

Chapter 25

Configuring Link Services IQ Interfaces

On the Adaptive Services (AS) Physical Interface Card (PIC) and the internal Adaptive Services Module (ASM) in the M7i platform, you can configure link services IQ interfaces (lsq). Link services intelligent queuing (IQ) interfaces are similar to link services interfaces, which are described in “Configuring Link Services and Multilink Interfaces” on page 409. The important difference is that link services IQ interfaces fully support JUNOS class-of-service (CoS) components.

This chapter describes the Layer 2 service package and the CoS capabilities of link services IQ interfaces. For detailed information about Layer 3 services, see the *JUNOS Software Services Interfaces Configuration Guide* and the *JUNOS Software Feature Guide*.

This chapter is organized as follows:

Layer 2 Service Package Capabilities and Interfaces on page 450

AS PIC Link Services IQ CoS Components on page 451

Configuring Fragmentation by Forwarding Class on page 453

Configuring Link-Layer Overhead on page 455

CoS Configuration Tasks on page 456

CoS Configuration Tasks on page 456

Common Uses for the Link Services IQ Interface on page 459

Layer 2 Service Package Capabilities and Interfaces

As described in “Enabling AS PIC Service Packages” on page 158, you can configure the AS PIC and the internal ASM in the M7i platform to use either the Layer 2 or the Layer 3 service package.

When you enable the Layer 2 service package, the AS PIC supports *link services*. On the AS PIC and the ASM, link services includes the following:

JUNOS CoS components—The sections “AS PIC Link Services IQ CoS Components” on page 451 and “Common Uses for the Link Services IQ Interface” on page 459 describe how the JUNOS CoS components work on AS PIC link services IQ interfaces (lsq). For detailed information about JUNOS CoS components, see “CoS Overview” on page 799 and “CoS Configuration Guidelines” on page 813.

Link fragmentation interleaving (LFI) on Frame Relay links using FRF.12 end-to-end fragmentation—The standard for FRF.12 is defined in the specification FRF.12, *Frame Relay Fragmentation Implementation Agreement*.

LFI on Multilink Point-to-Point Protocol (MLPPP) links.

Multilink Frame Relay (MLFR) UNI NNI (FRF.16)—The standard for FRF.16 is defined in the specification FRF.16.1, *Multilink Frame Relay UNI/NNI Implementation Agreement*.

MLPPP (RFC 1990)—The standard for RFC 1990 is defined in the specification RFC 1990, *The PPP Multilink Protocol (MP)*.

Multiclass extension to MLPPP (RFC 2686)—The standard for RFC 2686 is defined in the specification RFC 2686, *The Multi-Class Extension to Multi-Link PPP*. The JUNOS software PPP implementation does not support the negotiation of address field compression and protocol field compression PPP Network Control Protocol (NCP) options, which means that the software always sends a full 4-byte PPP header.

For the link services IQ interface on the AS PIC, the configuration syntax is almost the same as for Multilink and Link Services PICs. The primary difference is the use of the interface-type descriptor lsq instead of ml or ls. When you enable the Layer 2 service package on the AS PIC, the following interfaces are automatically created:

```
gr-fpc/pic/port
ip-fpc/pic/port
lsq-fpc/pic/port
lsq-fpc/pic/port:0
...
lsq-fpc/pic/port:N
mt-fpc/pic/port
pd-fpc/pic/port
pe-fpc/pic/port
sp-fpc/pic/port
vt-fpc/pic/port
```

Interface types `gr`, `ip`, `mt`, `pd`, `pe`, and `vt` are standard tunnel interfaces that are available on the AS PIC whether you enable the Layer 2 or the Layer 3 service package. These tunnel interfaces function the same for both service packages, except that the Layer 2 service package does not support some tunnel functions, as shown in Table 14 on page 159. For more information about tunnel interfaces, see “Configuring Tunnel Interfaces” on page 569.

Interface type `lsq-fpc/pic/port` is the physical link services IQ interface (lsq). Interface types `lsq-fpc/pic/port:0` through `lsq-fpc/pic/port:N` represent FRF.16 bundles. These interface types are created when you include the `mlfr-uni-nni-bundles` statement at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level.



NOTE: Interface type `sp` is created because it is needed by the JUNOS software. For the Layer 2 service package, the `sp` interface is not configurable, but you should not disable it.

AS PIC Link Services IQ CoS Components

For AS PIC link services IQ interfaces (lsq), you can specify a scheduler map for each logical unit. A logical unit represents either an MLPPP bundle or a data-link connection identifier (DLCI) configured on an FRF.16 bundle. The scheduler is applied to the traffic sent to an AS PIC running the Layer 2 link service package.

If you configure a scheduler map on a bundle, you must include the `per-unit-scheduler` statement at the `[edit interfaces lsq-fpc/pic/port]` hierarchy level. If you configure a scheduler map on an FRF.16 DLCI, you must include the `per-unit-scheduler` statement at the `[edit interfaces lsq-fpc/pic/port/channel]` hierarchy level. For more information, see “Associating a Scheduler Map with a DLCI or VLAN” on page 846.

If you need latency guarantees for multiclass or LFI traffic, you must use channelized IQ PICs for the constituent links. With non-IQ PICs, because queuing is not done at the channelized interface level on the constituent links, latency-sensitive traffic might not receive the type of service that it should. Constituent links from the following PICs support latency guarantees:

- Channelized E1 IQ PIC
- Channelized OC3 IQ PIC
- Channelized OC12 IQ PIC
- Channelized STM1 IQ PIC
- Channelized T3 IQ PIC

For scheduling queues on a logical interface, you can configure the following scheduler map properties at the [edit class-of-service schedulers] hierarchy level:

buffer-size—The queue size

priority—The transmit priority (low, high, or strict-high)

transmit-rate—The subscribed transmit rate

When you configure MLPPP and FRF.12 on M-series and T-series routing platforms, you should configure a single scheduler with non-zero percent transmission rates and buffer sizes for queues 0 through 3, and assign this scheduler to the link services IQ interface (lsq) and to each constituent link.

When you configure FRF.16 on M-series and T-series routing platforms, you can assign a single scheduler map to the link services IQ interface (lsq) and to each link services IQ DLCI, or you can assign different scheduler maps to the various DLCIs of the bundle, as shown in “Examples: Configuring an NxT1 Bundle Using FRF.16” on page 469. For the constituent links of an FRF.16 bundle, you do not need to configure a custom scheduler. Because LFI and multiclass are not supported for FRF.16, the traffic from each constituent link is transmitted from queue 0. This means you should allow most of the bandwidth to be used by queue 0. The default scheduler transmission rate and buffer size percentages for queues 0 through 3 are 95, 0, 0, and 5 percent, respectively. This default scheduler sends all user traffic to queue 0 and all network-control traffic to queue 3, and therefore it is well suited to the behavior of FRF.16. If you want, you can configure a custom scheduler that explicitly replicates the 95, 0, 0, and 5 percent queuing behavior, and apply it to the constituent links.

For AS PIC link services IQ interfaces, scheduling properties work as they do in other PICs, except as noted in the following sections.

Scheduler Buffer Size

You can configure the scheduler buffer size in three ways: as a temporal value, as a percentage, and as a remainder. On a single logical interface (MLPPP or an FRF.16 DLCI), each queue can have a different buffer size.

