Virtual Circuit vs. Datagram

- Objective of both: move packets through routers from source to destination
- Datagram Model:
  - *Routing*: determine next hop to each destination a priori
  - *Forwarding*: destination address in packet header, used at each hop to look up for next hop
    - routes may change during "session"
  - analogy: driving, asking directions at every gas station, or based on the road signs at every turn
- Virtual Circuit Model:
  - *Routing*: determine a path from source to each destination
  - *"Call" Set-up*: fixed path ("virtual circuit") set up at "*call*" *setup time*, remains fixed thru "call"
  - *Data Forwarding*: each packet carries "tag" or "label" (virtual circuit id, VCI), which determines next hop
  - *routers maintain "per-call" state*

Virtual Circuits

"source-to-dest path behaves much like telephone circuit" (but actually over packet network)

- performance-wise
- network actions along source-to-dest path

- call setup/teardown for each call *before* data can flow
  - need special control protocol: "signaling"
  - *every* router on source-dest path maintains "state" (VCI translation table) for each passing call
  - VCI translation table at routers along the path of a call "weaving together" a "logical connection" for the call
- link, router resources (bandwidth, buffers) may be *reserved* and *allocated* to each VC
  - to get "circuit-like" performance
- Compare w/ transport-layer "connection": only involves two end systems, no fixed path, can't reserve bandwidth!

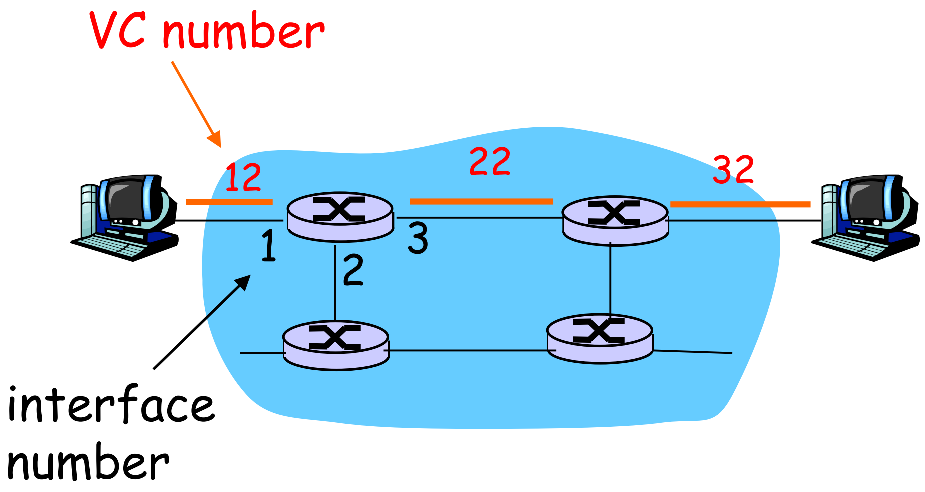
VC Implementation

a VC consists of:

1. path from source to destination
2. VC numbers, one number for each link along path
3. entries in forwarding tables in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - New VC number comes from forwarding table

