Detection of Rogue APs Using Clock Skews: Does it Really Work?

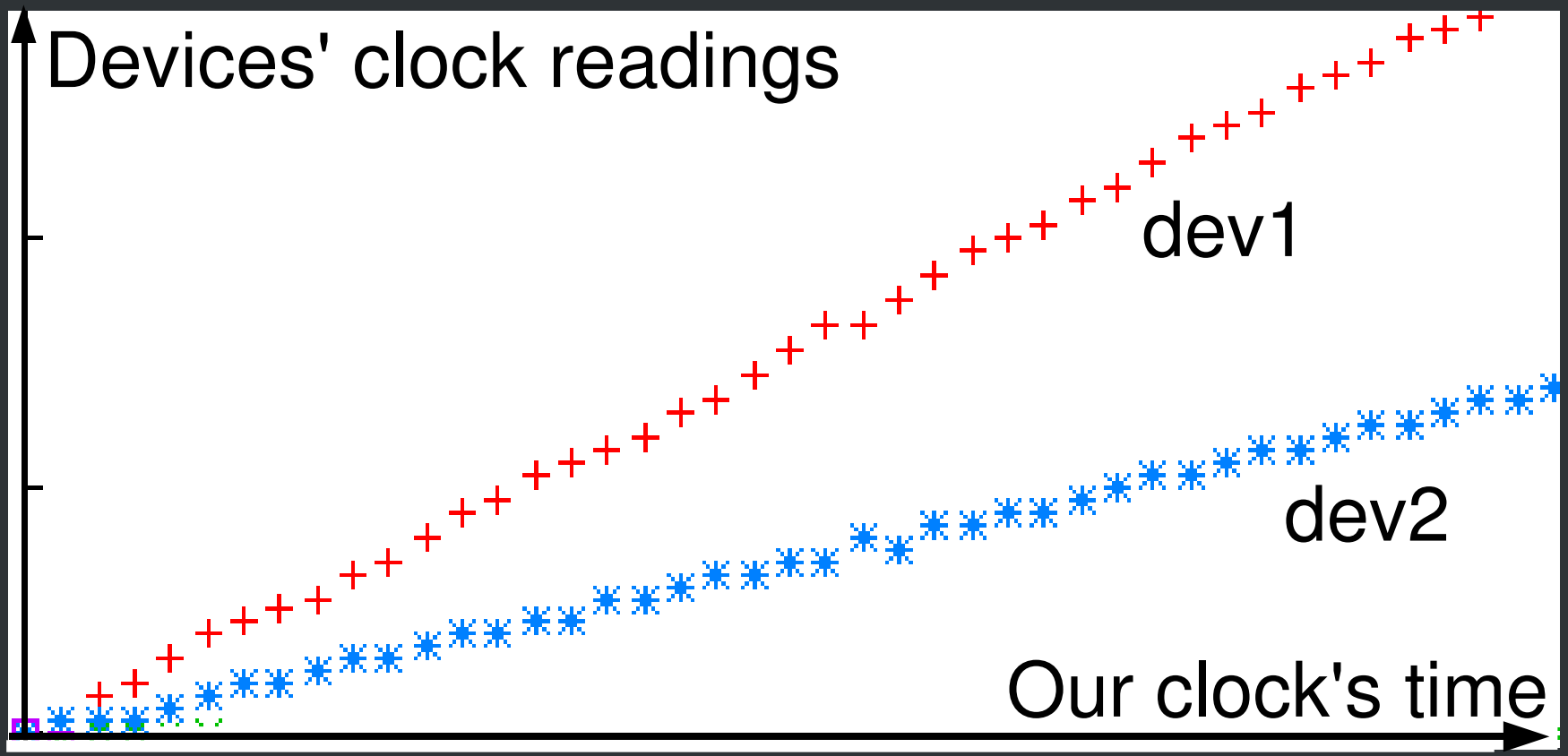
Sergey Bratus Chrisil Arackaparambil Anna Shubina
Dartmouth College

This talk in 30 seconds

- Clock skews are interesting
  - usable for fingerprinting devices (kind of)
  - available for the asking or listening (mostly)
- People worked out how to use them
- We can spoof them for 802.11 APs
- How can we detect spoofing?

Clock Skews (NOT to scale!)

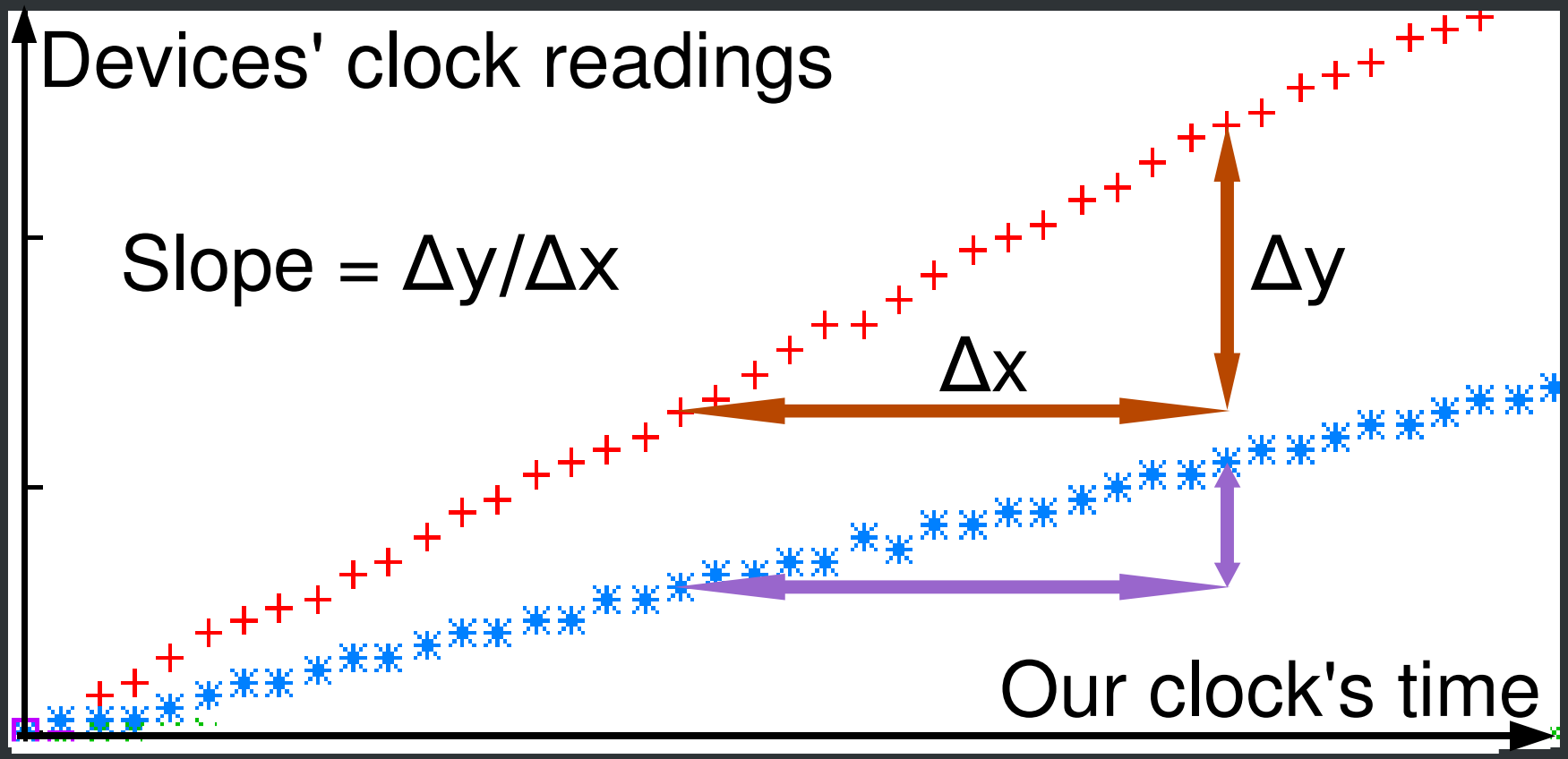
Physical devices have inherent, tiny but consistent drifts in their hardware clocks



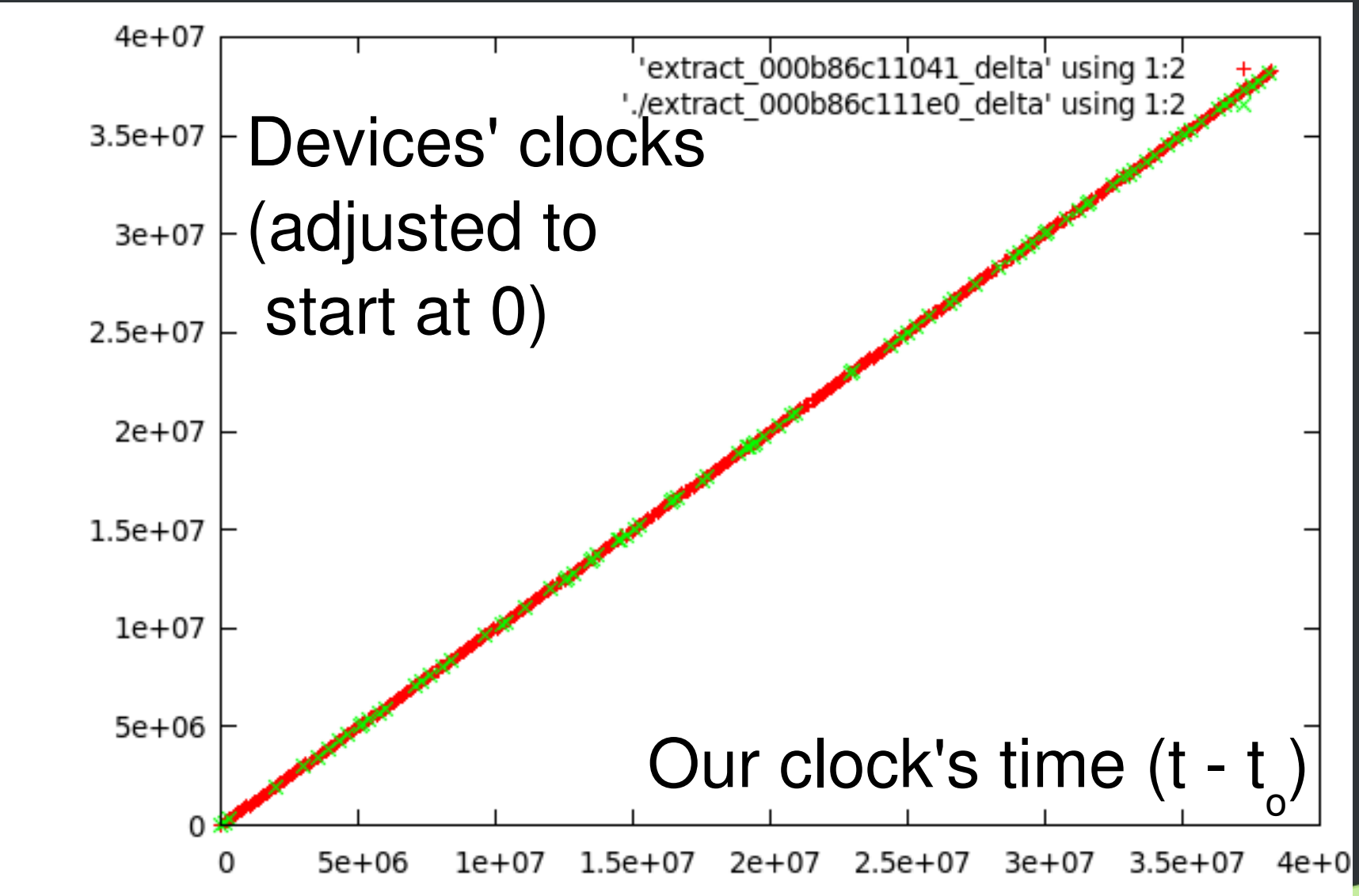
Clock Skews (NOT to scale!)