If you specify a temporal value, the queuing algorithm starts dropping packets when it queues more than a computed number of bytes. This number is computed by multiplying the logical interface speed by the temporal value. For MLPPP bundles, logical interface speed is equal to the bundle bandwidth, which is the sum of constituent link speeds minus link-layer overhead. For MLFR FRF.16 DLCIs, logical interface speed is equal to the bundle bandwidth multiplied by the DLCI shaping rate. In all cases, the maximum temporal value is limited to 200 milliseconds.

Buffer size percentages are implicitly converted into temporal values by multiplying the percentage by 200 milliseconds. For example, buffer size specified as `buffer-size percent 20` is the same as a 40-millisecond temporal delay. The link services IQ implementation guarantees 200 milliseconds of buffer delay for all interfaces with T1 and higher speeds. For slower interfaces, it guarantees one second of buffer delay.

The queueing algorithm evenly distributes leftover bandwidth among all queues that are configured with the `buffer-size remainder` statement. The queueing algorithm guarantees enough space in the transmit buffer for two MTU-sized packets.

Scheduler Shaping Rate

You use the shaping rate to set the percentage of total bundle bandwidth that is dedicated to a DLCI. For link services IQ DLCIs, only percentages are accepted, which allows adjustments in response to dynamic changes in bundle bandwidth—for example, when a link goes up or down.

The sum of percentages for each DLCI cannot exceed 100 percent. If the sum of configured percentages is less than 100 percent, the leftover bandwidth is equally divided among DLCIs that do not specify a DLCI scheduler. If none of the DLCIs in an MLFR FRF.16 bundle specify a DLCI scheduler, the total bandwidth is evenly divided across all DLCIs.

Scheduler Transmit Rate

The transmit priority of each queue is determined by the scheduler and the forwarding class. Each queue receives a guaranteed amount of bandwidth specified with the `scheduler transmit-rate` statement.

For scheduling between DLCIs in an MLFR FRF.16 bundle, you can configure a shaping rate for each DLCI. A shaping rate is expressed as a percentage of the aggregate bundle bandwidth. Shaping rate percentages for all DLCIs within a bundle can add up to 100 percent or less. Leftover bandwidth is distributed equally to DLCIs that do not have the `shaping-rate` statement included at the [edit class-of-service interfaces `lsq-fpc/pic/port:channel` unit `logical-unit-number`] hierarchy level.

Configuring Fragmentation by Forwarding Class

For AS PIC link services IQ interfaces (lsq), you can specify fragmentation properties for specific forwarding classes. Traffic on each forwarding class can be either multilink encapsulated (meaning fragmented and sequenced) or nonencapsulated (meaning hashed with no fragmentation). By default, traffic in all forwarding classes is multilink encapsulated.

When you do not configure fragmentation properties for the queues on MLPPP interfaces, the fragmentation threshold you set at the [edit interfaces `interface-name` unit `logical-unit-number` `fragment-threshold`] hierarchy level is the fragmentation threshold for all forwarding classes within the MLPPP interface. For MLFR FRF.16 interfaces, the fragmentation threshold you set at the [edit interfaces `interface-name` `mlfr-uni-nni-bundle-options` `fragment-threshold`] hierarchy level is the fragmentation threshold for all forwarding classes within the MLFR FRF.16 interface.

If you do not set a maximum fragment size anywhere in the configuration, packets are still fragmented if they exceed the smallest maximum transmission unit (MTU) or maximum received reconstructed unit (MRRU) of all the links in the bundle. A nonencapsulated flow uses only one link. If the flow exceeds a single link, then the forwarding class must be multilink encapsulated, unless the packet size exceeds the MTU/MRRU.

Also, if you do not set a maximum fragment size anywhere in the configuration, you can configure the MRRU by including the `mrru` statement at the [edit interfaces *lsq-fpc/pic/port* unit *logical-unit-number*] or [edit interfaces *interface-name* minimum-links] hierarchy level. The MRRU is similar to the MTU, but is specific to link services interfaces. By default, the MRRU size is 1504 bytes, and you can configure it to be from 1500 through 4500 bytes. For more information, see “Configuring MRRU” on page 420.

To configure fragmentation properties on a queue, include the `fragmentation-maps` statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
fragmentation-maps {
  map-name {
    forwarding-class class-name {
      fragment-threshold bytes;
      no-fragmentation;
    }
  }
}
```

To set a per-forwarding class fragmentation threshold, include the `fragment-threshold` statement in the fragmentation map. This statement sets the maximum size of each multilink fragment.

To set traffic on a queue to be nonencapsulated rather than multilink encapsulated, include the `no-fragmentation` statement in the fragmentation map. This statement specifies that an extra fragmentation header is not prepended to the packets received on this queue, and that static link load balancing is used to ensure in-order packet delivery.

For a given forwarding class, you can include either the `fragment-threshold` or `no-fragmentation` statement; they are mutually exclusive.

To associate a fragmentation map with a multilink PPP interface or MLFR FRF.16 DLCI, include the fragmentation-map statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number*] hierarchy level:

```
[edit class-of-service interfaces]
lsq-fpc/pic/port {
  unit logical-unit-number { # Multilink PPP
    fragmentation-map map-name;
  }
}
lsq-fpc/pic/port:channel { # MLFR FRF.16
  unit logical-unit-number {
    fragmentation-map map-name;
  }
}
```

For configuration examples, see “Common Uses for the Link Services IQ Interface” on page 459.

For Link Services PIC link services interfaces (ls), fragmentation maps are not supported. Instead, you can enable LFI by including the interleave-fragments statement at the [edit interfaces *interface-name* unit *logical-unit-number*] hierarchy level. For more information, see “Configuring Link Services Delay-Sensitive Packet Interleaving” on page 417.

Configuring Link-Layer Overhead

Link-layer overhead can cause packet drops on constituent links because of bit stuffing on serial links. Bit stuffing is used to prevent data from being interpreted as control information.

By default, the percentage of total bundle bandwidth to be set aside for link-layer overhead is 4 percent. In most network environments, the average link-layer overhead is 1.6 percent. Therefore, we recommend 4 percent as a safeguard. For more information, see the IETF document *Hash and Stuffing: Overlooked Factors in Network Device Benchmarking* (expires August 29, 2005).

For AS PIC link services IQ interfaces (lsq), you can configure the percentage of bundle bandwidth to be set aside for link-layer overhead. To do this, include the link-layer-overhead statement:

```
link-layer-overhead percent;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name mlfr-uni-nni-bundle-options]

[edit interfaces interface-name unit logical-unit-number]

[edit logical-routers logical-router-name interfaces interface-name
unit logical-unit-number]
```

You can configure the value to be from 0 percent through 50 percent.

CoS Configuration Tasks

To configure link services and CoS on an AS PIC, you must perform the following steps:

1. Enable the Layer 2 service package. You enable service packages per PIC, not per port. When you enable the Layer 2 service package, the entire AS PIC uses the configured package. To enable the Layer 2 service package, include the service-package statement at the [edit chassis fpc *slot-number* pic *pic-number* adaptive-services] hierarchy level, and specify layer-2:

```
[edit chassis fpc slot-number pic pic-number adaptive-services]
service-package layer-2;
```

For more information about AS PIC service packages, see “Enabling AS PIC Service Packages” on page 158 and “Layer 2 Service Package Capabilities and Interfaces” on page 450.

2. Configure a multilink PPP or FRF.16 bundle by combining constituent links into a virtual link, or bundle.

Configuring an MLPPP Bundle

To configure an MLPPP bundle, configure constituent links and bundle properties by including the following statements in the configuration:

```
[edit interfaces interface-name unit logical-unit-number]
encapsulation ppp;
family mlppp {
    bundle lsq-fpc/pic/port.logical-unit-number;
}
```

```
[edit interfaces lsq-fpc/pic/port unit logical-unit-number]
drop-timeout milliseconds;
encapsulation multilink-ppp;
fragment-threshold bytes;
link-layer-overhead percent;
minimum-links number;
mrru bytes;
short-sequence;
family inet {
    address address;
}
```

**Configuring an MLFR
FRF.16 Bundle**

To configure an MLFR FRF.16 bundle, configure constituent links and bundle properties by including the following statements in the configuration:

```
[edit chassis fpc slot-number pic slot-number]
mlfr-uni-nni-bundles number;
```

For more information about the `mlfr-uni-nni-bundles` statement, see the *JUNOS System Basics Configuration Guide*. MLFR FRF.16 uses channels as logical units.