VC Translation/Forwarding Table



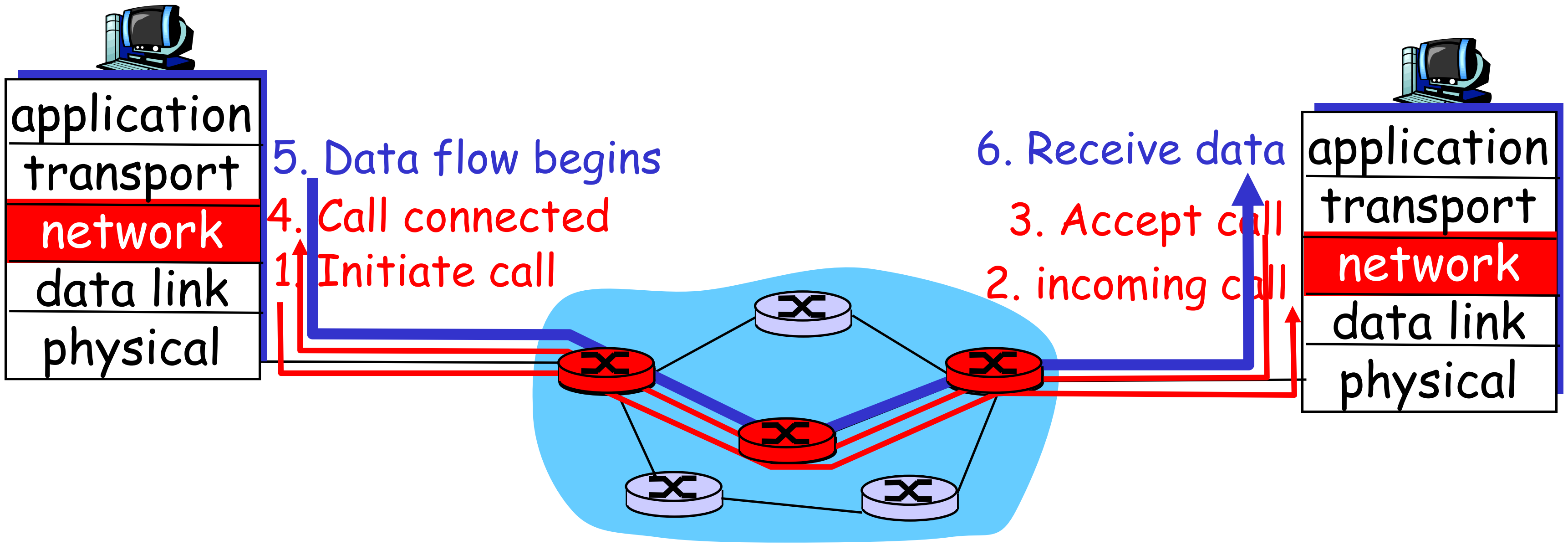
Forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

Routers maintain connection state information!

Virtual Circuit: Signaling Protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- used in part of today's Internet: Multi-Protocol Label Switching (MPLS) operated at "layer 2+1/2" (between data link layer and network layer) for "traffic engineering" purpose



Virtual Circuit Setup/Teardown

Call Set-Up:

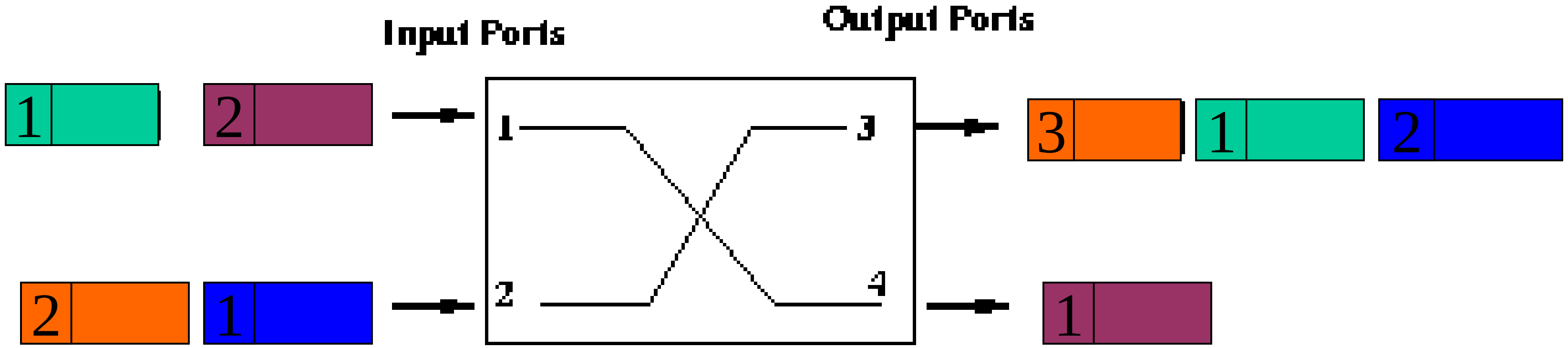
- Source: select a path from source to destination
  - Use routing table (which provides a "map of network")
- Source: send VC setup request control ("signaling") packet
  - Specify path for the call, and also the (initial) output VCI
  - perhaps also resources to be reserved, if supported
- Each router along the path:
  - Determine output port and choose a (local) output VCI for the call
    - need to ensure that NO two distinct VCs leaving the same output port have the same VCI!
  - Update VCI translation table ("forwarding table")
    - add an entry, establishing an mapping between incoming VCI & port no. and outgoing VCI & port no. for the call

