuration
  - TDMA slot allocation/assignment
- Device driver support:
  - TDMA slot scheduling
  - TDMA timer/slot synchronization

How Does it Work?

- 802.11 protocol extensions
- 802.11 ACK frames
- Driver support (Atheros):
  - Packet transmit support
  - TDMA slot scheduling
  - TDMA slot synchronization

TDMA 802.11 Protocol

- TDMA operation identified in Beacon frame
  - Capabilities field
  - TDMA Information Element (IE)

TDMA 802.11 Protocol (cont)

- Master/Slave roles for stations
  - Slot 0 is master
  - Slot >0 are slaves
- Master always beacons
- Slave only beacons in response
- Master arbitrates slot allocation

802.11 ACK frames

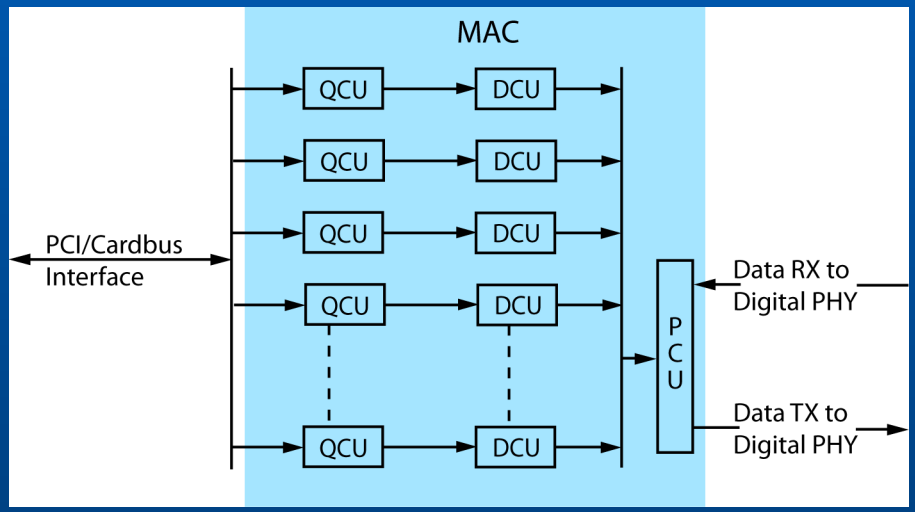
- Do not want h/w generated ACK frames
- Use 802.11 QoS frame format with “no ACK” policy
- Many devices also have a “no ACK” mode

Atheros TDMA Driver Support

- Packet transmit support
  - ath does this in hardware
- TDMA slot scheduling
  - ath does this in hardware
- TDMA timer/slot synchronization
  - ath does this in software
- Atheros driver support is ~350 LOC

Packet Transmit Support

- Follow TDMA schedule:
  - Periodic transmit (h/w frame scheduler)
  - Limit to slot duration (h/w burst duration)
- Issues:
  - Async DMA engine



TDMA Slot Scheduling

- Transmit scheduler driven by hardware beacon timers
- Set beacon interval to “superframe” duration but complications:
  - No “PCU packet kill” on burst duration overflow (add guard time)
  - AR5212 beacon timer granularity 1 TU (round up to multiple of 1024 usec)

TDMA Slot Scheduling (cont)

- Final calculation looks like:

```
tdmabintval = roundup((tdmaslotlen + tdmaguard) * tdma_slotcnt, 1024);
tdmabintval >>= 10;                    /* TSF -> TU */
if (tdmabintval & 1)
        tdmabintval++;
```

TDMA Slot Synchronization

- Avoid TDMA “slot sliding”
- Most systems use a global synchronized clock
  - Requires 3+ machines or external clock source (many research papers, e.g. for sensor nets)
  - Requires explicit handling of propagation delay

FreeBSD Slot Synchronization

- Properties:
  - No global clock
  - Works with 2 stations
  - Hides propagation delay
- Depends on:
  - Wireless device hardware support