```
[edit interfaces interface-name]
encapsulation multilink-frame-relay-uni-nni;
unit logical-unit-number {
  family mlfr-uni-nni {
    bundle lsq-fpc/pic/port:channel;
  }
}
```

For MLFR FRF.16, you must configure one end as a DCE:

```
[edit interfaces lsq-fpc/pic/port:channel]
encapsulation multilink-frame-relay-uni-nni;
dce;
mlfr-uni-nni-options {
  acknowledge-retries number;
  acknowledge-timer milliseconds;
  action-red-differential-delay (disable-tx | remove-link);
  drop-timeout milliseconds;
  fragment-threshold bytes;
  hello-timer milliseconds;
  link-layer-overhead percent;
  lmi-type (ansi | itu);
  minimum-links number;
  mrru bytes;
  n391 number;
  n392 number;
  n393 number;
  red-differential-delay milliseconds;
  t391 number;
  t392 number;
  yellow-differential-delay milliseconds;
}
unit logical-unit-number {
  dlcid dlci-identifier;
  family inet {
    address address;
  }
}
```

For information about MLFR UNI NNI properties, see “Configuring Link Services and Multilink Interfaces” on page 409.

3. To configure CoS components for each multiple-link bundle, enable per-unit scheduling on the interface, configure a scheduler map, apply the scheduler to each queue, configure a fragmentation map, and apply the fragmentation map to each bundle.

```
[edit interfaces]
lsq-fpc/pic/port {
    per-unit-scheduler; # Enables per-unit scheduling on the bundle
}

[edit class-of-service]
interfaces {
    lsq-fpc/pic/port { # Multilink PPP
        unit logical-unit-number {
            scheduler-map map-name; # Applies scheduler map to each queue
        }
    }
    lsq-fpc/pic/port:channel { # MLFR FRF.16
        unit logical-unit-number {
            # Scheduler map provides scheduling information for
            # the queues within a single DLCI.
            #
            scheduler-map map-name;
            shaping-rate percent percent;
        }
    }
}
forwarding-classes {
    queue queue-number class-name priority (high | low);
}
scheduler-maps {
    map-name {
        forwarding-class class-name scheduler scheduler-name;
    }
}
schedulers {
    scheduler-name {
        buffer-size (percent percentage | remainder | temporal microseconds);
        priority priority-level;
        transmit-rate (percent percentage | rate | remainder) <exact>;
    }
}

fragmentation-maps {
    map-name {
        forwarding-class class-name {
            fragment-threshold bytes;
            no-fragmentation;
        }
    }
}
}
```

Associate a fragmentation map with a multilink PPP interface or MLFR FRF.16 DLCI:

```
[edit class-of-service]
interfaces {
  Isq-fpc/pic/port {
    unit logical-unit-number { # Multilink PPP
      fragmentation-map map-name;
    }
  }
  Isq-fpc/pic/port:channel { # MLFR FRF.16
    unit logical-unit-number {
      fragmentation-map map-name;
    }
  }
}
```

Common Uses for the Link Services IQ Interface

The following sections detail some common uses for the link services IQ interface, with configuration guidelines and a configuration example for each case.

Configuring an NxT1 Bundle Using MLPPP on page 459

Configuring an NxT1 Bundle Using FRF.16 on page 465

Configuring a Single Fractional T1 Using MLPPP and LFI on page 471

Configuring a Single Fractional T1 Using FRF.12 on page 476



NOTE: In Europe and parts of Asia, customers use E1 instead of T1. We are using T1 here as an example, with the understanding that E1 is equally supported.

Configuring an NxT1 Bundle Using MLPPP

To configure an NxT1 bundle using MLPPP, you aggregate N different T1 links into a bundle. The NxT1 bundle is called a logical interface, because it could represent, for example, a routing adjacency. To aggregate T1 links into an MLPPP bundle, include the bundle statement at the [edit interfaces t1-fpc/pic/port unit logical-unit-number family mlppp] hierarchy level:

```
[edit interfaces t1-fpc/pic/port unit logical-unit-number family mlppp]
bundle Isq-fpc/pic/port.logical-unit-number;
```

To configure the link services IQ interface properties, include the following statements in the configuration:

```
[edit interfaces lsq-fpc/pic/port unit logical-unit-number]
drop-timeout milliseconds;
encapsulation multilink-ppp;
fragment-threshold bytes;
link-layer-overhead percent;
minimum-links number;
mrru bytes;
short-sequence;
family inet {
    address address;
}
```

The logical link services IQ interface represents the MLPPP bundle. For the MLPPP bundle, there are four associated queues. A scheduler removes packets from the queues according to a scheduling policy. Typically, you designate one queue to have strict priority, and the remaining queues are serviced in proportion to weights you configure.

For MLPPP, you should assign a single scheduler map to the link services IQ interface (lsq) and to each constituent link. The default scheduler, which assigns 95, 0, 0, and 5 percent bandwidth for the transmission rate and buffer size of queues 0, 1, 2, and 3, is not adequate when you configure LFI or multiclass traffic. Therefore, for MLPPP, you should configure a single scheduler with non-zero percent transmission rates and buffer sizes for queues 0 through 3, and assign this scheduler to the link services IQ interface (lsq) and to each constituent link, as shown in “Example: Configuring an NxT1 Bundle Using MLPPP” on page 463.

If there is more than one link in the bundle, you must include the per-unit-scheduler statement at the [edit interfaces lsq-fpc/pic/port] hierarchy level:

```
[edit interfaces]
lsq-fpc/pic/port {
    per-unit-scheduler;
}
```

To configure and apply the scheduling policy, include the following statements in the configuration:

```
[edit class-of-service]
interfaces {
    t1-fpc/pic/port unit logical-unit-number {
        scheduler-map map-name;
    }
}
forwarding-classes {
    queue queue-number class-name;
}
```