Call Tear-Down: similar, but remove entry instead

	Input Port	Input VCI	Output Port	Output VCI
green call	1	1	3	1
purple call	1	2	4	1
blue call	2	1	3	2
orange call	2	2	3	3

four "calls" going thru the router, each entry corresponding one call

VCI translation table (aka "forwarding table"), built at call set-up phase



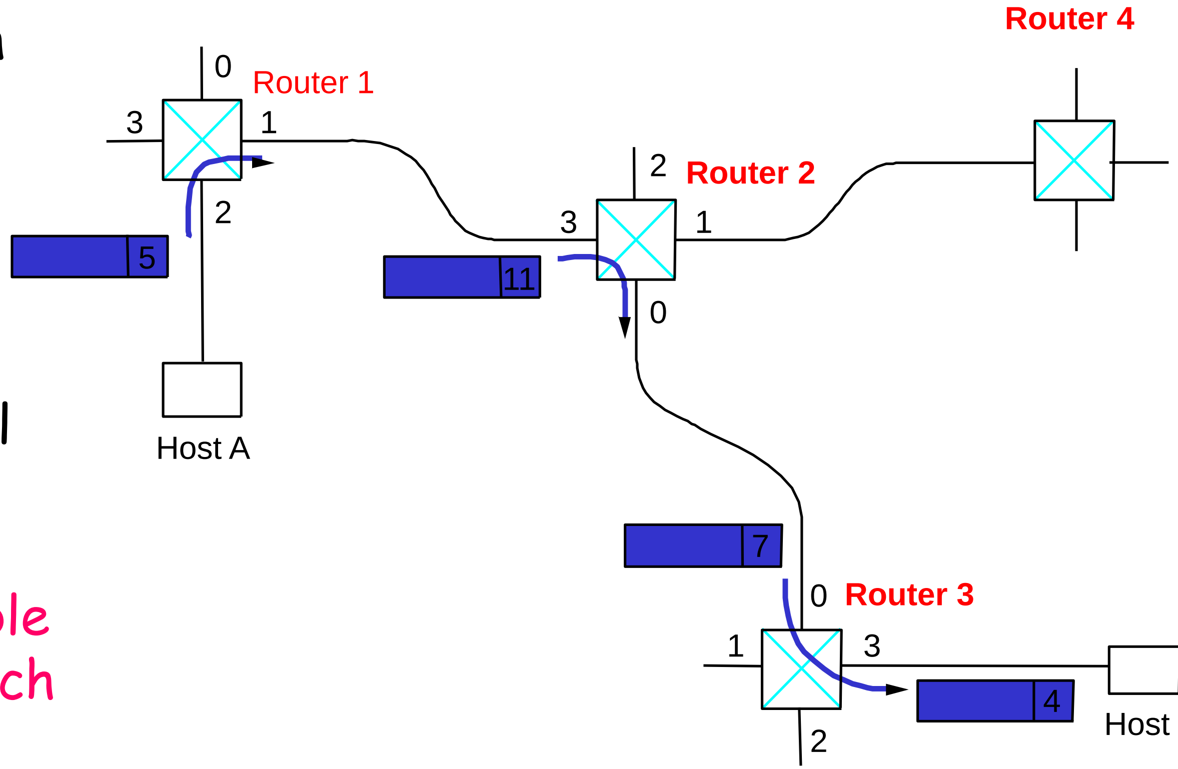
During data packet forwarding phase, input VCI is used to look up the table, and is "swapped" w/ output VCI (VCI translation, or "label swapping")

Virtual Circuit: Example

"call" from host A to host B along path:
host A→ router 1→ router 2 → router 3 → host B