Different slopes: some clocks are faster

clock skew = slope - 1



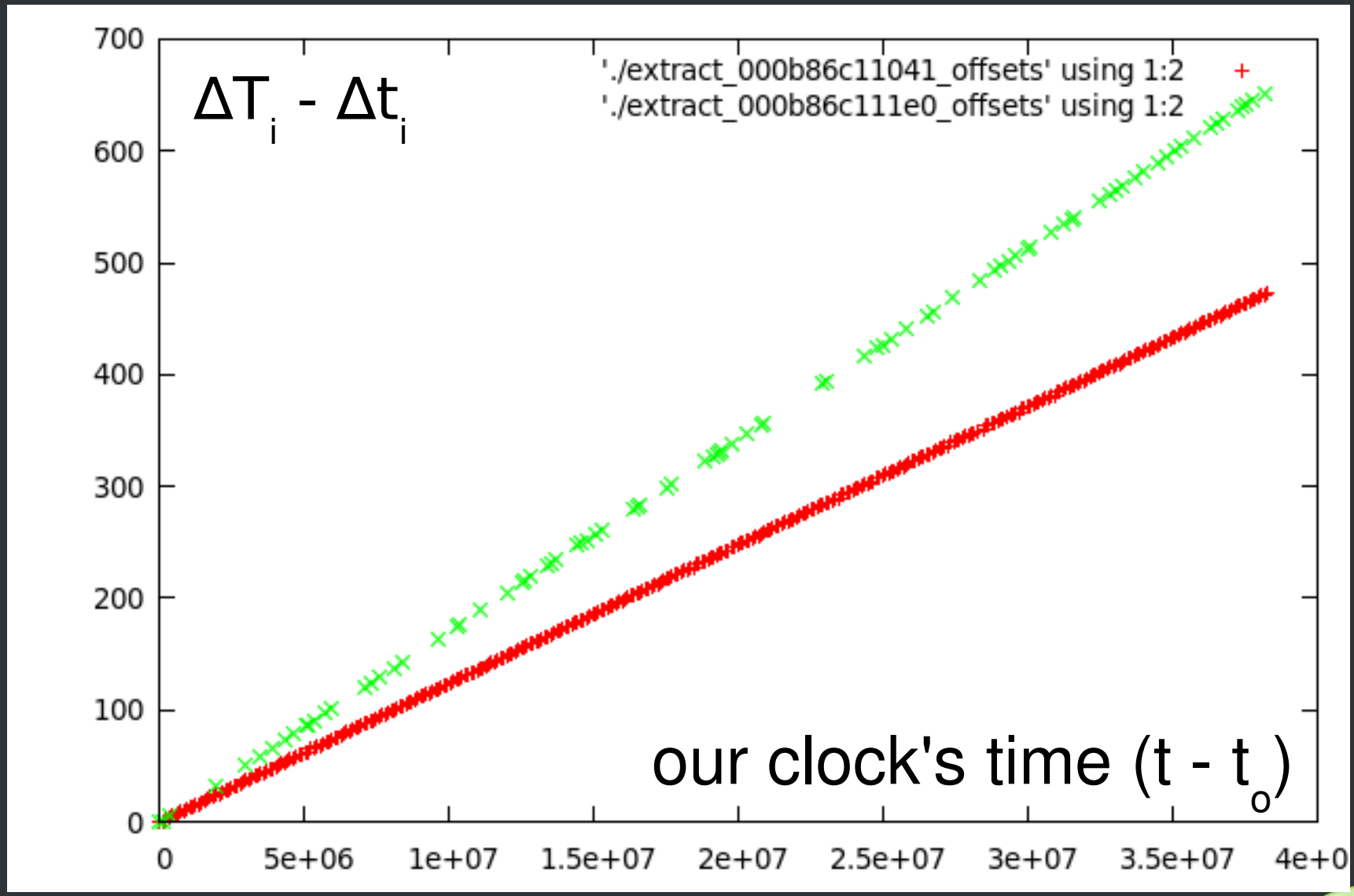
Drawn to scale:



Measuring skews

- Measurements: (t_i, T_i), $i = 1, 2, \ldots, n$
  - t_i: local timestamp
  - T_i: remote timestamp observed
- Clock offsets points: (x_i, y_i)
  - $x_i = \Delta t_i = (t_i - t_0)$
  - $y_i = \Delta T_i - \Delta t_i = (T_i - T_0) - (t_i - t_0)$

Use clock-offset points instead: skew = slope



How to measure skews

- Fit a line $y = mx + c$ to the clock-offset points
- m = clock skew
- Examples
  - skew(Linksys1) = 0.00000668 = 6.68 ppm
  - skew(Linksys2) = - 0.00000785 = -7.85 ppm
  - ppm = parts per million (a millionth, 10^{-6})

Who came up with this

- Kohno, Broido, claffy (2005):
  ”Remote physical device fingerprinting”

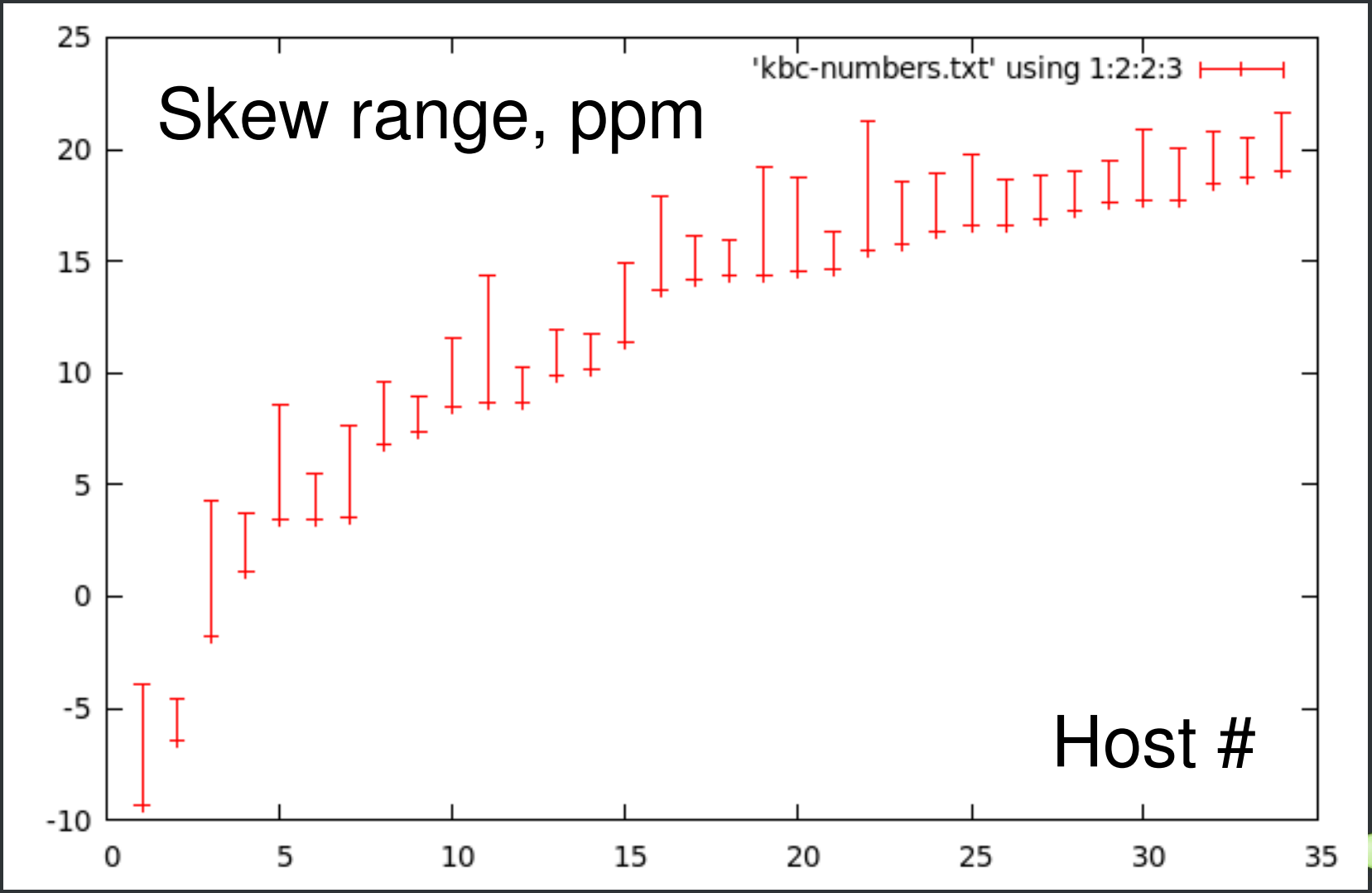
  Clock skews are useful for remotely fingerprinting networked devices
  - different devices have different skews
  - these skews are stable enough over time

Remote fingerprinting

- Have timestamps, can fingerprint!
- Layer 3: can use TCP, ICMP timestamps
  - ICMP timestamp requests (type 13, code 0)
  - TCP timestamp option (option 8 in TCP header)
- ”Do these hosts have the same HW clock?”
  - Of possible cloud-mapping interest?
- Statistics to compensate for network latency, variation