```

scheduler-maps {
  map-name {
    forwarding-class class-name scheduler scheduler-name;
  }
}
schedulers {
  scheduler-name {
    buffer-size (percent percentage | remainder | temporal microseconds);
    priority priority-level;
    transmit-rate (rate | percent percentage | remainder) <exact>;
  }
}

```



NOTE: The Link Services IQ PIC does not support scheduler drop profiles (RED).

For link services IQ interfaces, a strict-high-priority queue might starve the other three queues because traffic in a strict-high-priority queue is transmitted before any other queue is serviced. This implementation is unlike the standard JUNOS CoS implementation in which a strict-high-priority queue does round-robin with high-priority queues, as described in “Configuring Priority Scheduling” on page 838.

After the scheduler transmits a packet from a queue, certain actions are taken. The action depends on the type of queue from which the packet came. There are two types of queues: multilink encapsulated (meaning fragmented and sequenced) and nonencapsulated (meaning hashed with no fragmentation). Each of the queues can be designated as either multilink encapsulated or nonencapsulated, independently of each other. By default, traffic in all forwarding classes is multilink encapsulated. To configure packet fragmentation handling on a queue, include the fragmentation-maps statement at the [edit class-of-service] hierarchy level:

```

[edit class-of-service]
fragmentation-maps {
  map-name {
    forwarding-class class-name {
      fragment-threshold bytes;
      no-fragmentation;
    }
  }
}

```

For *NxT1* bundles using MLPPP, there is no advantage to using a nonencapsulated queue instead of a multilink-encapsulated queue. This is because the byte-wise load balancing used in multilink-encapsulated queues is superior to the flow-wise load balancing used in nonencapsulated queues. All other considerations are equal in this particular scenario. Therefore, we recommend that you configure all queues to be multilink encapsulated. You do this by including the `fragment-threshold` statement in the configuration. If you do choose to set traffic on a queue to be nonencapsulated rather than multilink encapsulated, you can do so by including the `no-fragmentation` statement in the fragmentation map. For more information about fragmentation maps, see “Configuring Fragmentation by Forwarding Class” on page 453.

When a packet is removed from a multilink-encapsulated queue, the software gives the packet an MLPPP header. The MLPPP header contains a sequence number field, which is filled with the next available sequence number from a counter. The software then places the packet on one of the *N* different T1 links. The link is chosen on a packet-by-packet basis, to balance the load across the various T1 links.

If the packet exceeds the minimum link MTU, or if a queue has a fragmentation threshold configured at the `[edit class-of-service fragmentation-maps map-name forwarding-class class-name]` hierarchy level, the software splits the packet into two or more fragments, which are assigned consecutive multilink sequence numbers. The outgoing link for each fragment is selected independent of all other fragments.

If you do not include the `fragment-threshold` statement in the fragmentation map, the fragmentation threshold you set at the `[edit interfaces interface-name unit logical-unit-number]` hierarchy level is the default for all forwarding classes. If you do not set a maximum fragment size anywhere in the configuration, packets are fragmented if they exceed the smallest MTU of all the links in the bundle.

Also, if you do not set a maximum fragment size anywhere in the configuration, you can configure the MRRU by including the `mrru` statement at the `[edit interfaces lsq-fpc/pic/port unit logical-unit-number]` hierarchy level. The MRRU is similar to the MTU, but is specific to link services interfaces. By default, the MRRU size is 1504 bytes, and you can configure it to be from 1500 through 4500 bytes. For more information, see “Configuring MRRU” on page 420.

When a packet is removed from a nonencapsulated queue, it is not given an MLPPP header. Rather, the packet is transmitted with a plain PPP header. Because there is no MLPPP header, there is no sequence number information. Therefore, the software must take special measures to avoid packet reordering. To avoid packet reordering, the software places the packet on one of the *N* different T1 links. The link is determined by hashing the values in the header. For IP, the software computes the hash based on source address, destination address, and IP protocol. For MPLS, the software computes the hash based on up to five MPLS labels, or four MPLS labels and the IP header.

For UDP and TCP, the software computes the hash based on the source and destination ports, as well as the source and destination IP addresses. This guarantees that all packets belonging to the same TCP/UDP flow always pass through the same T1 link, and therefore cannot be reordered. However, it does not guarantee that the load on the various T1 links is balanced. If there is a large number of flows, load balancing is usually not a problem.

The N different T1 interfaces link to another router, which is not necessarily a Juniper Networks router. The router at the far end gathers packets from all the T1 links. If a packet has an MLPPP header, the sequence number field is used to put the packet back into sequence number order. If the packet has a plain PPP header, the software accepts the packet in the order in which it arrives and makes no attempt to reassemble or reorder the packet.

Example: Configuring an NxT1 Bundle Using MLPPP

Configure an N xT1 bundle using MLPPP:

```
[edit chassis]
fpc 1 {
  pic 3 {
    adaptive-services {
      service-package layer-2;
    }
  }
}

[edit interfaces]
t1-0/0/0 {
  unit 0 {
    encapsulation ppp;
    family mlppp {
      bundle lsq-1/3/0.1; # This adds t1-0/0/0 to the specified bundle.
    }
  }
}
t1-0/0/1 {
  unit 0 {
    encapsulation ppp;
    family mlppp {
      bundle lsq-1/3/0.1;
    }
  }
}
lsq-1/3/0 {
  unit 1 { # This is the virtual link that concatenates multiple T1s.
    encapsulation multilink-ppp;
    drop-timeout 1000;
    fragment-threshold 128;
    link-layer-overhead 0.5;
    minimum-links 2;
    mrru 4500;
    short-sequence;
    family inet {
      address 1.2.3.4/24;
    }
  }
}

[edit interfaces]
lsq-1/3/0 {
  per-unit-scheduler;
}
```

```

[edit class-of-service]
interfaces {
  lsq-1/3/0 { # multilink PPP constituent link
    unit 0 {
      scheduler-map sched-map1;
    }
  }
  t1-0/0/0 { # multilink PPP constituent link
    unit 0 {
      scheduler-map sched-map1;
    }
  }
  t1-0/0/1 { # multilink PPP constituent link
    unit 0 {
      scheduler-map sched-map1;
    }
  }
}
forwarding-classes {
  queue 0 be;
  queue 1 ef;
  queue 2 af;
  queue 3 nc;
}
scheduler-maps {
  sched-map1 {
    forwarding-class af scheduler af-scheduler;
    forwarding-class be scheduler be-scheduler;
    forwarding-class ef scheduler ef-scheduler;
    forwarding-class nc scheduler nc-scheduler;
  }
}
schedulers {
  af-scheduler {
    transmit-rate percent 30;
    buffer-size percent 30;
    priority low;
  }
  be-scheduler {
    transmit-rate percent 25;
    buffer-size percent 25;
    priority low;
  }
  ef-scheduler {
    transmit-rate percent 40;
    buffer-size percent 40;
    priority strict-high; # voice queue
  }
  nc-scheduler {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority high;
  }
}
}

```

```

fragmentation-maps {
  fragmap-1 {
    forwarding-class be {
      fragment-threshold 180;
    }
    forwarding-class ef {
      fragment-threshold 100;
    }
  }
}

[edit interfaces]
lsq-1/3/0 {
  unit 0 {
    fragmentation-map fragmap-1;
  }
}

```

Configuring an NxT1 Bundle Using FRF.16

To configure an NxT1 bundle using FRF.16, you aggregate N different T1 links into a bundle. The NxT1 bundle carries a potentially large number of Frame Relay PVCs, identified by their DLCIs. Each DLCI is called a logical interface, because it could represent, for example, a routing adjacency.

To aggregate T1 links into an FRF.16 bundle, include the `mlfr-uni-nni-bundles` statement at the `[edit chassis fpc slot-number pic slot-number]` hierarchy level and include the bundle statement at the `[edit interfaces t1-fpc/pic/port unit logical-unit-number family mlfr-uni-nni]` hierarchy level:

```

[edit chassis fpc slot-number pic slot-number]
mlfr-uni-nni-bundles number;

[edit interfaces t1-fpc/pic/port unit logical-unit-number family mlfr-uni-nni]
bundle lsq-fpc/pic/port:channel;

```

To configure the link services IQ interface properties, include the following statements in the configuration:

```
[edit interfaces lsq-fpc/pic/port:channel]
encapsulation multilink-frame-relay-uni-nni;
dce;
mlfr-uni-nni-options {
  acknowledge-retries number;
  acknowledge-timer milliseconds;
  action-red-differential-delay (disable-tx | remove-link);
  drop-timeout milliseconds;
  fragment-threshold bytes;
  hello-timer milliseconds;
  link-layer-overhead percent;
  lmi-type (ansi | itu);
  minimum-links number;
  mrru bytes;
  n391 number;
  n392 number;
  n393 number;
  red-differential-delay milliseconds;
  t391 number;
  t392 number;
  yellow-differential-delay milliseconds;
}
unit logical-unit-number {
  dlcid dlci-identifier;
  family inet {
    address address;
  }
}
```

The link services IQ channel represents the FRF.16 bundle. For FRF.16, each DLCI has four associated queues. A scheduler removes packets from the queues according to a scheduling policy. On the link services IQ interface, you typically designate one queue to have strict priority. The remaining queues are serviced in proportion to weights you configure.

For link services IQ interfaces, a strict-high-priority queue might starve the other three queues because traffic in a strict-high-priority queue is transmitted before any other queue is serviced. This implementation is unlike the standard JUNOS CoS implementation in which a strict-high-priority queue does round-robin with high-priority queues, as described in “Configuring Priority Scheduling” on page 838.

If there is more than one link in the bundle, you must include the per-unit-scheduler statement at the [edit interfaces lsq-fpc/pic/port:channel] hierarchy level:

```
[edit interfaces]
lsq-fpc/pic/port:channel {
  per-unit-scheduler;
}
```

For FRF.16, you can assign a single scheduler map to the link services IQ interface (lsq) and to each link services IQ DLCI, or you can assign different scheduler maps to the various DLCIs of the bundle, as shown in “Examples: Configuring an NxT1 Bundle Using FRF.16” on page 469.

For the constituent links of an FRF.16 bundle, you do not need to configure a custom scheduler. Because LFI and multiclass are not supported for FRF.16, the traffic from each constituent link is transmitted from queue 0. This means you should allow most of the bandwidth to be used by queue 0. The default scheduler transmission rate and buffer size percentages for queues 0 through 3 are 95, 0, 0, and 5 percent, respectively. This default scheduler sends all user traffic to queue 0 and all network-control traffic to queue 3, and therefore it is well suited to the behavior of FRF.16. If desired, you can configure a custom scheduler that explicitly replicates the 95, 0, 0, and 5 percent queuing behavior, and apply it to the constituent links.

To configure and apply the scheduling policy, include the following statements in the configuration:

```
[edit class-of-service]
interfaces {
  lsq-fpc/pic/port:channel {
    unit logical-unit-number {
      scheduler-map map-name;
    }
  }
}
forwarding-classes {
  queue queue-number class-name;
}
scheduler-maps {
  map-name {
    forwarding-class class-name scheduler scheduler-name;
  }
}
}
schedulers {
  scheduler-name {
    buffer-size (percent percentage | remainder | temporal microseconds);
    priority priority-level;
    transmit-rate (rate | percent percentage | remainder) <exact>;
  }
}
```



NOTE: The Link Services IQ PIC does not support scheduler drop profiles (RED).

To configure packet fragmentation handling on a queue, include the `fragmentation-maps` statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
fragmentation-maps {
  map-name {
    forwarding-class class-name {
      fragment-threshold bytes;
    }
  }
}
```

For FRF.16 traffic, only multilink-encapsulated (meaning fragmented and sequenced) queues are supported. This is the default queuing behavior for all forwarding classes. FRF.16 does not allow for nonencapsulated traffic because the protocol requires that all packets carry the fragmentation header. This means that if a large packet is split into multiple fragments, the fragments must have consecutive sequential numbers. Therefore, you cannot include the `no-fragmentation` statement at the [edit class-of-service fragmentation-maps *map-name* forwarding-class *class-name*] hierarchy level for FRF.16 traffic. For FRF.16, if you want to carry voice or any other latency-sensitive traffic, you should not use slow links. For T1 speeds and above, the serialization delay is small enough so that you do not need to use explicit LFI.

When a packet is removed from a multilink-encapsulated queue, the software gives the packet an FRF.16 header. The FRF.16 header contains a sequence number field, which is filled with the next available sequence number from a counter. The software then places the packet on one of the *N* different T1 links. The link is chosen on a packet-by-packet basis, to balance the load across the various T1 links.

If the packet exceeds the minimum link MTU, or if a queue has a fragmentation threshold configured at the [edit class-of-service fragmentation-maps *map-name* forwarding-class *class-name*] hierarchy level, the software splits the packet into two or more fragments, which are assigned consecutive multilink sequence numbers. The outgoing link for each fragment is selected independent of all other fragments.

If you do not include the `fragment-threshold` statement in the fragmentation map, the fragmentation threshold you set at the [edit interfaces *interface-name* unit *logical-unit-number*] or [edit interfaces *interface-name* mlfr-uni-nni-bundle-options] hierarchy level is the default for all forwarding classes. If you do not set a maximum fragment size anywhere in the configuration, packets are fragmented if they exceed the smallest MTU of all the links in the bundle.

Also, if you do not set a maximum fragment size anywhere in the configuration, you can configure the MRRU by including the `mrru` statement at the [edit interfaces *lsq-fpc/pic/port* unit *logical-unit-number*] or [edit interfaces *interface-name* mlfr-uni-nni-bundle-options] hierarchy level. The MRRU is similar to the MTU, but is specific to link services interfaces. By default, the MRRU size is 1504 bytes, and you can configure it to be from 1500 through 4500 bytes. For more information, see “Configuring MRRU” on page 420.

The *N* different T1 interfaces link to another router, which is not necessarily a Juniper Networks router. The router at the far end gathers packets from all the T1 links. Because each packet has an FRF.16 header, the sequence number field is used to put the packet back into sequence number order.

