NAME

ieee80211_scan - 802.11 scanning support

SYNOPSIS

#include <net80211/ieee80211_var.h>

int
ieee80211_start_scan(*struct ieee80211vap **, *int flags*, *u_int duration*, *u_int mindwell*, *u_int maxdwell*,
    u_int nssid, *const struct ieee80211_scan_ssid ssids[]*);

int
ieee80211_check_scan(*struct ieee80211vap **, *int flags*, *u_int duration*, *u_int mindwell*, *u_int maxdwell*,
    u_int nssid, *const struct ieee80211_scan_ssid ssids[]*);

int
ieee80211_check_scan_current(*struct ieee80211vap **);

int
ieee80211_bg_scan(*struct ieee80211vap **, *int*);

int
ieee80211_cancel_scan(*struct ieee80211vap **);

int
ieee80211_cancel_scan_any(*struct ieee80211vap **);

int
ieee80211_scan_next(*struct ieee80211vap **);

int
ieee80211_scan_done(*struct ieee80211vap **);

int
ieee80211_probe_curchan(*struct ieee80211vap **, *int*);

void
ieee80211_add_scan(*struct ieee80211vap **, *const struct ieee80211_scanparams **,
    *const struct ieee80211_frame **, *int subtype*, *int rssi*, *int noise*);

void

ieee80211_scan_timeout(*struct ieee80211com **);

void
ieee80211_scan_assoc_fail(*struct ieee80211vap **, *const uint8_t mac[IEEE80211_ADDR_LEN]*, *int reason*);

void
ieee80211_scan_flush(*struct ieee80211vap **);

void
ieee80211_scan_iterate(*struct ieee80211vap **, *ieee80211_scan_iter_func*, *void **);

void
ieee80211_scan_dump_channels(*const struct ieee80211_scan_state **);

void
ieee80211_scanner_register(*enum ieee80211_opmode*, *const struct ieee80211_scanner **);

void
ieee80211_scanner_unregister(*enum ieee80211_opmode*, *const struct ieee80211_scanner **);

void
ieee80211_scanner_unregister_all(*const struct ieee80211_scanner **);

*const struct ieee80211_scanner **
ieee80211_scanner_get(*enum ieee80211_opmode*);

DESCRIPTION

The **net80211** software layer provides an extensible framework for scanning. Scanning is the procedure by which a station locates a BSS to join (in infrastructure and IBSS mode), or a channel to use (when operating as an AP or an IBSS master). Scans are either "active" or "passive". An active scan causes one or more ProbeRequest frames to be sent on visiting each channel. A passive request causes each channel in the scan set to be visited but no frames to be transmitted; the station only listens for traffic. Note that active scanning may still need to listen for traffic before sending ProbeRequest frames depending on regulatory constraints.

A scan operation involves constructing a set of channels to inspect (the scan set), visiting each channel and collecting information (e.g. what BSS are present), and then analyzing the results to make decisions such as which BSS to join. This process needs to be as fast as possible so **net80211** does things like intelligently construct scan sets and dwell on a channel only as long as necessary. Scan results are cached and the scan cache is used to avoid scanning when possible and to enable roaming between access points when operating in infrastructure mode.

Scanning is handled by pluggable modules that implement *policy* per-operating mode. The core scanning support provides an infrastructure to support these modules and exports a common API to the rest of the **net80211** layer. Policy modules decide what channels to visit, what state to record to make decisions, and selects the final station/channel to return as the result of a scan.

Scanning is done synchronously when initially bringing a vap to an operational state and optionally in the background to maintain the scan cache for doing roaming and rogue AP monitoring. Scanning is not tied to the **net80211** state machine that governs vaps except for linkage to the IEEE80211_S_SCAN state. Only one vap at a time may be scanning; this scheduling policy is handled in **ieee80211_new_state**() and is transparent to scanning code.

Scanning is controlled by a set of parameters that (potentially) constrains the channel set and any desired SSID's and BSSID's. **net80211** comes with a standard scanner module that works with all available operating modes and supports "background scanning" and "roaming" operation.

SCANNER MODULES

Scanning modules use a registration mechanism to hook into the **net80211** layer. Use **ieee80211_scanner_register**() to register a scan module for a particular operating mode and **ieee80211_scanner_unregister**() or **ieee80211_scanner_unregister_all**() to clear entries (typically on module unload). Only one scanner module can be registered at any time for an operating mode.