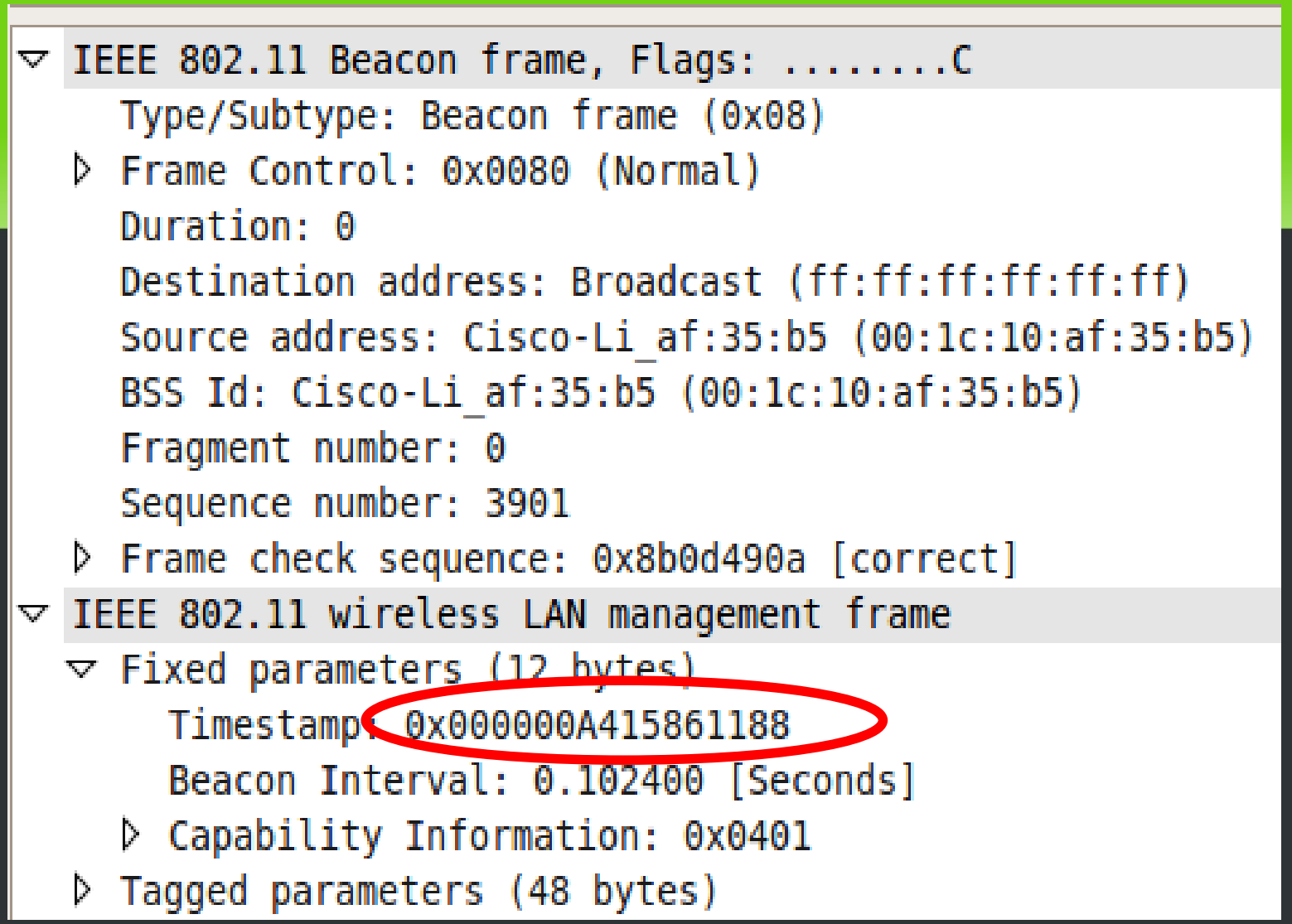
Real-world clock skews

Kohno, Broido, and Claffy data over 12 hours



Clock Skews for 802.11 APs

- Jana, Kasera (2008)
  ”On fast and accurate detection of unauthorized wireless access points using clock skews”
- Bratus, Cornelius, Peebles, Hansen
  ”Active 802.11 fingerprinting” @ BH 2008
  - Beacons transmitted by APs periodically (usually 10/sec)
  - Beacons contain timestamps!
  - Supplied by RF interface chipset clock



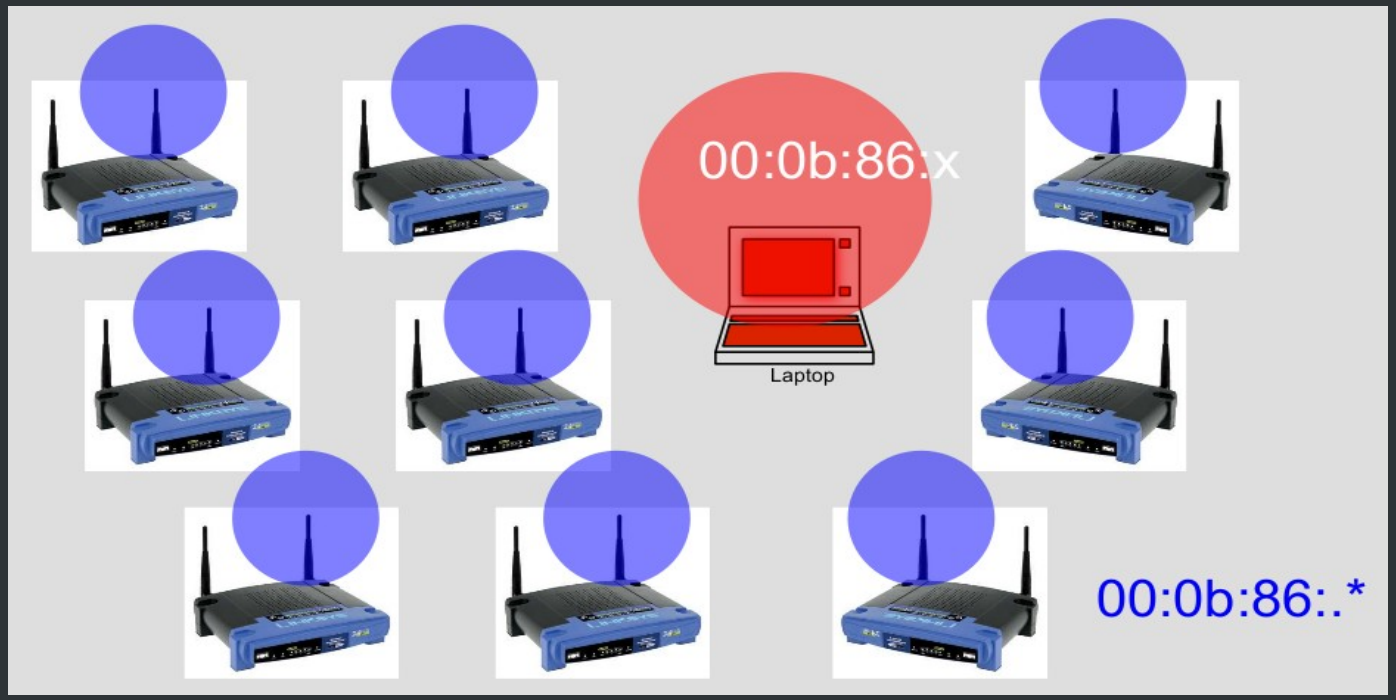

```
$ tshark -r pcapfile -R "wlan.sa==00:1c:10:af:35:b5 &&
  wlan.fc.type_subtype==0x08" -T fields -e
  wlan_mgt.fixed.timestamp
```

Why fingerprint?

- Protecting a wireless client from an ”evil-twin AP”:
  - Shmoo, 2004–2005
  - Karma (Dino Dai Zovi, ...)
  - Johnny Cache, David Maynor @BH 2006
  - ”Month of kernel bugs” in 802.11

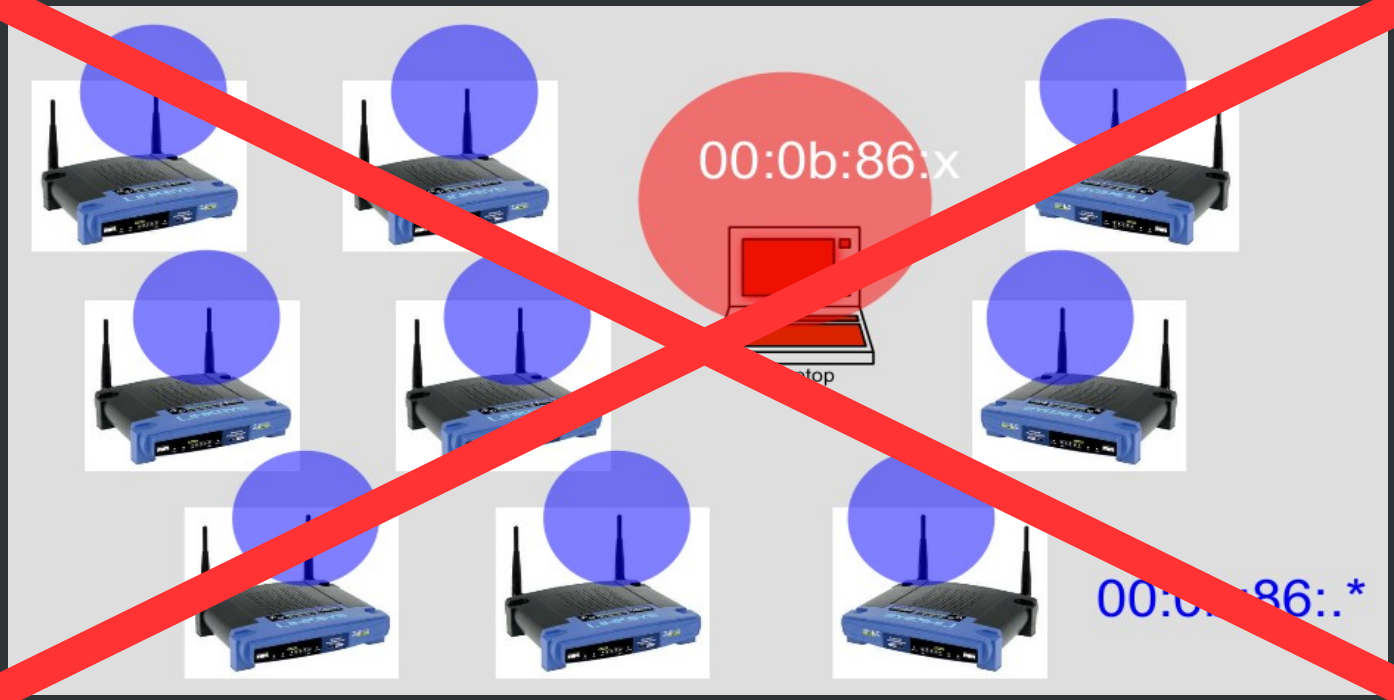
Rogue/Fake APs

- It goes something like this...



Rogue/Fake APs

- It goes something like this...



Rogue/Fake APs

- It goes something like this...

<may I have a Star Wars theme?>

Rogue/Fake APs

- It goes something like this...

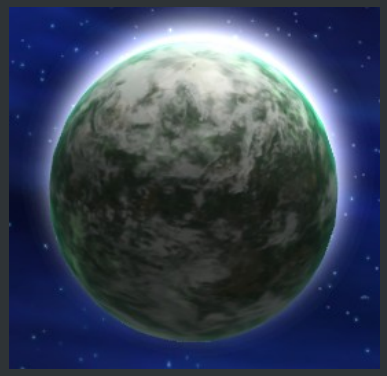
STA

Rogue/Fake APs

- It goes something like this...

STA

APs

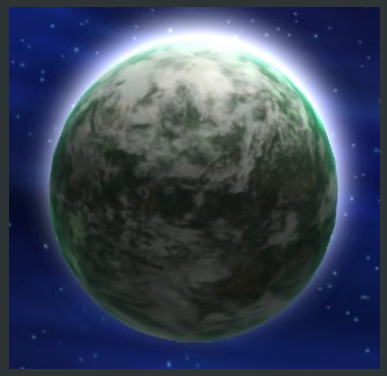


Rogue/Fake APs

- It goes something like this...

STA

APs

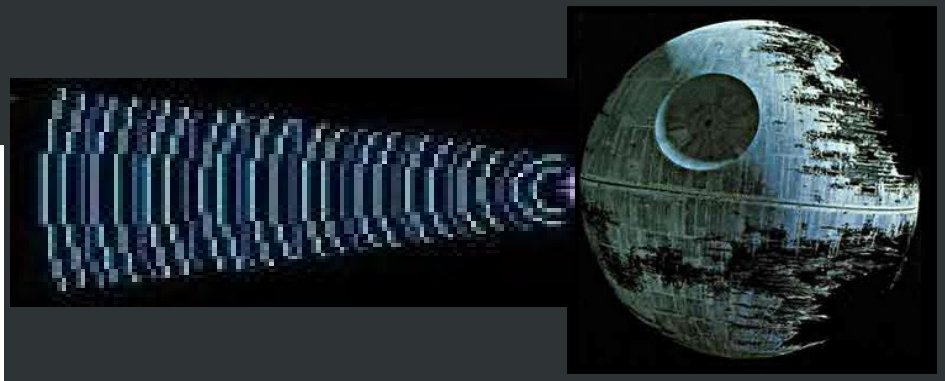


Rogue/Fake APs

- It goes something like this...