Examples: Configuring an NxT1 Bundle Using FRF.16

Configure an NxT1 Bundle using FRF.16 with multiple CoS scheduler maps:

```
[edit chassis fpc 1 pic 3]
adaptive-services {
    service-package layer-2;
}
mlfr-uni-nni-bundles 2; # Creates channelized LSQ interfaces/FRF.16 bundles.

[edit interfaces]
t1-0/0/0 {
    encapsulation multilink-frame-relay-uni-nni;
    unit 0 {
        family mlfr-uni-nni {
            bundle lsq-1/3/0:1;
        }
    }
}
t1-0/0/1 {
    encapsulation multilink-frame-relay-uni-nni;
    unit 0 {
        family mlfr-uni-nni {
            bundle lsq-1/3/0:1;
        }
    }
}

lsq-1/3/0:1 { # Bundle link consisting of t1-0/0/0 and t1-0/0/1
    per-unit-scheduler;
    encapsulation multilink-frame-relay-uni-nni;
    dce; # One end needs to be configured as DCE.
    mlfr-uni-nni-bundle-options {
        drop-timeout 180;
        fragment-threshold 64;
        hello-timer 180;
        minimum-links 2;
        mrru 3000;
        link-layer-overhead 0.5;
    }
    unit 0 {
        dlcI 26; # Each logical unit maps a single DLCI.
        family inet {
            address 1.2.3.4/24;
        }
    }
    unit 1 {
        dlcI 42;
        family inet {
            address 10.20.30.40/24;
        }
    }
}
```

```

    unit 2 {
      dlcI 69;
      family inet {
        address 10.20.30.40/24;
      }
    }
  }

[edit class-of-service]
scheduler-maps {
  sched-map-Isq0 {
    forwarding-class af scheduler af-scheduler-Isq0;
    forwarding-class be scheduler be-scheduler-Isq0;
    forwarding-class ef scheduler ef-scheduler-Isq0;
    forwarding-class nc scheduler nc-scheduler-Isq0;
  }
  sched-map-Isq1 {
    forwarding-class af scheduler af-scheduler-Isq1;
    forwarding-class be scheduler be-scheduler-Isq1;
    forwarding-class ef scheduler ef-scheduler-Isq1;
    forwarding-class nc scheduler nc-scheduler-Isq1;
  }
}

schedulers {
  af-scheduler-Isq0 {
    transmit-rate percent 60;
    buffer-size percent 60;
    priority low;
  }
  be-scheduler-Isq0 {
    transmit-rate percent 30;
    buffer-size percent 30;
    priority low;
  }
  ef-scheduler-Isq0 {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority strict-high;
  }
  nc-scheduler-Isq0 {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority high;
  }
  af-scheduler-Isq1 {
    transmit-rate percent 50;
    buffer-size percent 50;
    priority low;
  }
  be-scheduler-Isq1 {
    transmit-rate percent 30;
    buffer-size percent 30;
    priority low;
  }
}

```

```

ef-scheduler-lsq1 {
    transmit-rate percent 15;
    buffer-size percent 15;
    priority strict-high;
}
nc-scheduler-lsq1 {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority high;
}
}

interfaces {
    lsq-1/3/0:1 { # MLFR FRF.16
        unit 0 {
            scheduler-map sched-map-lsq0;
        }
        unit 1 {
            scheduler-map sched-map-lsq1;
        }
    }
}

```

Configuring a Single Fractional T1 Using MLPPP and LFI

When you configure a single fractional T1, the fractional T1 is called a logical interface, because it could represent, for example, a routing adjacency.

For the logical interface, there are four associated queues. A scheduler removes packets from the queues according to a scheduling policy. Typically, you designate one queue to have strict priority, and the remaining queues are serviced in proportion to weights you configure.

To configure a single fractional T1 using MLPPP and LFI, you associate one DS0 (fractional T1) interface with a link services IQ interface. The fractional T1 interface is called a logical interface, because it could represent, for example, a routing adjacency. To associate a fractional T1 interface with a link services IQ interface, include the bundle statement at the [edit interfaces *ds-fpc/pic/port:channel* unit *logical-unit-number* family mlppp] hierarchy level:

```

[edit interfaces ds-fpc/pic/port:channel unit logical-unit-number family mlppp]
bundle lsq-fpc/pic/port.logical-unit-number;

```

To configure the link services IQ interface properties, include the following statements in the configuration:

```
[edit interfaces lsq-fpc/pic/port unit logical-unit-number]
drop-timeout milliseconds;
encapsulation multilink-ppp;
fragment-threshold bytes;
link-layer-overhead percent;
minimum-links number;
mrru bytes;
short-sequence;
family inet {
    address address;
}
```

The logical link services IQ interface represents the MLPPP bundle. For the MLPPP bundle, there are four associated queues. A scheduler removes packets from the queues according to a scheduling policy. Typically, you designate one queue to have strict priority, and the remaining queues are serviced in proportion to weights you configure.

For MLPPP, you should assign a single scheduler map to the link services IQ interface (lsq) and to each constituent link. The default scheduler, which assigns 95, 0, 0, and 5 percent bandwidth for the transmission rate and buffer size of queues 0, 1, 2, and 3, is not adequate when you configure LFI or multiclass traffic. Therefore, for MLPPP, you should provide some non-zero percentage for the transmission rates and buffer sizes of queues 0 through 3, and assign this scheduler to the link services IQ interface (lsq) and to each constituent link, as shown in “Example: Configuring a Single Fractional T1 Using MLPPP and LFI” on page 474.

To configure and apply the scheduling policy, include the following statements in the configuration:

```
[edit class-of-service]
interfaces {
    ds-fpc/pic/port.channel {
        scheduler-map map-name;
    }
}
forwarding-classes {
    queue queue-number class-name;
}
scheduler-maps {
    map-name {
        forwarding-class class-name scheduler scheduler-name;
    }
}
schedulers {
    scheduler-name {
        buffer-size (percent percentage | remainder | temporal microseconds);
        priority priority-level;
        transmit-rate (rate | percent percentage | remainder) <exact>;
    }
}
```



NOTE: The Link Services IQ PIC does not support scheduler drop profiles (RED).

For link services IQ interfaces, a strict-high-priority queue might starve all the other queues because traffic in a strict-high-priority queue is transmitted before any other queue is serviced. This implementation is unlike the standard JUNOS CoS implementation in which a strict-high-priority queue receives infinite credits and does round-robin with high-priority queues, as described in “Configuring Priority Scheduling” on page 838.

After the scheduler dequeues a packet, certain actions are taken. The action depends on the type of queue from which the packet came. There are two types of queues: multilink encapsulated (meaning fragmented and sequenced) and nonencapsulated (meaning hashed with no fragmentation). Each of the queues can be designated as either multilink encapsulated or nonencapsulated, independently of each other. By default, traffic in all forwarding classes is multilink encapsulated. To configure packet fragmentation handling on a queue, include the fragmentation-maps statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
fragmentation-maps {
  map-name {
    forwarding-class class-name {
      fragment-threshold bytes;
      no-fragmentation;
    }
  }
}
```

If you require the queue to transmit small packets with low latency, you should configure the queue to be nonencapsulated. You do this by including the no-fragmentation statement in the configuration. If you require the queue to transmit large packets with normal latency, you should configure a queue to be multilink encapsulated. You do this by including the fragment-threshold statement in the configuration. When you require the queue to transmit large packets with low latency, we recommend using a faster link and configuring the queue to be nonencapsulated. For more information about fragmentation maps, see “Configuring Fragmentation by Forwarding Class” on page 453.

When a packet is removed from a multilink-encapsulated queue, it is fragmented. If the packet exceeds the minimum link MTU, or if a queue has a fragmentation threshold configured at the [edit class-of-service fragmentation-maps map-name forwarding-class class-name] hierarchy level, the software splits the packet into two or more fragments, which are assigned consecutive multilink sequence numbers.

If you do not include the fragment-threshold statement in the fragmentation map, the fragmentation threshold you set at the [edit interfaces interface-name unit logical-unit-number] hierarchy level is the default for all forwarding classes. If you do not set a maximum fragment size anywhere in the configuration, packets are fragmented if they exceed the smallest MTU of all the links in the bundle.

Also, if you do not set a maximum fragment size anywhere in the configuration, you can configure the MRRU by including the `mrru` statement at the `[edit interfaces lsq-fpc/pic/port unit logical-unit-number]` hierarchy level. The MRRU is similar to the MTU, but is specific to link services interfaces. By default, the MRRU size is 1504 bytes, and you can configure it to be from 1500 through 4500 bytes. For more information, see “Configuring MRRU” on page 420.

When a packet is removed from a multilink-encapsulated queue, the software gives the packet an MLPPP header. The MLPPP header contains a sequence number field, which is filled with the next available sequence number from a counter. The software then places the packet on fractional T1 link, although not necessarily consecutively. Traffic from another queue might be interleaved between two fragments in the manner described below.

When a packet is removed from a nonencapsulated queue, it is not given an MLPPP header. Rather, the packet is transmitted with a plain PPP header. The packet is then placed on the fractional T1 link as soon as possible. If necessary, the packet is placed between the fragments of a packet from another queue.

The fractional T1 interface links to another router, which is not necessarily a Juniper Networks router. The router at the far end gathers packets from the fractional T1 link. If a packet has an MLPPP header, the software assumes the packet is a fragment of a larger packet, and the fragment number field is used to reassemble the larger packet. If the packet has a plain PPP header, the software accepts the packet in the order in which it arrives and makes no attempt to reassemble or reorder the packet.

Example: Configuring a Single Fractional T1 Using MLPPP and LFI