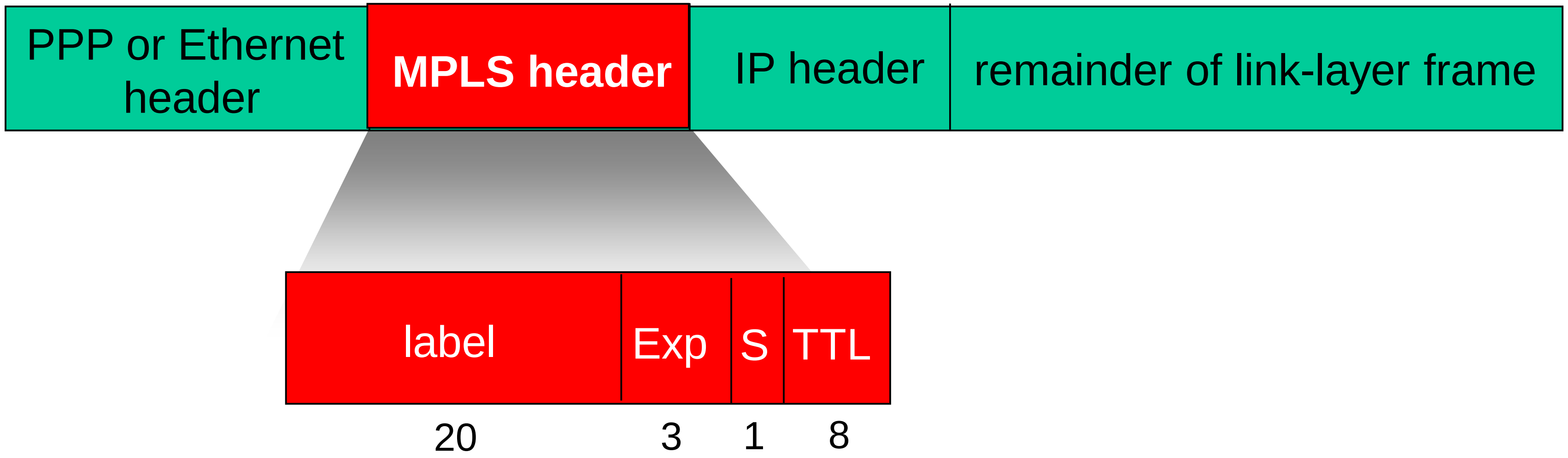
- each router along path maintains an entry for the call in its VCI translation table
- the entries piece together a "logical connection" for the call

- Exercise: write down the VCI translation table entry for the call at each router



Multiprotocol Label Switching (MPLS)

- initial goal: speed up IP forwarding by using fixed length label (instead of IP address) to do forwarding
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!



PPP or Ethernet header

MPLS header

IP header

remainder of link-layer frame

label

Exp

S

TTL

20

3

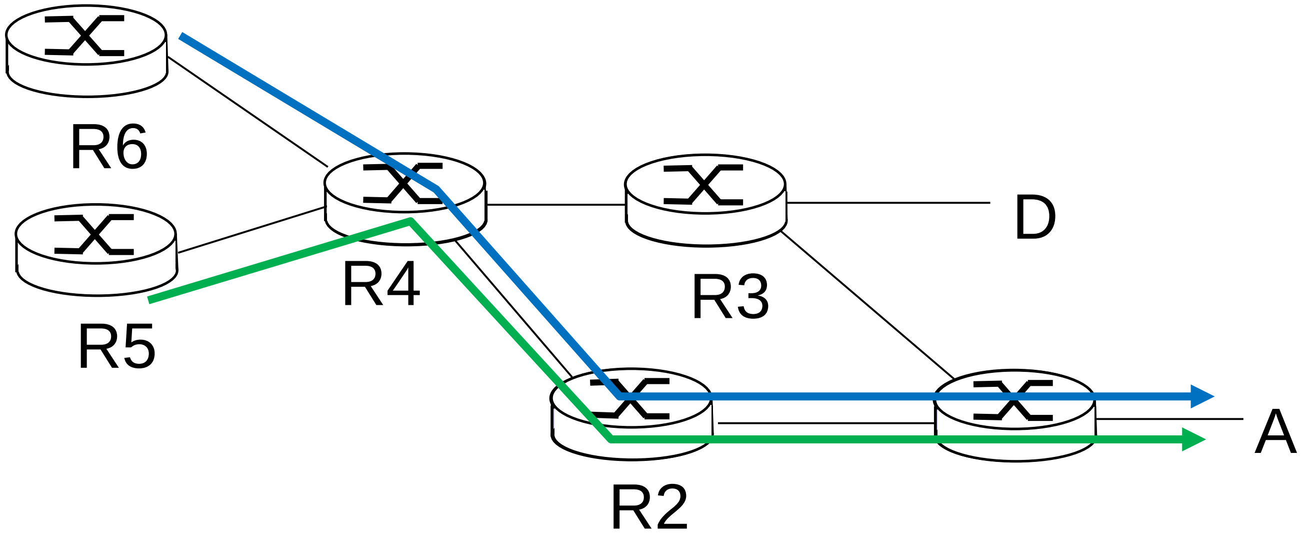
1

8

MPLS Capable Routers

- a.k.a. label-switched router
- forward packets to outgoing interface based only on label value (*don't inspect IP address*)
  - MPLS forwarding table distinct from IP forwarding tables
- *flexibility:* MPLS forwarding decisions can *differ* from those of IP
  - use destination *and* source addresses to route flows to same destination differently (traffic engineering)
  - re-route flows quickly if link fails: pre-computed backup paths (useful for VoIP)

