# *MPLS Traffic Engineering MIB*

*Cheenu Srinivasan*
<cheenu@lucent.com>

*Arun Viswanathan*
<arunv@lucent.com>

*Lucent Technologies*
*Holmdel, NJ*

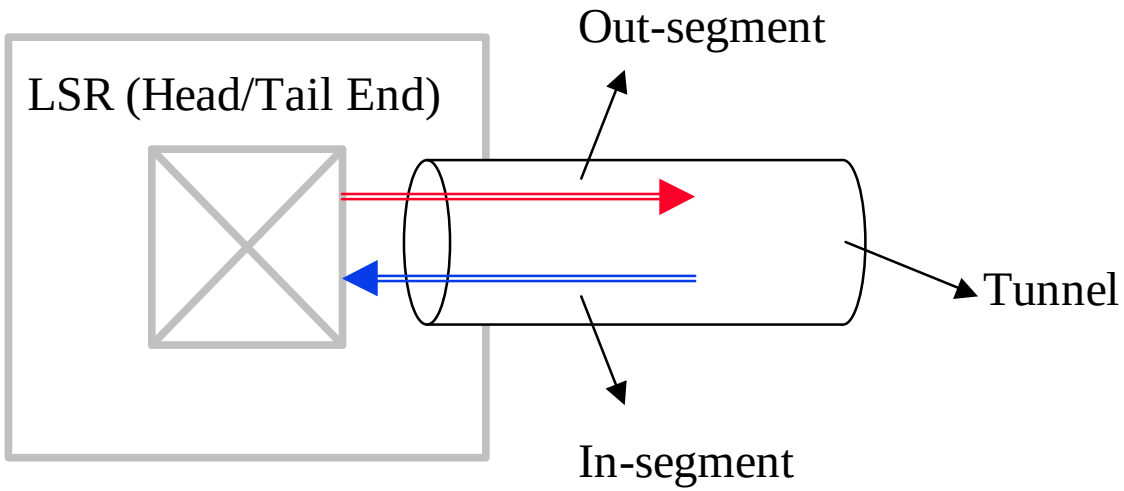
# Overview

- *Feature checklist*
- *Modeling an LSR*
- *Components of the MIB*

# Some Features To Keep in Mind

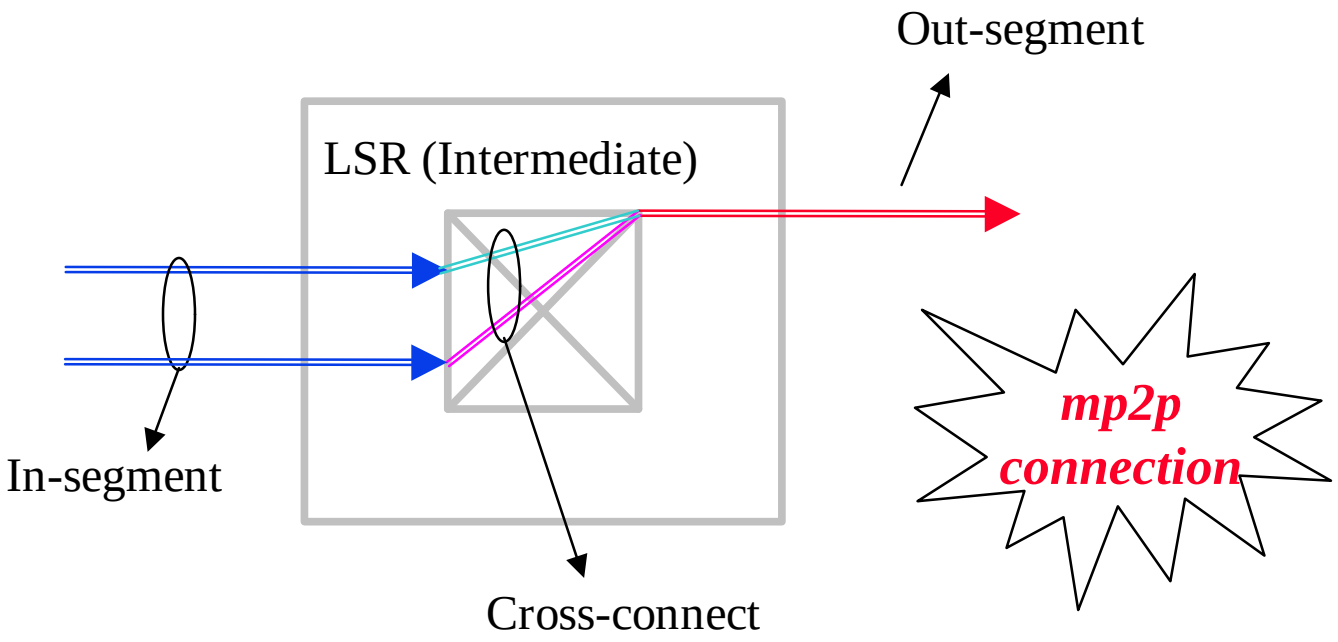
- *Support for p2p bi-directional & mp2p uni-directional tunnels*
- *Support for both manual and signalled tunnel setup*
- *Should be possible (but not necessary) for tunnels to be interfaces*
- *Support for p2p, p2mp, mp2p connections at cross-connect*
- *In a multicast (p2mp) connection each outgoing segment can have its own label stack including top label*
- *For mp2p connection packets from the different merged connections can have distinct label stack beneath top label*
- *Support for "pop-and-go" without pushing any labels*
- *Ability to measure tunnel and cross-connect performance*