```
[edit interfaces]
lsq-0/2/0 {
  per-unit-scheduler;
  unit 0 {
    encapsulation multilink-ppp;
    link-layer-overhead 0.5;
    family inet {
      address 10.40.1.1/30;
    }
  }
}
ct3-1/0/0 {
  partition 1 interface-type ct1;
}
ct1-1/0/0:1 {
  partition 1 timeslots 1-2 interface-type ds;
}
ds-1/0/0:1:1 {
  encapsulation ppp;
  unit 0 {
    family mlppp {
      bundle lsq-0/2/0.0;
    }
  }
}
```

```

[edit class-of-service]
interfaces {
  ds-1/0/0:1:1 { # multilink PPP constituent link
    unit 0 {
      scheduler-map sched-map1;
    }
  }
}
forwarding-classes {
  queue 0 be;
  queue 1 ef;
  queue 2 af;
  queue 3 nc;
}
scheduler-maps {
  sched-map1 {
    forwarding-class af scheduler af-scheduler;
    forwarding-class be scheduler be-scheduler;
    forwarding-class ef scheduler ef-scheduler;
    forwarding-class nc scheduler nc-scheduler;
  }
}
schedulers {
  af-scheduler {
    transmit-rate percent 20;
    buffer-size percent 20;
    priority low;
  }
  be-scheduler {
    transmit-rate percent 20;
    buffer-size percent 20;
    priority low;
  }
  ef-scheduler {
    transmit-rate percent 50;
    buffer-size percent 50;
    priority strict-high; # voice queue
  }
  nc-scheduler {
    transmit-rate percent 10;
    buffer-size percent 10;
    priority high;
  }
}

```

```

fragmentation-maps {
  fragmap-1 {
    forwarding-class af {
      no-fragmentation;
    }
    forwarding-class be {
      fragment-threshold 180;
    }
    forwarding-class ef {
      fragment-threshold 100;
    }
  }
}

[edit interfaces]
lsq-0/2/0 {
  unit 0 {
    fragmentation-map fragmap-1;
  }
}

```

Configuring a Single Fractional T1 Using FRF.12

When you configure a single fractional T1, the fractional T1 carries a potentially large number of Frame Relay PVCs identified by their DLCIs. Each DLCI is called a logical interface, because it could represent, for example, a routing adjacency. To associate the DS0 interface with a link services IQ interface, include the bundle statement at the [edit interfaces *ds-fpc/pic/port:channel* unit *logical-unit-number* family *mfr-end-to-end*] hierarchy level:

```

[edit interfaces ds-fpc/pic/port:channel unit logical-unit-number
family mfr-end-to-end]
bundle lsq-fpc/pic/port.logical-unit-number;

```

To configure the link services IQ interface properties, include the following statements in the configuration:

```

[edit interfaces lsq-fpc/pic/port unit logical-unit-number]
drop-timeout milliseconds;
encapsulation multilink-frame-relay-end-to-end;
fragment-threshold bytes;
link-layer-overhead percent;
minimum-links number;
mrru bytes;
short-sequence;
family inet {
  address address;
}

```

For each logical interface, there are four associated queues. A scheduler removes packets from the queues according to a scheduling policy. Typically, you designate one queue to have strict priority, and the remaining queues are serviced in proportion to weights you configure.

The logical link services IQ interface represents the FRF.12 bundle. For the FRF.12 bundle, there are four associated queues. A scheduler removes packets from the queues according to a scheduling policy. Typically, you designate one queue to have strict priority, and the remaining queues are serviced in proportion to weights you configure.

For FRF.12, you should assign a single scheduler map to the link services IQ interface (lsq) and to each constituent link. The default scheduler, which assigns 95, 0, 0, and 5 percent bandwidth for the transmission rate and buffer size of queues 0, 1, 2, and 3, is not adequate when you configure LFI or multiclass traffic. Therefore, for FRF.12, you should configure a single scheduler with non-zero percent transmission rates and buffer sizes for queues 0 through 3, and assign this scheduler to the link services IQ interface (lsq) and to each constituent link, as shown in “Examples: Configuring a Single Fractional T1 Using FRF.12” on page 479.

To configure and apply the scheduling policy, include the following statements in the configuration:

```
[edit class-of-service]
interfaces {
  ds-fpc/pic/port.channel {
    scheduler-map map-name;
  }
}
forwarding-classes {
  queue queue-number class-name;
}
scheduler-maps {
  map-name {
    forwarding-class class-name scheduler scheduler-name;
  }
}
schedulers {
  scheduler-name {
    buffer-size (percent percentage | remainder | temporal microseconds);
    priority priority-level;
    transmit-rate (rate | percent percentage | remainder) <exact>;
  }
}
```



NOTE: The Link Services IQ PIC does not support scheduler drop profiles (RED).

For link services IQ interfaces, a strict-high-priority queue might starve the other three queues because traffic in a strict-high-priority queue is transmitted before any other queue is serviced. This implementation is unlike the standard JUNOS CoS implementation in which a strict-high-priority queue does round-robin with high-priority queues, as described in “Configuring Priority Scheduling” on page 838.

After the scheduler dequeues a packet, certain actions are taken. The action depends on the type of queue from which the packet came. There are two types of queues: multilink encapsulated (meaning fragmented and sequenced) and nonencapsulated (meaning hashed with no fragmentation). Each of the queues can be designated as either multilink encapsulated or nonencapsulated, independently of each other. By default, traffic in all forwarding classes is multilink encapsulated. To configure packet fragmentation handling on a queue, include the fragmentation-maps statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
fragmentation-maps {
  map-name {
    forwarding-class class-name {
      fragment-threshold bytes;
      no-fragmentation;
    }
  }
}
```

If you require the queue to transmit small packets with low latency, you should configure the queue to be nonencapsulated. You do this by including the no-fragmentation statement in the configuration. If you require the queue to transmit large packets with normal latency, you should configure a queue to be multilink encapsulated. You do this by including the fragment-threshold statement in the configuration. When you require the queue to transmit large packets with low latency, we recommend using a faster link and configuring the queue to be nonencapsulated. For more information about fragmentation maps, see “Configuring Fragmentation by Forwarding Class” on page 453.