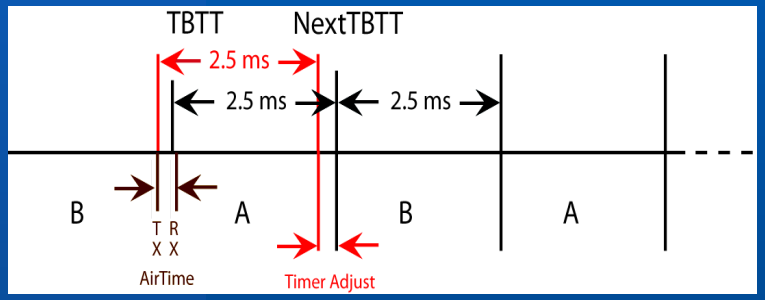
⇒ Limits support to 2 stations (point-to-point)

FreeBSD Slot Synchronization

- Beacon frames are always the first frame in a TDMA slot
- Atheros hardware stamps each received frame with a snapshot of the TSF (local clock)
- Master station runs at known frequency

⇒ Use local clock and known network configuration to calculate NextTBTT on slave from time stamp of beacon frames received from master

TDMA Timer Adjustment



1. Beacon frame RX'd by B at local time T
2. Subtract “air time” of beacon frame to find TX time in local clock
3. Add slot time (including guard time) to find NextTBTT of B
4. Compare NextTBTT against current setting to calculate adjustment

TDMA Timer Adjustment

- Adjusting hardware timers/clock while running is hard:
  - Cannot move clock/timers backwards (hardware will stop)
  - Beacon timers have 1 TU granularity but adjustments are usually a few usecs

⇒ Observation: *clocks do not need to be synchronized, only the difference between clocks*

TDMA Timer Adjustment

- Write TSF directly (works on “recent” Atheros hardware)
- When time moves backwards T, instead add:

  superframe – T
- Do the above carefully
- Reference ath_tdma_update in if_ath.c

TDMA Timer Adjustment

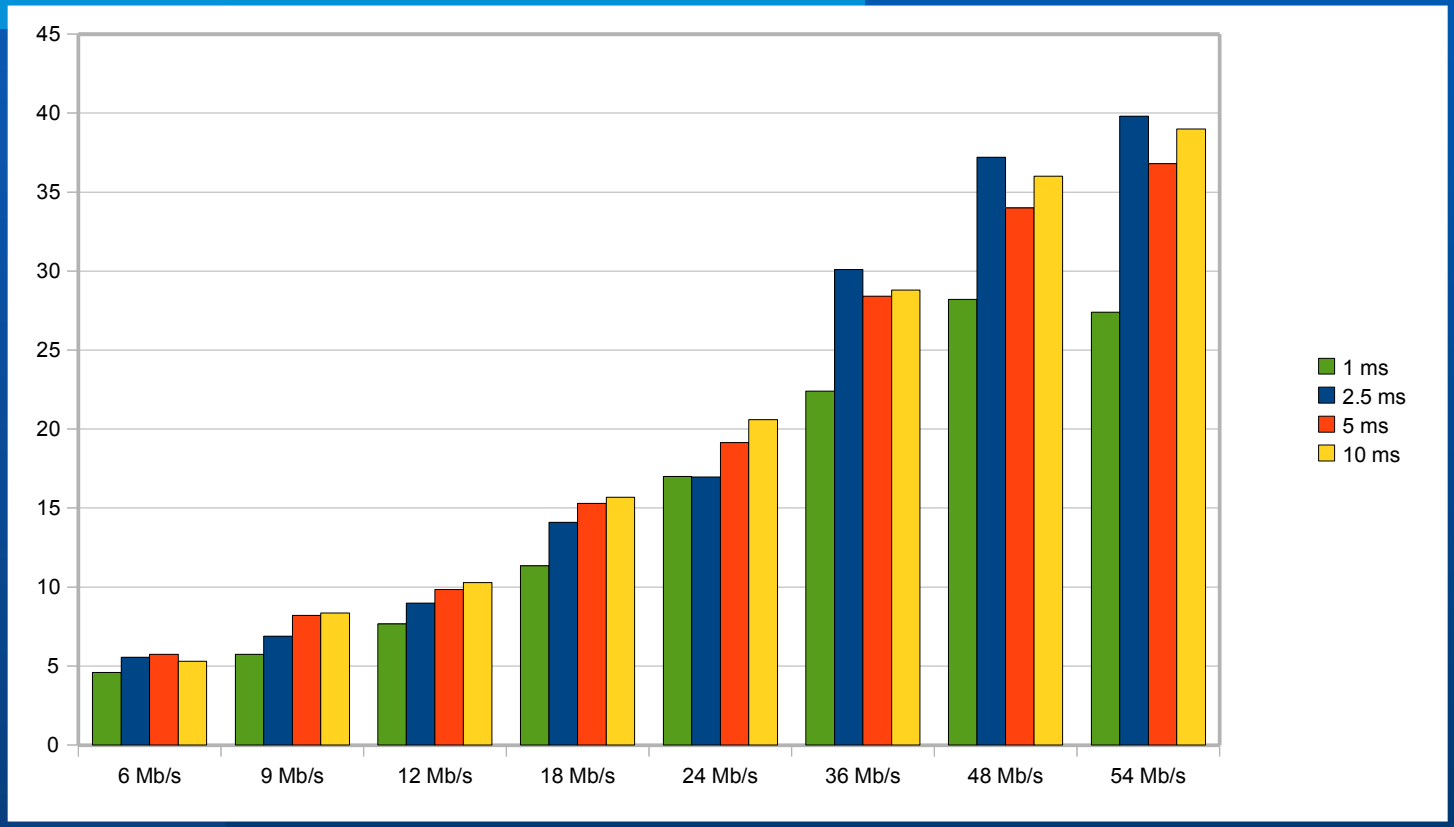
- Typical timer adjustment is <3 usecs (+/-)
- Timers are adjusted only on receipt of beacon frames: beacon schedule can be adjusted to match need
- Debugging problems is hard because inspection affects operation
- Problems are usually manifest through Transmit Override Receive (TOR) errors reported by h/w

Performance

- Expected bandwidth can be calculated from TDMA configuration (frames are burst with SIFS spacing)
- Guard time accounts for most lost bandwidth
- Fixed transmit rate determines maximum bandwidth (dependent on setup and regulatory)
- Rule of thumb:

$$\text{cumulative-bw} = 70\% \quad \text{transmit-rate} \quad (\text{channel-width} / 20)$$

Cumulative B/W on 5180/20 w/ 80dB of pad



Performance Observations

- Minimum usable TDMA slot length ~1ms
- Good slot length is 2.5ms (T1/Voice)
- High transmit rates appear to have a problem (modulation, hardware?)
- Clock jitter in 2.4GHz band is noticeable

Real World Deployments

- La Silla-Cero Tololo 100 km link
- Monitoring: forest fires, oil rigs, etc.
- Many others..

Future Work

- Link auto-configuration:
  - Include TX rate in TDMA IE
  - Test for optimal parameters
- Bulk ACK's:
  - Leverage 802.11n AMPDU packet formats
  - Naturally tied to TDMA slot schedule
  - Slot length must be small enough

Future Work

- Dynamic bandwidth allocation:
  - Easy to dynamically change station slot length
  - Hard to decide how/when to change
  - Reliable protocols (TCP) require inverse BW
- 802.11n parts:
  - MCS transmit rates
  - Higher resolution timers
  - PCU packet kill (eliminates guard interval)

Future Work

- Mobile applications:
  - Can we leverage timer algorithm?
  - How well will handoffs work?