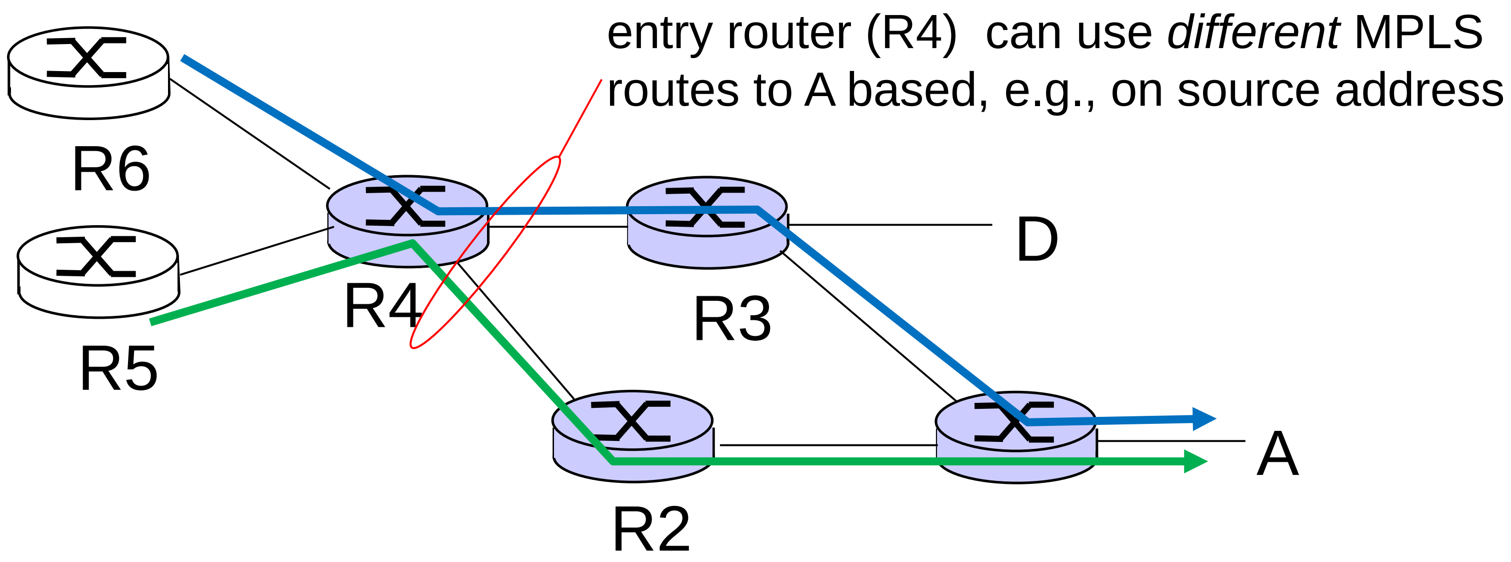
MPLS versus IP paths



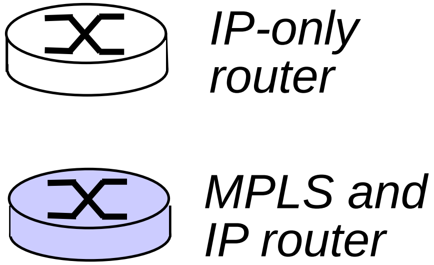
- *IP routing:* *path to destination determined by destination address alone*