Evil AP

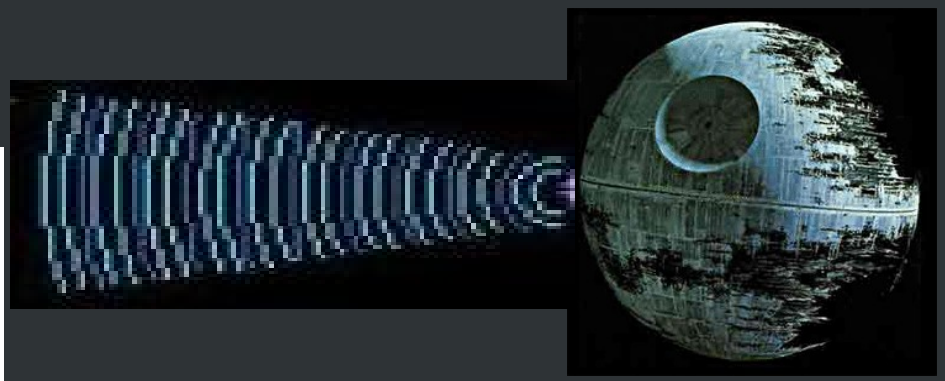
STA



Rogue/Fake APs

- It goes something like this...

STA



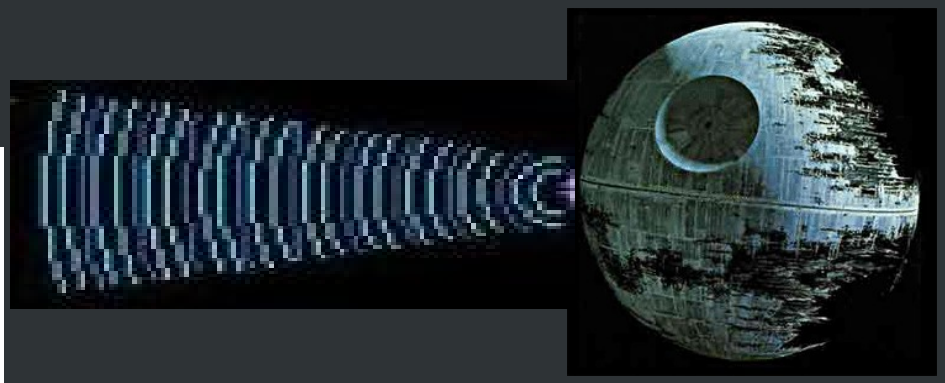
...@#$!!!

Rogue/Fake APs

- It goes something like this...

Evil AP

STA

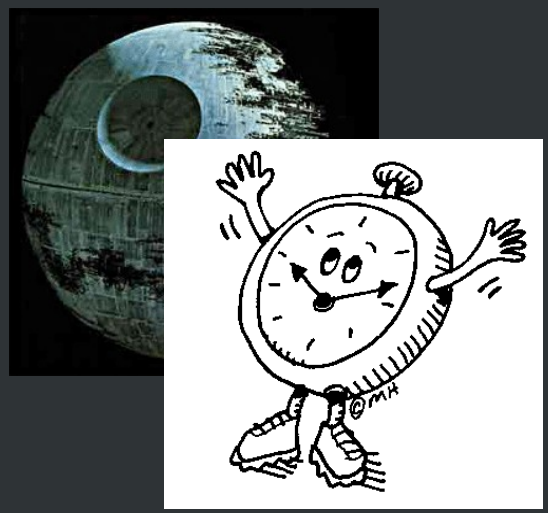


Too bad

Rogue/Fake APs

- It goes something like this...

Evil AP

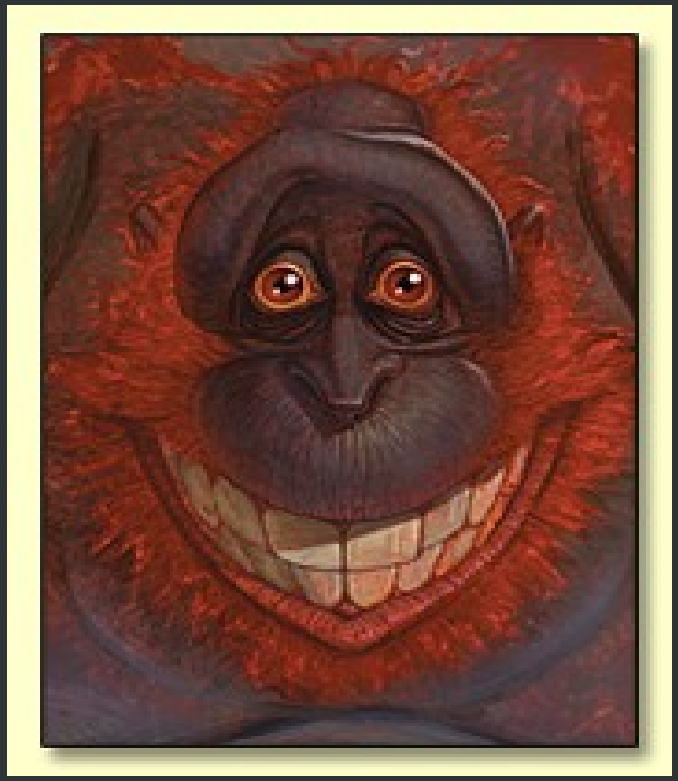


(Beacons with timestamps)

Rogue/Fake APs

- It goes something like this...

... back to 802.11 monkey business ...



The TSF Timer

- Wireless nodes maintain a Timing Sync Function (TSF) timer – 64 bit (microsec)
- Clients adjust their timer from beacon TSF timestamps to sync with AP timer
- RF chipset inserts timestamp into beacon at the moment of transmission
- We have a high-precision stream of timestamps to measure clock skews!

Fast & Accurate Detection of Rogue Access Points

- Jana–Kasera results:

Table 6: Clock Skew estimates in residential setting A as measured from *laptop2*

AP	1st Measurement(LPM)	1st Measurement(LSF)	2nd Measurement(LPM)	2nd Measurement(LSF)
Linksys1	-64.23 ppm	-64.10 ppm	-64.90 ppm	-64.77 ppm
Linksys2	-45.69 ppm	-45.96ppm	-46.94 ppm	-46.71 ppm
Linksys3	-62.05 ppm	-61.84 ppm	-62.77 ppm	-62.64 ppm
Belkin1	-56.37 ppm	-56.57 ppm	-56.71 ppm	-56.85 ppm
Belkin2	-1105.50 ppm	-1105.69 ppm	-1106.29 ppm	-1106.06 ppm
Netgear1	-58.08 ppm	-57.78 ppm	-58.86 ppm	-59.25 ppm
Dlink1	-47.27 ppm	-47.17 ppm	-47.80 ppm	-48.14 ppm
Unknown1	-40.91 ppm	-40.99 ppm	-41.61 ppm	-41.47 ppm

+/- 0.30 – 0.70 for the same AP

How hard is it to spoof?

- Supposing we want to set up evil fake AP that passes the clock skew test?
  - Do we need special equipment?
- Jana–Kasera: pretty hard
  - Constructing beacons to match a known skew and injecting them in RF monitor mode gives inconsistent measurements (est. +/- 100 difference)
  - SW RF mode injection is not fast enough: latency is one problem

Can we do better?

- Can we spoof clock skews with available 802.11 hardware?
  - ... like an Atheros card?
- There are some funny things to observe about actual cards+madwifi ...

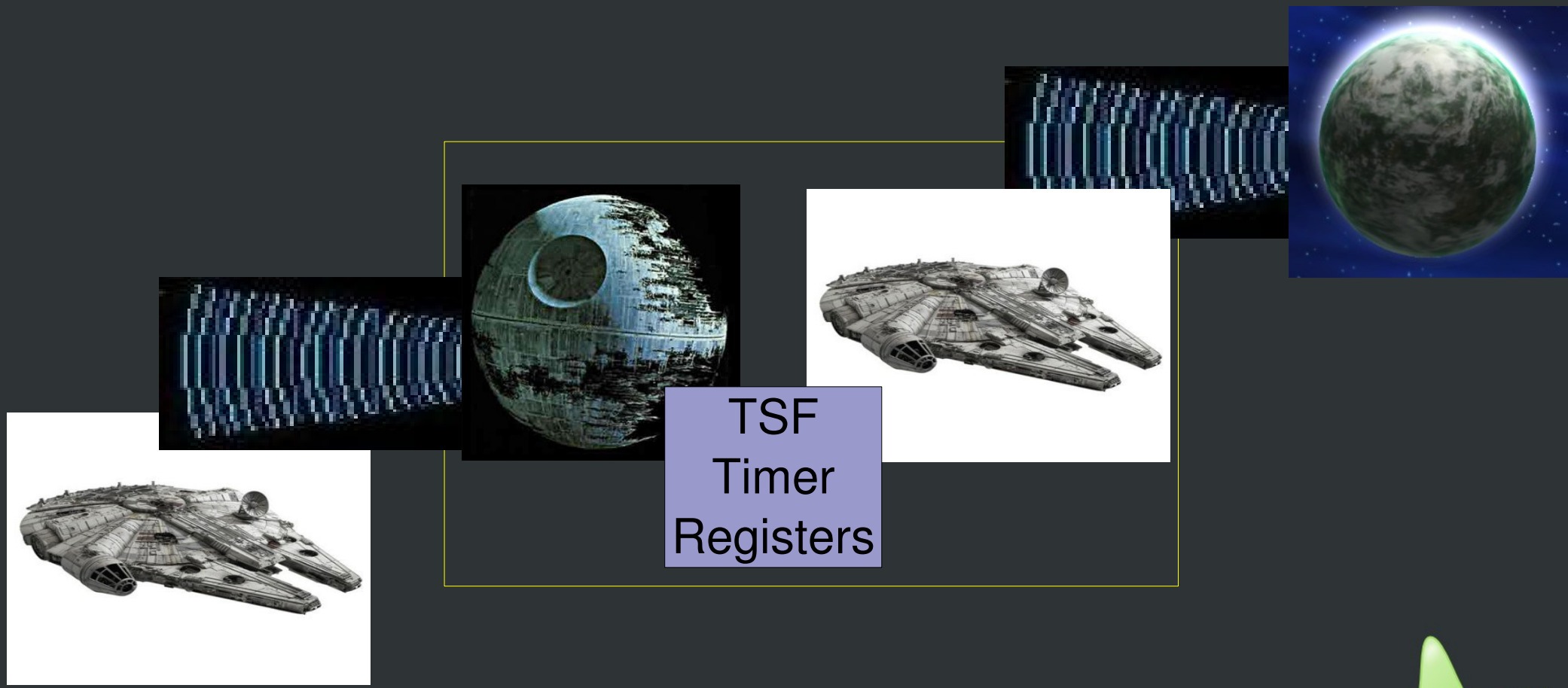
Monitor Mode Synchronization

1) Even after switching from STA to Monitor mode, card continues to update TSF timer from the AP it used to be associated with
2) Skew of that AP measured by the card is zero
3) BTW, if that AP ceases to transmit beacons, the TSF timer of the card begins to drift with its own, actual skew