When a packet is removed from a multilink-encapsulated queue, it is fragmented. If the packet exceeds the minimum link MTU, or if a queue has a fragmentation threshold configured at the [edit class-of-service fragmentation-maps map-name forwarding-class class-name] hierarchy level, the software splits the packet into two or more fragments, which are assigned consecutive multilink sequence numbers.

If you do not include the fragment-threshold statement in the fragmentation map, the fragmentation threshold you set at the [edit interfaces interface-name unit logical-unit-number] hierarchy level is the default for all forwarding classes. If you do not set a maximum fragment size anywhere in the configuration, packets are fragmented if they exceed the smallest MTU of all the links in the bundle.

Also, if you do not set a maximum fragment size anywhere in the configuration, you can configure the MRRU by including the mrru statement at the [edit interfaces lsq-fpc/pic/port unit logical-unit-number] hierarchy level. The MRRU is similar to the MTU, but is specific to link services interfaces. By default, the MRRU size is 1504 bytes, and you can configure it to be from 1500 through 4500 bytes. For more information, see “Configuring MRRU” on page 420.

When a packet is removed from a multilink-encapsulated queue, the software gives the packet an FRF.12 header. The FRF.12 header contains a sequence number field, which is filled with the next available sequence number from a counter. The software then places the packet on fractional T1 link, although not necessarily consecutively. Traffic from another queue might be interleaved between two fragments in the manner described below.

When a packet is removed from a nonencapsulated queue, it is not given an FRF.12 header. Rather, the packet is transmitted with a plain Frame Relay header. The packet is then placed on the fractional T1 link as soon as possible. If necessary, the packet is placed between the fragments of a packet from another queue.

The fractional T1 interface links to another router, which is not necessarily a Juniper Networks router. The router at the far end gathers packets from the fractional T1 link. If a packet has an FRF.12 header, the software assumes the packet is a fragment of a larger packet, and the fragment number field is used to reassemble the larger packet. If the packet has a plain Frame Relay header, the software accepts the packet in the order in which it arrives and makes no attempt to reassemble or reorder the packet.

A whole packet from a nonencapsulated queue can be placed between fragments of a multilink-encapsulated queue. However, fragments from one multilink-encapsulated queue cannot interleave with fragments from another multilink-encapsulated queue. This is not a limitation of the JUNOS software implementation. Rather, it is the intent of the specification FRF.12, *Frame Relay Fragmentation Implementation Agreement*. Hypothetically, if fragments from two different queues are interleaved, the header fields do not have enough information to separate the fragments.

Examples: Configuring a Single Fractional T1 Using FRF.12

Configure FRF.12 and a single fractional T1, with fragmentation only.

Also configure FRF.12 and a single fractional T1, with fragmentation and LFI.

FRF.12 with Fragmentation and without LFI

This example shows a 128k DSO interface. There is one traffic stream on ge-0/0/0, which is classified into queue 0 (be). Packets are fragmented in the link services IQ interface (lsq) according to the threshold configured in the fragmentation map.

```
[edit chassis]
fpc 0 {
  pic 3 {
    adaptive-services {
      service-package layer-2;
    }
  }
}

[edit interfaces]
ge-0/0/0 {
  unit 0 {
    family inet {
      address 20.1.1.1/24 {
        arp 20.1.1.2 mac 00.90.1b.12.34.56;
      }
    }
  }
}
ce1-0/2/0 {
  partition 1 timeslots 1-2 interface-type ds;
}
```

```

ds-0/2/0:1 {
  no-keepalives;
  dce;
  encapsulation frame-relay;
  unit 0 {
    dlci 100;
    family mfr-end-to-end {
      bundle lsq-0/3/0.0;
    }
  }
}
lsq-0/3/0 {
  unit 0 {
    encapsulation multilink-frame-relay-end-to-end;
    family inet {
      address 10.200.0.78/30;
    }
  }
}
fxp0 {
  unit 0 {
    family inet {
      address 172.29.1.162/24;
    }
  }
}
lo0 {
  unit 0 {
    family inet {
      address 127.0.0.1/32;
    }
  }
}

[edit class-of-service]
forwarding-classes {
  queue 0 be;
  queue 1 ef;
  queue 2 af;
  queue 3 nc;
}
interfaces {
  lsq-0/3/0 {
    unit 0 {
      fragmentation-map map1;
    }
  }
}
fragmentation-maps {
  map1 {
    forwarding-class {
      be {
        fragment-threshold 160;
      }
    }
  }
}
}

```

**FRF.12 with
Fragmentation and LFI**

This example shows a 512k DS0 bundle and four traffic streams on ge-0/0/0 that are classified into four queues. The fragment size is 160 for queue 0, queue 1, and queue 2. The voice stream on queue 3 has LFI configured.

```
[edit chassis]
fpc 0 {
  pic 3 {
    adaptive-services {
      service-package layer-2;
    }
  }
}
[edit interfaces]
ge-0/0/0 {
  unit 0 {
    family inet {
      address 20.1.1.1/24 {
        arp 20.1.1.2 mac 00.90.1b.12.34.56;
      }
    }
  }
}
ce1-0/2/0 {
  partition 1 timeslots 1-8 interface-type ds;
}
ds-0/2/0:1 {
  no-keepalives;
  dce;
  encapsulation frame-relay;
  unit 0 {
    dlci 100;
    family mlfr-end-to-end {
      bundle lsq-0/3/0.0;
    }
  }
}
lsq-0/3/0 {
  unit 0 {
    encapsulation multilink-frame-relay-end-to-end;
    family inet {
      address 10.200.0.78/30;
    }
  }
}
```

```

[edit class-of-service]
classifiers {
  inet-precedence ge-interface-classifier {
    forwarding-class be {
      loss-priority low code-points 000;
    }
    forwarding-class ef {
      loss-priority low code-points 010;
    }
    forwarding-class af {
      loss-priority low code-points 100;
    }
    forwarding-class nc {
      loss-priority low code-points 110;
    }
  }
}
forwarding-classes {
  queue 0 be;
  queue 1 ef;
  queue 2 af;
  queue 3 nc;
}
interfaces {
  lsq-0/3/0 {
    unit 0 {
      scheduler-map sched2;
      fragmentation-map map2;
    }
  }
  ds-0/2/0:1 {
    scheduler-map sched2;
  }
  ge-0/0/0 {
    unit 0 {
      classifiers {
        inet-precedence ge-interface-classifier;
      }
    }
  }
}
scheduler-maps {
  sched2 {
    forwarding-class be scheduler economy;
    forwarding-class ef scheduler business;
    forwarding-class af scheduler stream;
    forwarding-class nc scheduler voice;
  }
}

```

```
fragmentation-maps {
  map2 {
    forwarding-class {
      be {
        fragment-threshold 160;
      }
      ef {
        fragment-threshold 160;
      }
      af {
        fragment-threshold 160;
      }
      nc {
        no-fragmentation;
      }
    }
  }
}
schedulers {
  economy {
    transmit-rate percent 26;
    buffer-size percent 26;
  }
  business {
    transmit-rate percent 26;
    buffer-size percent 26;
  }
  stream {
    transmit-rate percent 35;
    buffer-size percent 35;
  }
  voice {
    transmit-rate percent 13;
    buffer-size percent 13;
  }
}
```