# Modeling an LSR




***Bi-directional Tunnel***

# Modeling an LSR




## Cross-connected Segments

# Configuration Steps for Traffic Engineering

- *Setting up the tunnels with appropriate parameters*
- *Setting up tunnel segments with appropriate parameters*
- *Setting up the cross-connect entries to associate segments*
- *Specifying label stack actions*

# mplsTunnelTable

- *Supports p2p uni- and bi-directional tunnels between LSR and remote end-point*
- *Indexed by* `mplsTunnelIndex` *- uniquely identifies the tunnel*
- `mplsTunnelIsIf` *- true(1), false(2); whether tunnel is an interface*
- `mplsTunnelIfIndex` *- LSR assigned* `ifIndex` *for tunnel*
- `mplsTunnelDirection` *- in(1), out(2), in-out(3)*
- `mplsTunnelXCIndex` *- pointer into* `mplsXCTable` *to identify tunnel segments*
- `mplsTunnelRemoteIpAddr` *- IP addr of remote end-point*
  - *Will be augmented to support IPv6 and other address forms*
- `mplsTunnelSignallingProto` *- none(1), ldp(2), rsvp(3)*
- `mplsTunnel{Admin,Oper}Status` *- desired/operational status of tunnel*

# *mplsTunnelHopTable*

- *Indexed by*
  - `mplsTunnelIndex` - *identifies tunnel this source route corresponds to*
  - `mplsHopIndex` - *identifies a particular hop*
- `mplsHopIpAddr` - *IPv4 address of the hop*
  - *Will be augmented to support for IPv6 and other address forms*
- `mplsHopStrictOrLoose` - *strict(1), loose(2)*

# *mplsInSegmentTable*

- *Indexed by*
  - `mplsInSegmentIfIndex` - *incoming interface*
  - `mplsInSegmentLabel` - *incoming label; 32-bit object interpreted in an interface-type dependent fashion*
- `mplsInSegmentAddrFamily` - *IANA address family of the incoming packets in order to perform address family specific actions such as when TTL expires*
- `mplsInSegmentXCIndex` - *cross-connect entry that this segment is part of (0 if none)*
- *Traffic parameters:* `mplsInSegment{MaxRate,MeanRate,MaxBurstSize}`

# *mplsOutSegmentTable*

- *Indexed by* `mplsOutSegmentIndex`
- `mplsOutSegmentIfIndex` - `ifIndex` *of the outgoing interface*
- `mplsOutSegmentPushTopLabel` - *true(1), false(2); whether a top label should be pushed onto the outgoing packet or perform "pop-and-go"*
- `mplsOutSegmentTopLabel` - *top label to pushed onto outgoing packet; when a stack needs to be pushed beneath this top label it is specified in* `mplsLabelStackTable`
- `mplsOutSegmentNextHopIPAddr` - *needed for shared media interfaces*
- `mplsOutSegmentXCIndex` - *cross-connect entry that this segment is part of (0 if none)*
- *Traffic parameters:* `mplsOutSegment{MaxRate,MeanRate,MaxBurstSize}`

# *mplsXCTable*

- *Supports p2p, p2mp, and mp2p connections*
- *Indicates association between set of in- and out-segments*
  - *set of segments forming a tunnel*
  - *cross-connect switching actions*
- *Indexed by*
  - `mplsXCIndex` - *uniquely identified this set of cross-connect entries*
  - `mplsInSegmentIfIndex`, `mplsInSegmentLabel` - *identifies the in-segment that this entry corresponds to*
  - `mplsOutSegmentIndex` - *identifies the out-segment that this entry corresponds to*
- `mplsXCLabelStackIndex` - *stack of labels, if any, to be pushed beneath top label specified in* `mplsOutSegmentTable`

# *Other Objects*

- `mplsLabelStackTable`
  - *indexed by* `mplsLabelStackIndex` *and* `mplsLabelStackIndex`
  - *stack of labels under top label; pointed to from* `mplsXCTable`
- `mpls{In,Out}SegmentPerfTable`
  - *performance of in- and out-segments*
  - *incorporates both tunnel and cross-connect performance*
  - *objects to measure octets, packets, errors, discards*
- *Up/Down notifications for segments, tunnels, cross-connects*

# Recap

- `mplsTunnelTable` - *configuring tunnels*
- `mplsTunnelHopTable` - *loose/strict source route hops*
- `mplsInSegmentTable`, `mplsInSegmentPerfTable` - *configuring in-segments and measuring their performance*
- `mplsOutSegmentTable`, `mplsOutSegmentPerfTable` - *configuring out-segments and measuring their performance*
- `mplsXCTable` - *configuring cross-connects, specifying segment associations*
- `mplsLabelStackTable` - *label stacks*

# Recap

- *Described MIB for MPLS traffic engineering*
  - *Addresses essential features described earlier for configuration and performance*
  - *Extensions planned in post-IETF draft iteration:*
    - *Support for IPv6 addresses, IPv4/IPv6 prefixes, AS number*
    - *Tspec objects in* `mplsTunnelTable` *for signalled setup*
- *Request that MIB be adopted as a WG draft*

# *Questions?*