Monitor Mode Synchronization

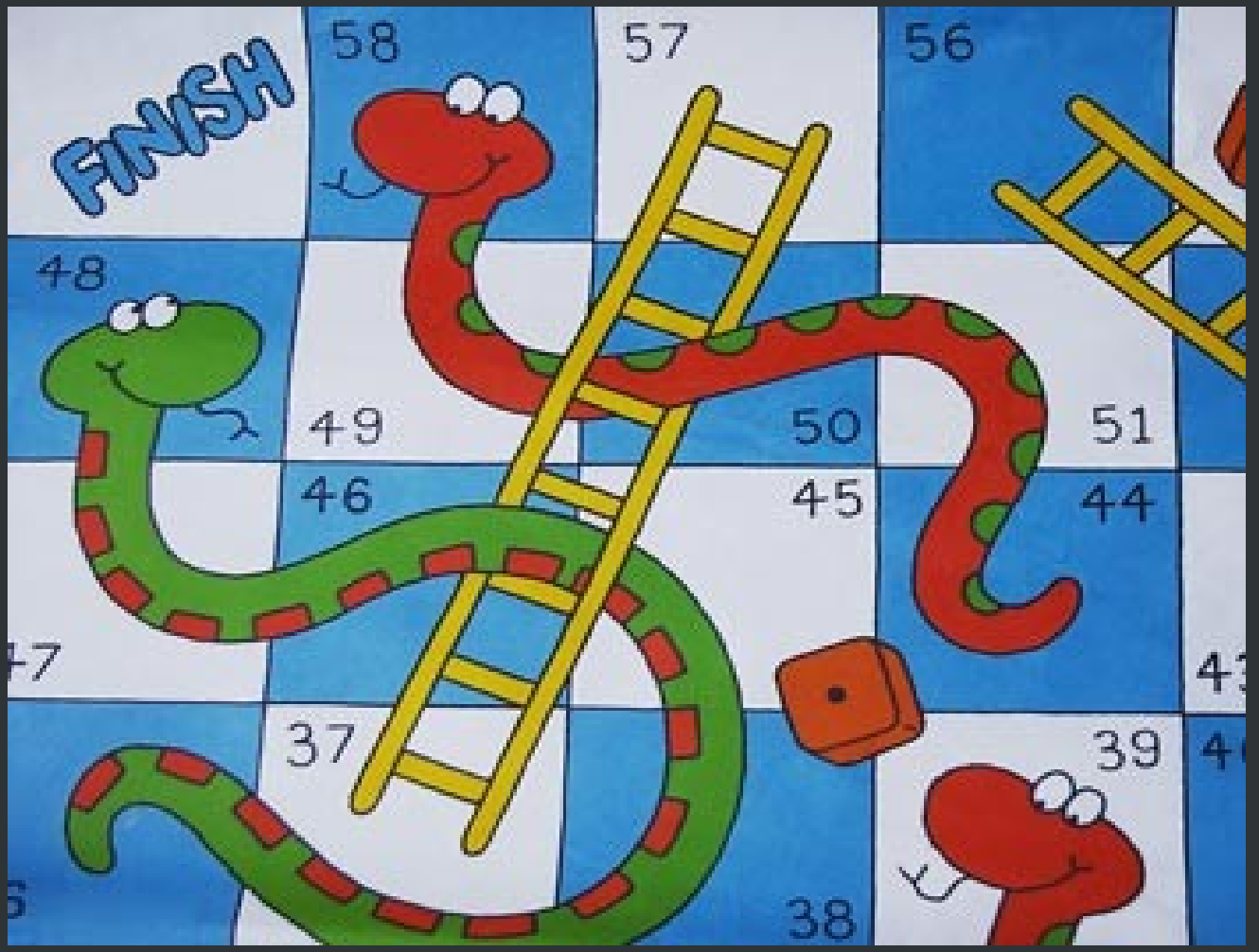
- So an Atheros card continues sync-ing its timer even after leaving association.
- Madwifi gives us ”virtual interfaces” and can bridge (one interface associated with an AP, another in Master mode)
- Can we just get it emit beacons and get AP spoofing for free?

Monitor Mode Synchronization



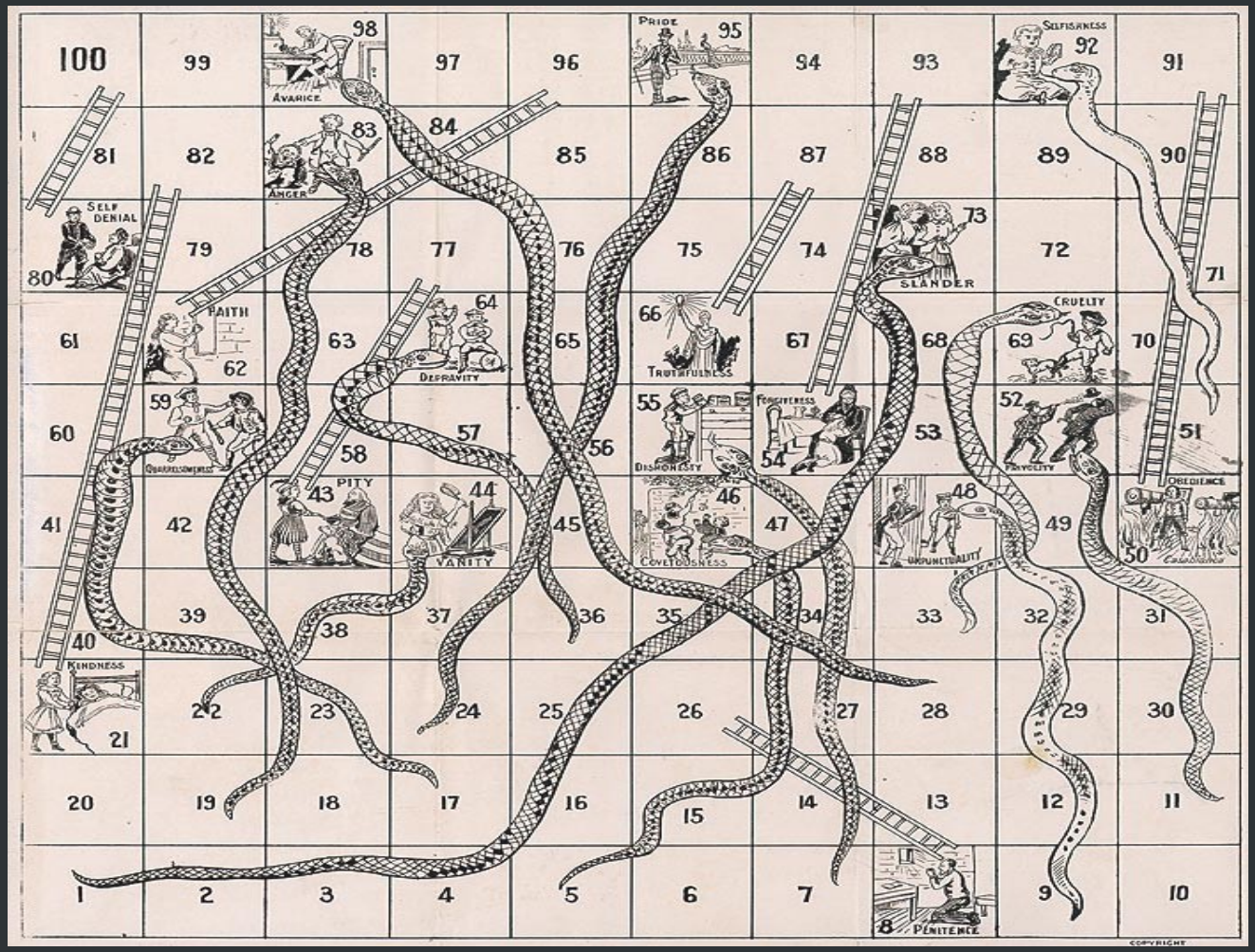
”Snakes and ladders”

- Pity it doesn't work :-(



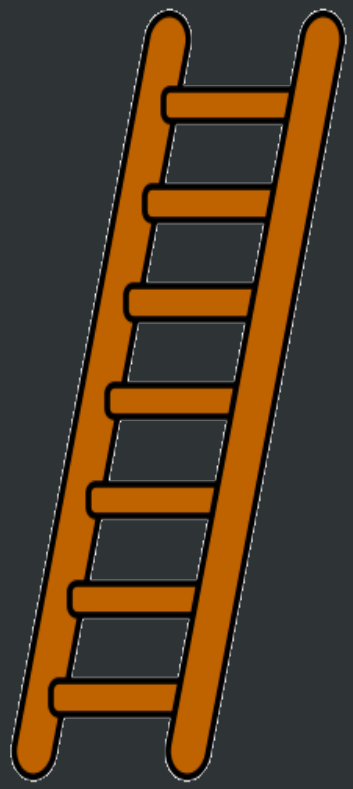
”Snakes and ladders”

- Pity it doesn't work :-(



Something like this:

- MadWifi: multiple virtual interfaces (VAPs) on same hardware card
- AP VAP and STA VAP
  - STA VAP associated with real AP, synchronizing its TSF timer with AP
  - AP VAP using same timer to send spoofing beacons



In ath_vap_create():

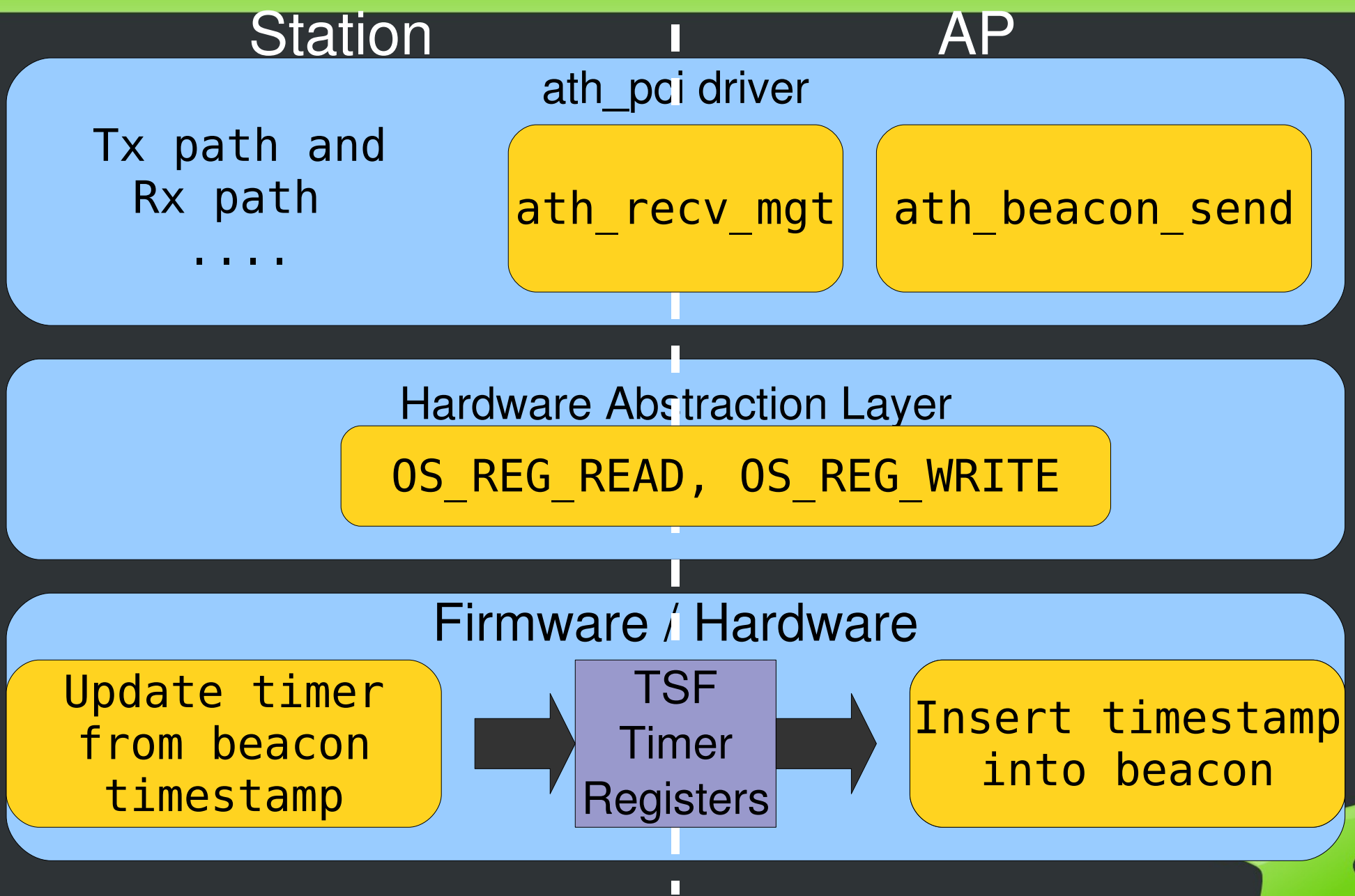
```
  switch(opmode) {
    case IEEE80211_M_HOSTAP:
      if ((sc->sc_nvaps != 0) && \
       (ic->ic_opmode == IEEE80211_M_STA))
         return NULL;

    case IEEE80211_M_STA:
      if (sc->sc_nvaps != 0) {
       flags |= IEEE80211_USE_SW_BEACON_TIMERS;
       sc->sc_nostabeacons = 1;
       ic_opmode = IEEE80211_M_HOSTAP; /* Run
with chip in AP mode */
      } else {
        ic_opmode = opmode;
      }
```