DRIVER SUPPORT

Scanning operations are usually managed by the **net80211** layer. Drivers must provide *ic_scan_start* and *ic_scan_stop* methods that are called at the start of a scan and when the work is done; these should handle work such as enabling receive of Beacon and ProbeResponse frames and disable any BSSID matching. The *ic_set_channel* method is used to change channels while scanning. **net80211** will generate ProbeRequest frames and transmit them using the **ic_raw_xmit** method. Frames received while scanning are dispatched to **net80211** using the normal receive path. Devices that off-load scan work to firmware most easily mesh with **net80211** by operating on a channel-at-a-time basis as this defers control to **net80211's** scan machine scheduler. But multi-channel scanning is supported if the driver manually dispatches results using **ieee80211_add_scan**() routine to enter results into the scan cache.

SCAN REQUESTS

Scan requests occur by way of the IEEE80211_SCAN_REQUEST ioctl or through a change in a vap's state machine that requires scanning. In both cases the scan cache can be checked first and, if it is deemed suitably "warm" then it's contents are used without leaving the current channel. To start a scan without checking the cache **ieee80211_start_scan**() can be called; otherwise **ieee80211_check_scan**()

can be used to first check the scan cache, kicking off a scan if the cache contents are out of date. There is also **ieee80211_check_scan_current**() which is a shorthand for using previously set scan parameters for checking the scan cache and then scanning.

Background scanning is done using **ieee80211_bg_scan**() in a co-routine fashion. The first call to this routine will start a background scan that runs for a limited period of time before returning to the BSS channel. Subsequent calls advance through the scan set until all channels are visited. Typically these later calls are timed to allow receipt of frames buffered by an access point for the station.

A scan operation can be canceled using **ieee80211_cancel_scan**() if it was initiated by the specified vap, or **ieee80211_cancel_scan_any**() to force termination regardless which vap started it. These requests are mostly used by **net80211** in the transmit path to cancel background scans when frames are to be sent. Drivers should not need to use these calls (or most of the calls described on this page).

The **ieee80211_scan_next**() and **ieee80211_scan_done**() routines do explicit iteration through the scan set and should not normally be used by drivers. **ieee80211_probe_curchan**() handles the work of transmitting ProbeRequest frames when visiting a channel during an active scan. When the channel attributes are marked with IEEE80211_CHAN_PASSIVE this function will arrange that before any frame is transmitted 802.11 traffic is first received (in order to comply with regulatory constraints).

Min/max dwell time parameters are used to constrain time spent visiting a channel. The maximum dwell time constrains the time spent listening for traffic. The minimum dwell time is used to reduce this time--when it is reached and one or more frames have been received then an immediate channel change will be done. Drivers can override this behaviour through the *iv_scan_mindwell* method.

SCAN CACHE MANAGEMENT

The scan cache contents are managed by the scan policy module and are opaque outside this module. The **net80211** scan framework defines API's for interacting. The validity of the scan cache contents are controlled by *iv_scanvalid* which is exported to user space through the IEEE80211_SCAN_VALID request.

The cache contents can be explicitly flushed with **ieee80211_scan_flush**() or by setting the IEEE80211_SCAN_FLUSH flag when starting a scan operation.

Scan cache entries are created with the **ieee80211_add_scan**() routine; usually on receipt of Beacon or ProbeResponse frames. Existing entries are typically updated based on the latest information though some information such as RSSI and noise floor readings may be combined to present an average.

The cache contents is aged through **ieee80211_scan_timeout**() calls. Typically these happen together with other station table activity; every IEEE80211_INACT_WAIT seconds (default 15).

Individual cache entries are marked usable with **ieee80211_scan_assoc_success**() and faulty with **ieee80211_scan_assoc_fail**() with the latter taking an argument to identify if there was no response to Authentication/Association requests or if a negative response was received (which might hasten cache eviction or blacklist the entry).

The cache contents can be viewed using the **ieee80211_scan_iterate**() call. Cache entries are exported in a public format that is exported to user applications through the IEEE80211_SCAN_RESULTS request.

SEE ALSO

ioctl(2), ieee80211(9), ieee80211_proto(9)