IP router

MPLS versus IP paths

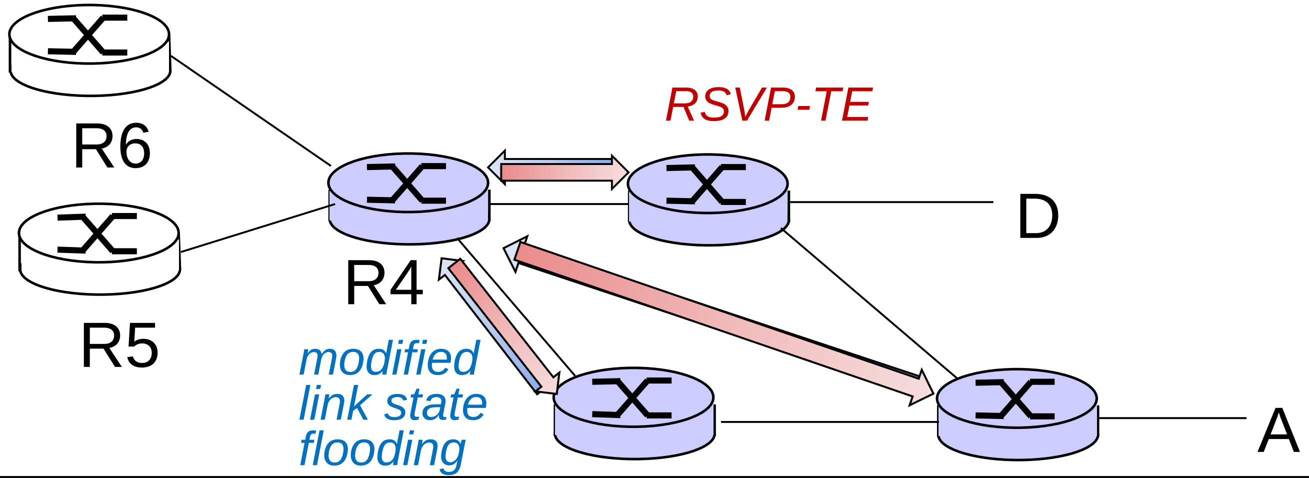

entry router (R4) can use *different* MPLS routes to A based, e.g., on source address
R6
R5
R4
R3
D
R2
A

- *IP routing:* path to destination determined by destination address alone
- *MPLS routing:* path to destination can be based on source *and* destination address
  - *fast reroute:* precompute backup routes in case of link failure



MPLS Signaling

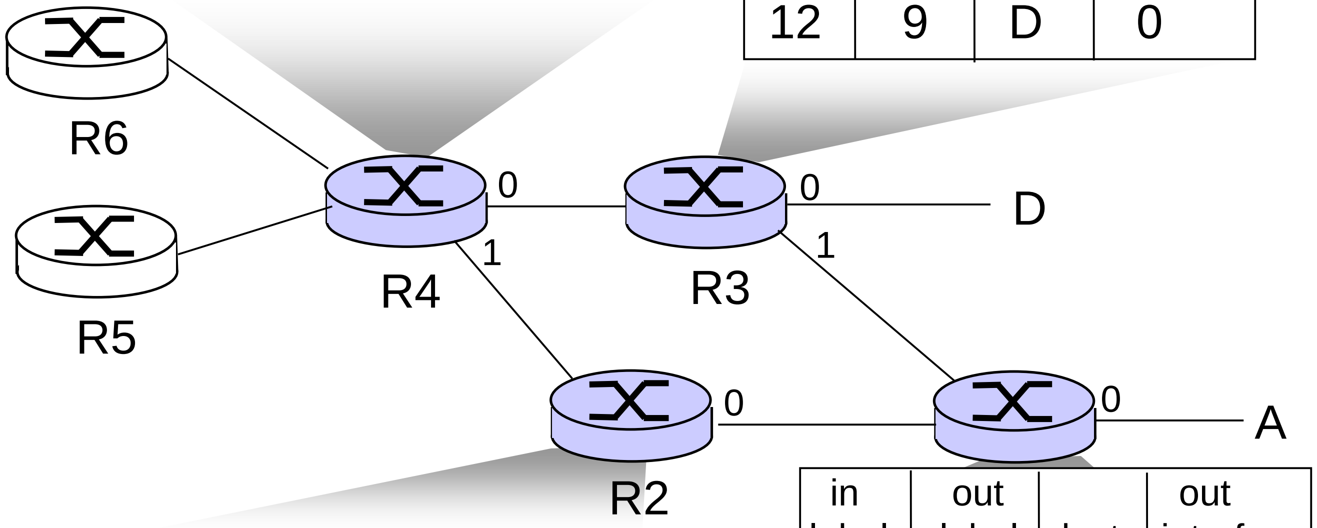
- modify OSPF, IS-IS link-state flooding protocols to carry info used by MPLS routing,
  - e.g., link bandwidth, amount of "reserved" link bandwidth
- *entry MPLS router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers*



MPLS Forwarding Tables

in label	out label	dest	out interface
	10	A	0
	12	D	0
	8	A	1

in label	out label	dest	out interface
10	6	A	1
12	9	D	0



in label	out label	dest	out interface
8	6	A	0

in label	out label	dest	out interface
6	-	A	0