AP VAP needs to be created before STA VAP

Run with chip in STA mode

Driver Architecture



Driver Architecture

- Timer updating logic is inside firmware
- Driver sets firmware to one operating mode (AP)
- Simulates other modes in software
- TSF timer will not be updated from real AP beacons
- Need to patch driver

Driver Patch

```
/*
 * Write two regs together.  Will use for TSF_L32 and TSF_U32,
 * the upper and lower half of the TSF
 */
tsf_dbl_reg_write(struct ath_softc *sc, u_int reg1, u_int32_t val1, u_int
reg2, u_int32_t val2)
{
        ATH_HAL_LOCK_IRQ(sc);

        OS_REG_WRITE(sc->sc_ah, reg1, val1);
        OS_REG_WRITE(sc->sc_ah, reg2, val2);

        ATH_HAL_UNLOCK_IRQ(sc);
}
```

HAL register write

Driver Patch

In ath_recv_mgmt:

Beacon frame received

```
    if (subtype == 0x80) {
```

Get timestamp from beacon

```
+       time = (unsigned long long)le64_to_cpu(ni_or_null->ni_tstamp.tsf);
+        ptr = (u_int32_t *) (&time);
+        tsf_dbl_reg_write(sc, AR_TSF_L32, ptr[0], AR_TSF_U32, ptr[1]);
```

Update TSF regs

...and we hit a snake

- Mucking around with the values in the TSF registers breaks beaconing
- ...beacon scheduling disrupted?

Driver Patch

In ath_recv_mgmt:

Beacon frame received

```
    if (subtype == 0x80) {
```

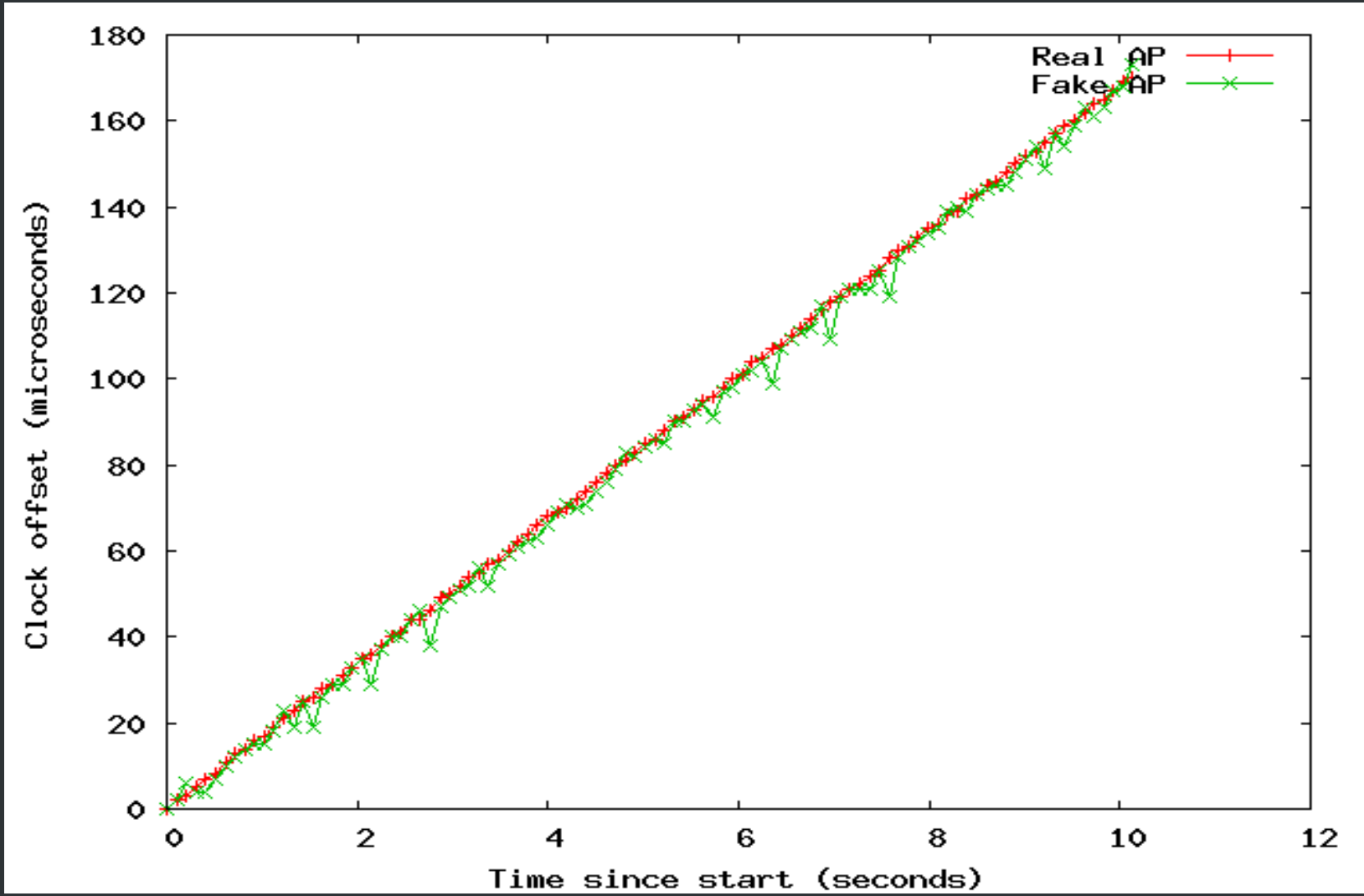
Get timstamp from beacon
Update TSF regs

```
+       time = (unsigned long long)le64_to_cpu(ni_or_null->ni_tstamp.tsf);
+        ptr = (u_int32_t *) (&time);
+        tsf_dbl_reg_write(sc, AR_TSF_L32, ptr[0], AR_TSF_U32, ptr[1]);
```

Force beacon transmission

```
 +      ath_beacon_send(sc, &needmark, new);
```

The Result


Real AP
Fake AP
Clock offset (microseconds)
Time since start (seconds)
0
20
40
60
80
100
120
140
160
180
0
2
4
6
8
10
12

The Result

Real Skew	Spoofed Skew
16.79	16.78
16.82	16.69
16.80	16.74
16.81	16.78

This is within typical variation of one AP's (+/- 0.30 – 0.70) clock skew – close enough!

Can we finesse it?

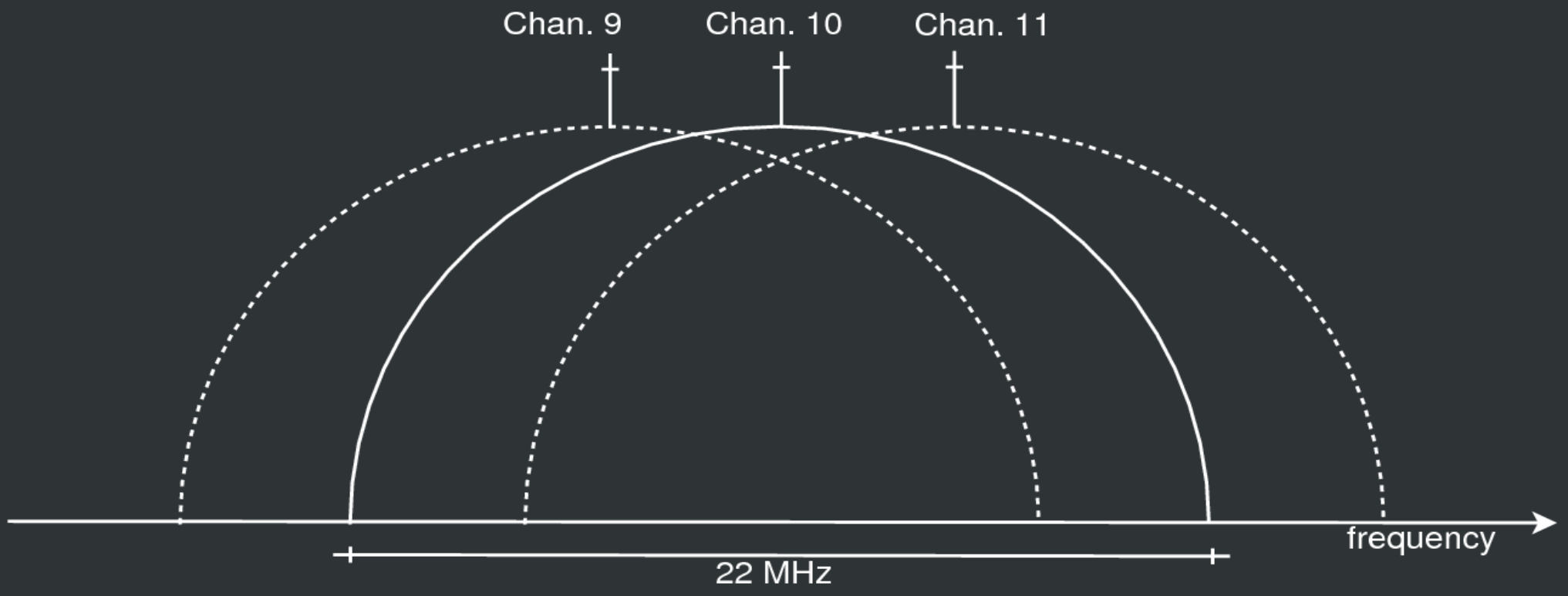
What if the client hears both fake and real AP on the same channel?

- Timestamps will collide, screw up skew

- Place the fake AP on a different channel!

