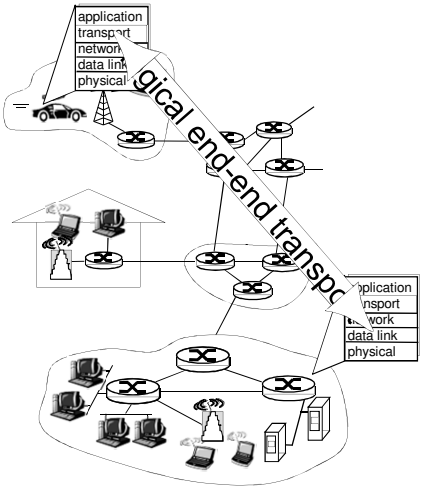
Transport Protocols and MPLS #9

Outline

- Goals:
  - Understand principles behind transport layer services
  - Multiplexing/demultiplexing (Ports/Sockets)
- Examples of Transport Protocols
  - UDP
  - TCP
    - reliable data transfer
    - flow control
    - congestion control
  - Note there are other transport layer protocols
- MPLS

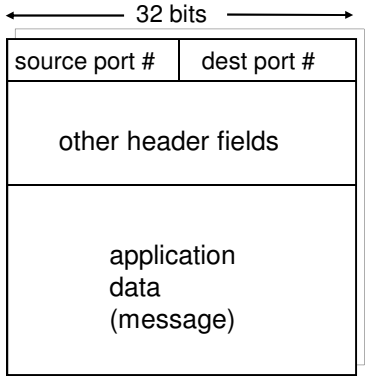
Transport services and protocols

- Provide *logical communication* between app processes running on different hosts
- Transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- More than one transport protocol available to apps
  - Internet: TCP, RTP, UDP and others



How multiplexing works

- IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries 1 transport-layer segment
  - each segment has source, destination port number
- host uses IP addresses & port numbers to direct segment to appropriate socket



TCP/UDP segment format

Ports and Sockets

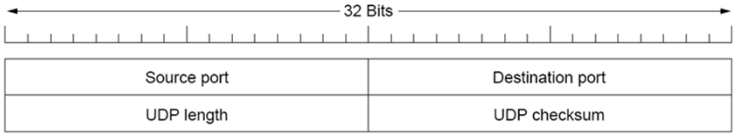
- Port address + IP address = socket address
- Ports are 16 bits
- Both sender and receive have socket addresses
- A connection is identified by a pair of sockets
- The port address is internal to the host (indicates application)
- A socket address is unique in the internet
- Once an application creates a socket and TCP connection then a ***write*** is used to send to the network and a ***read*** used to receive from the network.

Ports and Sockets

- Well Known Ports are those from 0 through 1023.
- The Registered Ports are those from 1024 through 49151
- The Dynamic and/or Private Ports are those from 49152 through 65535
- There are some common port numbers
  - Example:
    - File data transfer (21)
    - TELNET (23)
    - Simple Mail Transfer Protocol (SMTP)(25)
    - Remote Procedure Call [RPC] (111)
    - Web servers listens on port 80
- http://www.iana.org/assignments/port-numbers

Transport Layer: UDP

- UDP
  - Connectionless
  - No congestion control
  - No acknowledgments
  - Packets may be
    - lost
    - delivered out of order to app
  - No handshaking between UDP sender, receiver
  - Each UDP segment handled independently of others
  - UDP checksum covers header and data → optional



