NAME

ieee80211_node - software 802.11 stack node management functions

SYNOPSIS

#include <net80211/ieee80211_var.h>

*struct ieee80211_node **
ieee80211_find_rxnode(*struct ieee80211com *, const struct ieee80211_frame_min **);

*struct ieee80211_node **
ieee80211_find_rxnode_withkey(*struct ieee80211com *, const struct ieee80211_frame_min **, *ieee80211_keyix*);

*struct ieee80211_node **
ieee80211_ref_node(*struct ieee80211_node **);

void
ieee80211_unref_node(*struct ieee80211_node **);

void
ieee80211_free_node(*struct ieee80211_node **);

void
ieee80211_iterate_nodes(*struct ieee80211_node_table *, ieee80211_iter_func *f, void *arg*);

void
ieee80211_dump_nodes(*struct ieee80211_node_table **);

void
ieee80211_dump_node(*struct ieee80211_node **);

DESCRIPTION

The **net80211** layer that supports 802.11 device drivers maintains a database of peer stations called the "node table" in the *ic_sta* entry of the *ieee80211com* structure. Station mode vaps create an entry for the access point the station is associated to. AP mode vaps create entries for associated stations. Adhoc and mesh mode vaps create entries for neighbor stations. WDS mode vaps create an entry for the peer station. Stations for all vaps reside in the same table; each node entry has a *ni_vap* field that identifies the vap that created it. In some instances an entry is used by multiple vaps (e.g. for dynamic WDS a station associated to an ap vap may also be the peer of a WDS vap).

Node table entries are reference counted. That is, there is a count of all long term references that determines when an entry may be reclaimed. References are held by every in-flight frame sent to a station to ensure the entry is not reclaimed while the frame is queued or otherwise held by a driver. Routines that lookup a table entry return a "held reference" (i.e. a pointer to a table entry with the reference count incremented). The **ieee80211_ref_node**() and **ieee80211_unref_node**() calls explicitly increment/decrement the reference count of a node, but are rarely used. Instead most callers use **ieee80211_free_node**() to release a reference and, if the count goes to zero, reclaim the table entry.

The station table and its entries are exposed to drivers in several ways. Each frame transmitted to a station includes a reference to the associated node in the *m_pkthdr.rcvif* field. This reference must be reclaimed by the driver when transmit processing is done. For each frame received the driver must lookup the table entry to use in dispatching the frame "up the stack". This lookup implicitly obtains a reference to the table entry and the driver must reclaim the reference when frame processing is completed. Otherwise drivers frequently inspect the contents of the *iv_bss* node when handling state machine changes as important information is maintained in the data structure.

The node table is opaque to drivers. Entries may be looked up using one of the pre-defined API's or the **ieee80211_iterate_nodes**() call may be used to iterate through all entries to do per-node processing or implement some non-standard search mechanism. Note that **ieee80211_iterate_nodes**() is single-threaded per-device and the effort processing involved is fairly substantial so it should be used carefully.

Two routines are provided to print the contents of nodes to the console for debugging: **ieee80211_dump_node**() displays the contents of a single node while **ieee80211_dump_nodes**() displays the contents of the specified node table. Nodes may also be displayed using ddb(4) with the "show node" directive and the station node table can be displayed with "show statab".

DRIVER PRIVATE STATE

Node data structures may be extended by the driver to include driver-private state. This is done by overriding the *ic_node_alloc* method used to allocate a node table entry. The driver method must allocate a structure that is an extension of the *ieee80211_node* structure. For example the iwi(4) driver defines a private node structure as:

```
struct iwi_node {
        struct ieee80211_node   in_node;
                int                     in_station;
};
```

and then provides a private allocation routine that does this:

```
static struct ieee80211_node *
```

```
iwi_node_alloc(struct ieee80211vap *vap,
   const uint8_t mac[IEEE80211_ADDR_LEN])
{
      struct iwi_node *in;

      in = malloc(sizeof(struct iwi_node), M_80211_NODE,
                  M_NOWAIT | M_ZERO);
      if (in == NULL)
            return NULL;
      in->in_station = -1;
      return &in->in_node;
}
```

Note that when reclaiming a node allocated by the driver the "parent method" must be called to ensure **net80211** state is reclaimed; for example:

```
static void
iwi_node_free(struct ieee80211_node *ni)
{
      struct ieee80211com *ic = ni->ni_ic;
      struct iwi_softc *sc = ic->ic_ifp->if_softc;
      struct iwi_node *in = (struct iwi_node *)ni;

      if (in->in_station != -1)
            free_unr(sc->sc_unr, in->in_station);
      sc->sc_node_free(ni);     /* invoke net80211 free handler */
}
```

Beware that care must be taken to avoid holding references that might cause nodes from being reclaimed. **net80211** will reclaim a node when the last reference is reclaimed in its data structures. However if a driver holds additional references then **net80211** will not recognize this and table entries will not be reclaimed. Such references should not be needed if the driver overrides the *ic_node_cleanup* and/or *ic_node_free* methods.

KEY TABLE SUPPORT

Node table lookups are typically done using a hash of the stations' mac address. When receiving frames this is sufficient to find the node table entry for the transmitter. But some devices also identify the sending station in the device state received with each frame and this data can be used to optimize lookups on receive using a companion table called the "keytab". This table records a separate node table reference that can be fetched without any locking using the table index. This logic is handled with the

ieee80211_find_rxnode_withkey() call: if a keytab entry is found using the specified index then it is returned directly; otherwise a normal lookup is done and the keytab entry is written using the specified index. If the specified index is IEEE80211_KEYIX_NONE then a normal lookup is done without a table update.

SEE ALSO

ddb(4), ieee80211(9), ieee80211_proto(9)