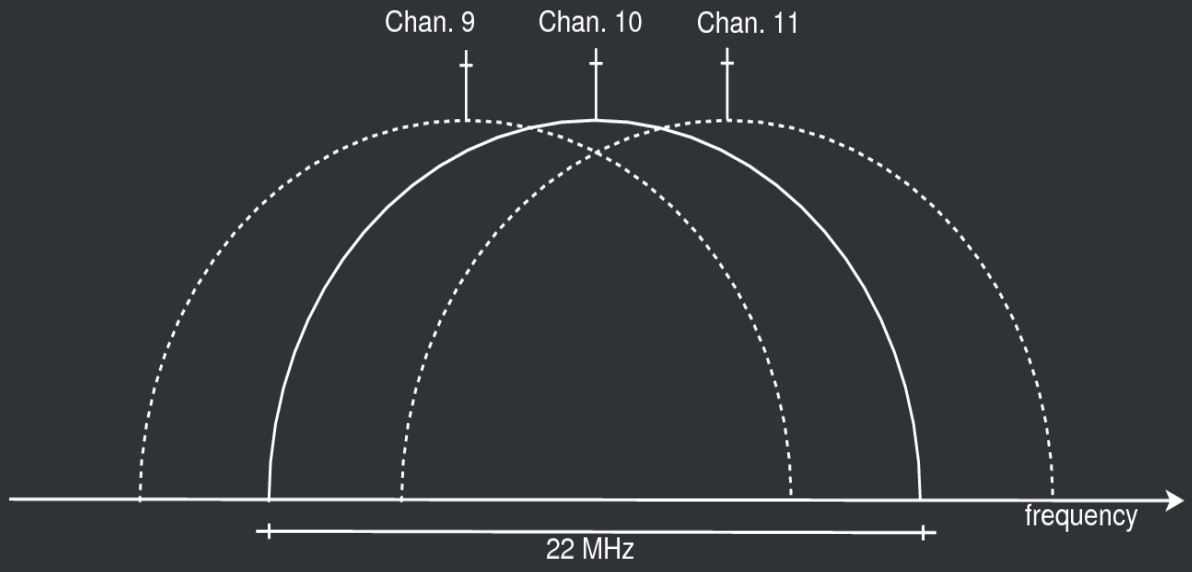
Bridging the Fake AP

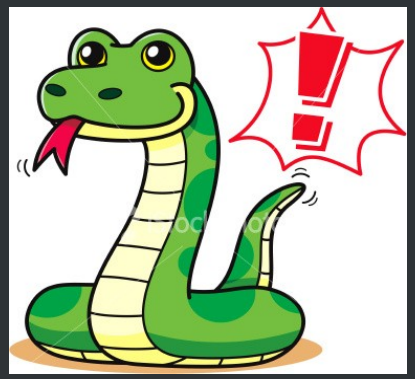
- Overlapping 802.11 channels



Staying on an overlapping channel

- The card will automatically switch to the channel of the associated AP
- Must keep it on the chosen neighboring channel instead





Keep card on overlapping channel

```
 add_channels(struct ieee80211com *ic,  // <skipped args>
-                    int nfreq)
+                   int nfreq,  struct ieee80211vap *vap)
 {
                // <skip>


-               ss->ss_chans[ss->ss_last++] = c;
+               ss->ss_chans[ss->ss_last++] = vap->iv_des_chan;


}
```

Channel found by scanning

Spoofing channel supplied by us

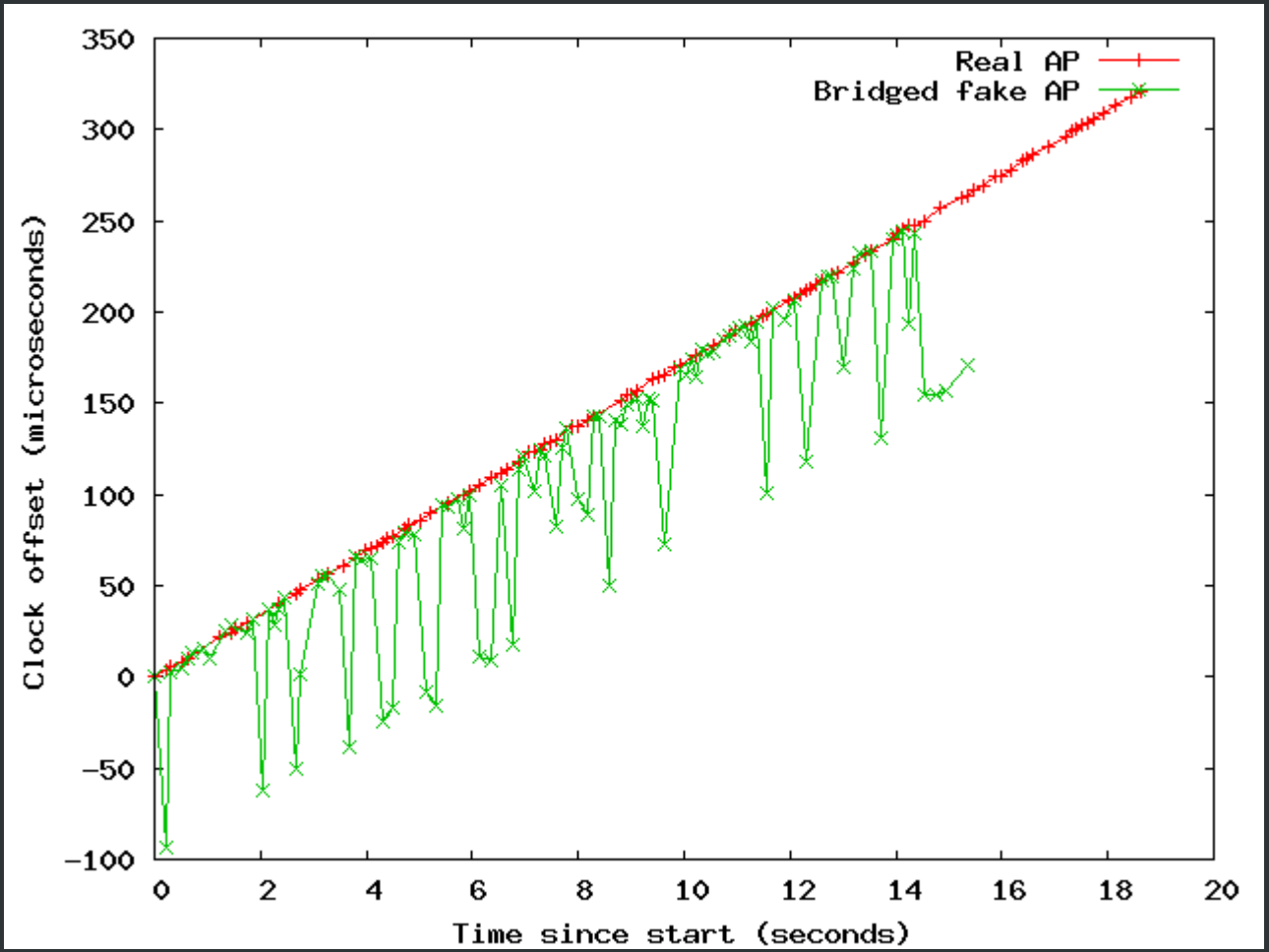
More Patching

```
ieee80211_recv_mgmt()
{
// <snip>

                if (scan.chan != scan.bchan &&
                    ic->ic_phytype != IEEE80211_T_FH) {
// <snip>
                        vap->iv_stats.is_rx_chanmismatch++;
-                       return 0;
+                       // return 0;
                }
```

Avoid filtering out beacons

The Result


Real AP
Bridged fake AP
Clock offset (microseconds)
Time since start (seconds)
350
300
250
200
150
100
50
0
−50
−100
0
2
4
6
8
10
12
14
16
18
20

The Result

Real Skew	Spoofed Skew
17.25	16.49
17.29	17.58
17.26	17.54
17.35	16.29

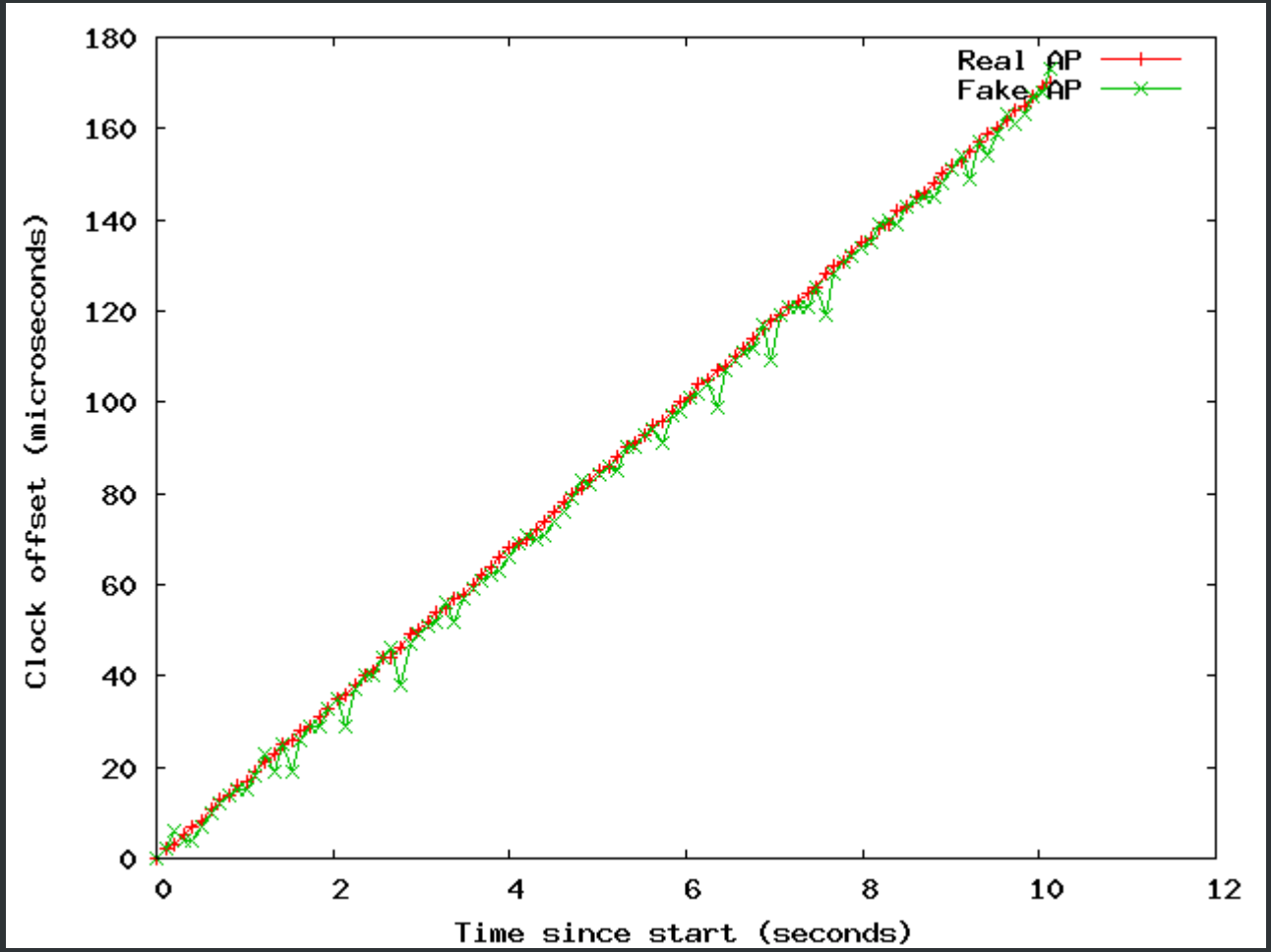
We are within skew variation most of the time

Detecting Spoofing

- Effect of beacon interval
- Line fitting error
  - Y-intercept of the fitted line (will not pass through the origin)
  - Jitter (how much jumping around)

