NAME

ALTQ - alternate queuing of network packets

SYNOPSIS

options ALTQ

options ALTQ_CBQ
options ALTQ_CODEL
options ALTQ_RED
options ALTQ_RIO
options ALTQ_HFSC
options ALTQ_CDNR
options ALTQ_PRIQ
options ALTQ_FAIRQ

DESCRIPTION

The **ALTQ** system is a framework which provides several disciplines for queuing outgoing network packets. This is done by modifications to the interface packet queues. See altq(9) for details.

The user interface for **ALTQ** is implemented by the pfctl(8) utility, so please refer to the pfctl(8) and the pf.conf(5) man pages for a complete description of the **ALTQ** capabilities and how to use it.

Kernel Options

The following options in the kernel configuration file are related to **ALTQ** operation:

ALTQ	Enable **ALTQ**.
ALTQ_CBQ	Build the "Class Based Queuing" discipline.
ALTQ_CODEL	Build the "Controlled Delay" discipline.
ALTQ_RED	Build the "Random Early Detection" extension.
ALTQ_RIO	Build "Random Early Drop" for input and output.
ALTQ_HFSC	Build the "Hierarchical Packet Scheduler" discipline.
ALTQ_CDNR	Build the traffic conditioner. This option is meaningless at the moment as the conditioner is not used by any of the available disciplines or consumers.
ALTQ_PRIQ	Build the "Priority Queuing" discipline.
ALTQ_FAIRQ	Build the "Fair Queuing" discipline.
ALTQ_NOPCC	Required if the TSC is unusable.
ALTQ_DEBUG	Enable additional debugging facilities.

Note that **ALTQ**-disciplines cannot be loaded as kernel modules. In order to use a certain discipline you have to build it into a custom kernel. The pf(4) interface, that is required for the configuration process

of **ALTQ** can be loaded as a module.

SUPPORTED DEVICES

The driver modifications described in altq(9) are required to use a certain network card with **ALTQ**. They have been applied to the following hardware drivers: ae(4), age(4), alc(4), ale(4), an(4), aue(4), axe(4), bce(4), bfe(4), bge(4), bxe(4), cas(4), dc(4), em(4), epair(4), et(4), fxp(4), gem(4), igb(4), ixgbe(4), jme(4), le(4), liquidio(4), msk(4), mxge(4), my(4), nfe(4), nge(4), npe(4), qlxgb(4), re(4), rl(4), sge(4), sis(4), sk(4), ste(4), stge(4), ti(4), udav(4), vge(4), vr(4), vte(4), and xl(4).

The tun(4), if_bridge(4), if_vlan(4), and ng_iface(4) pseudo drivers also do support **ALTQ**.

An example:

```
altq on igb0 cbq queue { def aq }
queue def bandwidth 90% cbq (default borrow)
queue aq bandwidth 10Mb cbq

pass in on igb0.10 proto udp all queue aq keep state
```

SEE ALSO

pf(4), pf.conf(5), ipfw(8), pfctl(8), altq(9)

HISTORY

The **ALTQ** system first appeared in March 1997 and found home in the KAME project (https://www.kame.net). It was imported to FreeBSD in 5.3 .