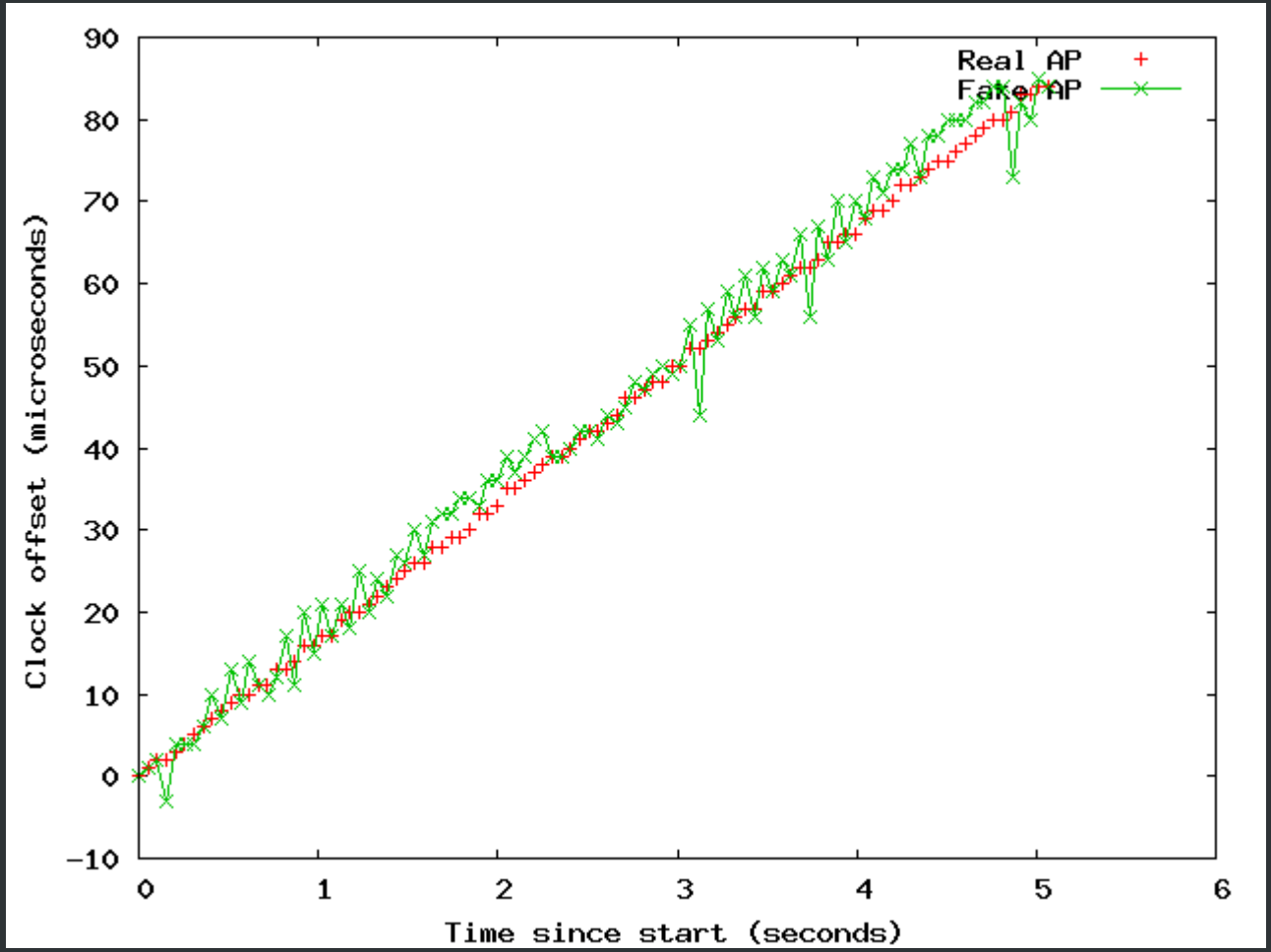
Effect of Beacon Interval

- Beacon Interval = 100ms (10 beac/sec)



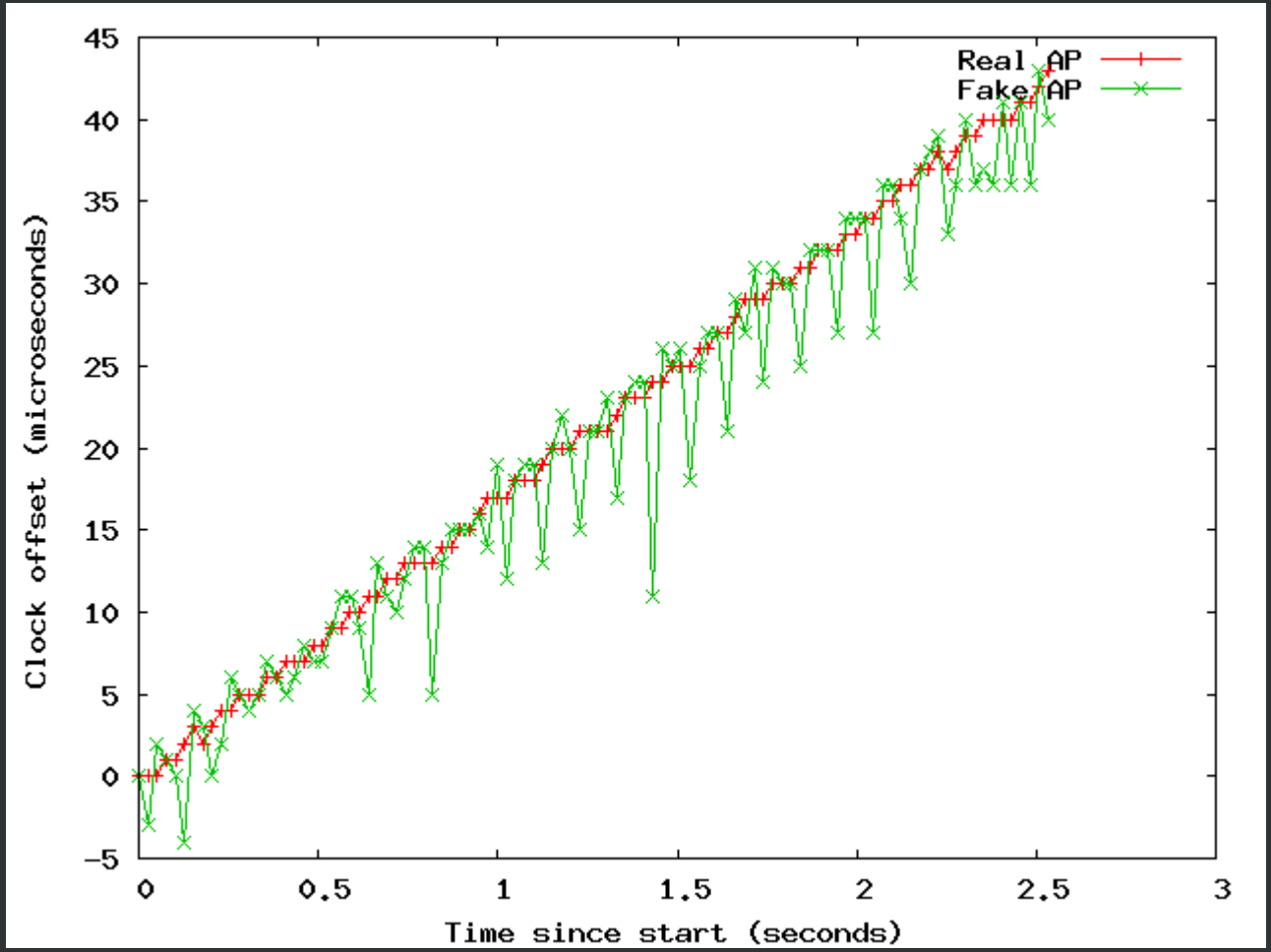
Effect of Beacon Interval

- Beacon Interval = 50ms (20 beac/sec)



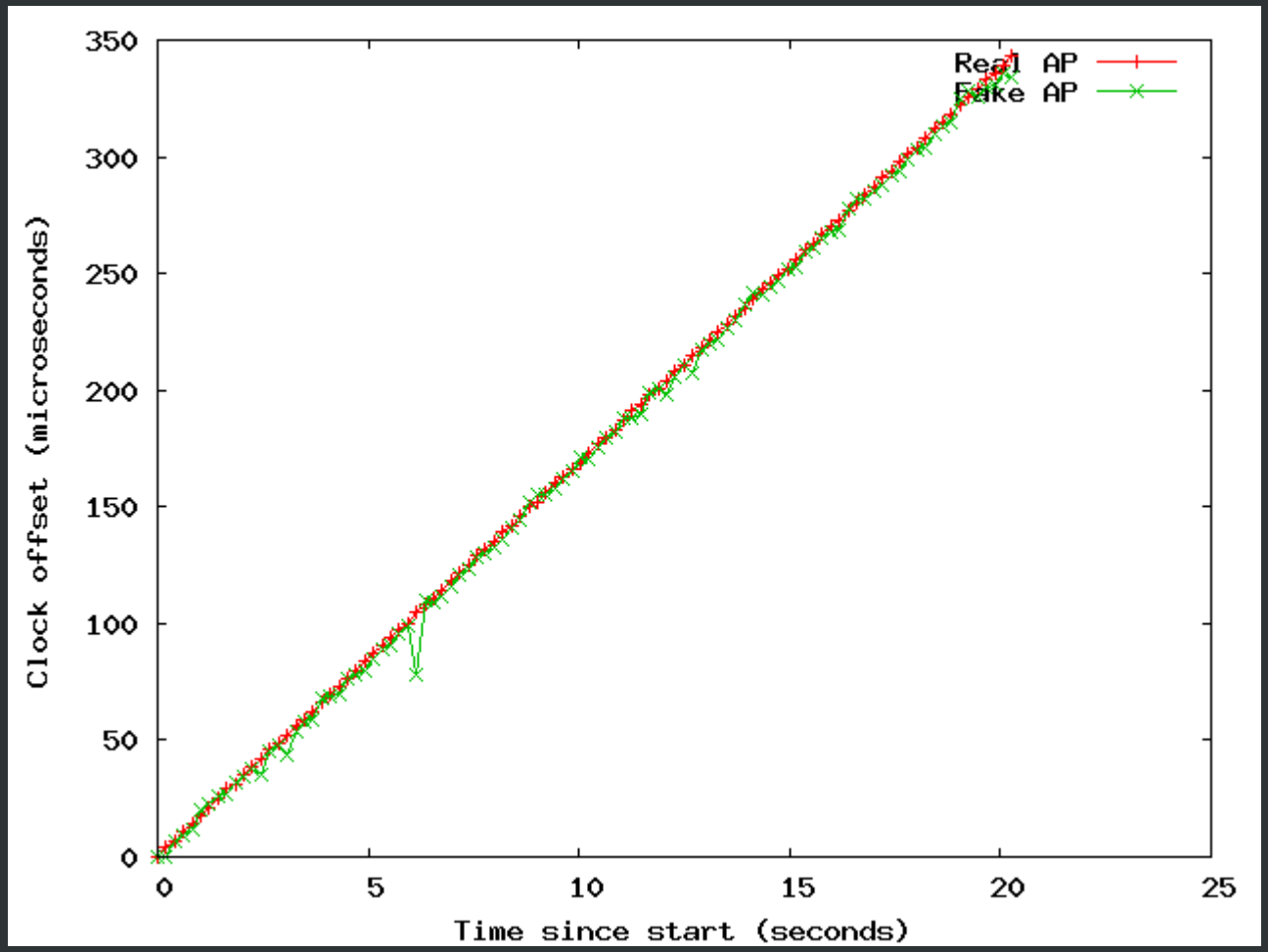
Effect of Beacon Interval

- Beacon Interval = 25ms (40 beac/sec)



Effect of Beacon Interval

- Beacon Interval = 200ms (5 beac/sec)



Conclusions

- Timestamp measurements are exciting
  - Interesting papers exist
- Asking *”What's the time, please?”* may be a good prelude to closer association
- Unfortunately, AP Beacon timestamps appear to be spoofable
- We should all try harder! ;-)

Thank you!

Patches to be posted at

http://baffle.cs.dartmouth.edu/

Line Fitting Error