From: "Computer Networks, 3rd Edition, A.S. Tanenbaum. Prentice Hall, 1996

UDP Use Cases

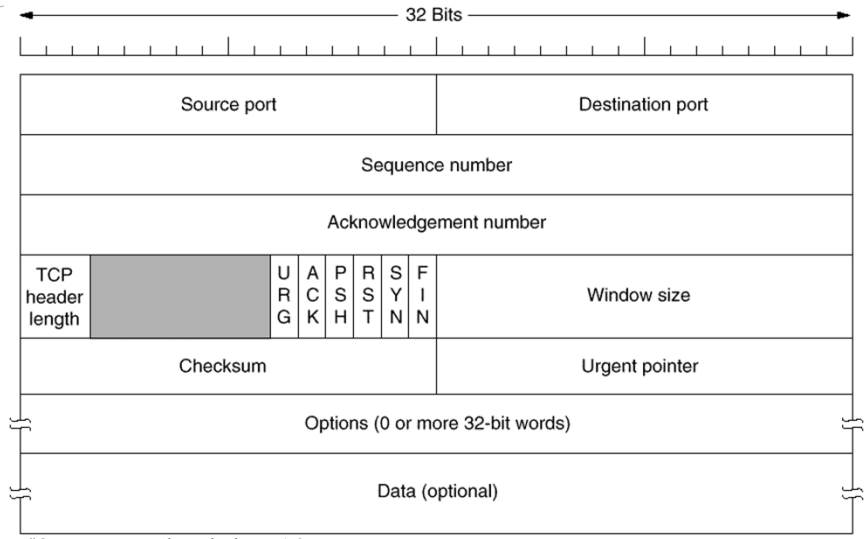
- Streaming multimedia apps
  - loss tolerant
  - rate sensitive
- DNS
- Simple Network Management Protocol(SNMP)
- Reliable transfer over UDP: add reliability at application layer → application-specific error recovery!

Modified from: Computer Networking: A Top Down Approach 4th edition. Jim Kurose, Keith Ross Addison-Wesley, July 2007.

Transport Layer: TCP

- TCP provides for assured delivery of PDU's
- TCP Services
  - Connection oriented (end-to-end)
    - Need call processing
    - Information on the status of each connection is available
  - Reliable data transfer
    - Uses acknowledgments
    - Uses sequence numbers

TCP Header



From: "Computer Networks, 3rd Edition, A.S. Tanenbaum. Prentice Hall, 1996

TCP Header

- Source/Destination identify local end points
- Window size (in Bytes) used to dynamically control source rate into the network
- Checksum, checks the header and data

TCP

- Stream-oriented
  - TCP collect user bytes and forms segments to be passed on to the IP layer
  - Sequence number based on byte counts
- Push
  - Upper layer protocol send Push message to TCP to force it to send all the bytes collected in a segment
- Resequencing
  - IP may deliver information out of order, TCP must put it back together

TCP

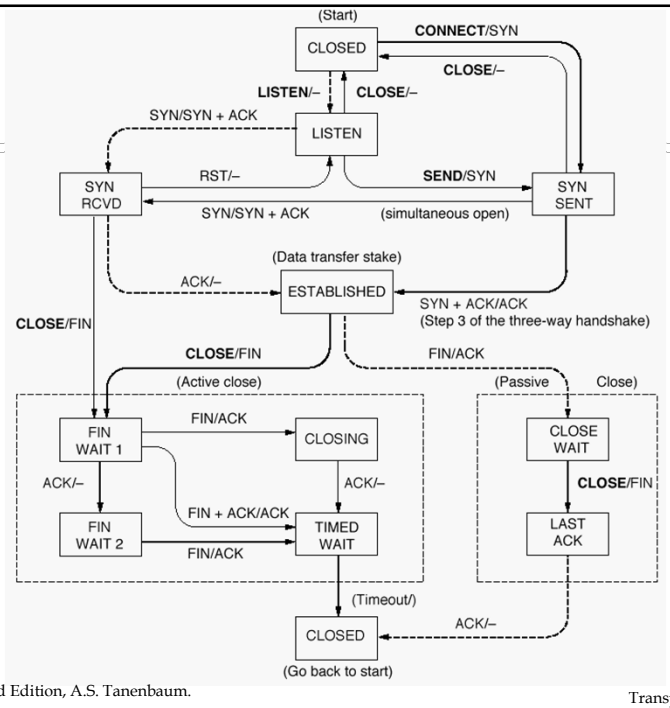
- Inclusive Acknowledgment
  - Acknowledgment number, acknowledges all received bytes prior to the one specified
- Flow control
  - Window size is in bytes
  - Transmit N-bytes and the must wait for acknowledgment
  - Window size is dynamic, i.e., it changes based on "knowledge" of network congestion

TCP

- Multiplexing
  - Allows multiple sessions within one host to be transmitted over an IP path (ports/sockets)
- Full duplex
- Security and precedence
- Graceful close
  - All traffic in flow is acknowledged before the session is ended.

TCP Session Processing

TCP Connection Management: Finite State Machine



From: "Computer Networks, 3rd Edition, A.S. Tanenbaum. Prentice Hall, 1996

TCP Connection Setup: Three-way Handshake

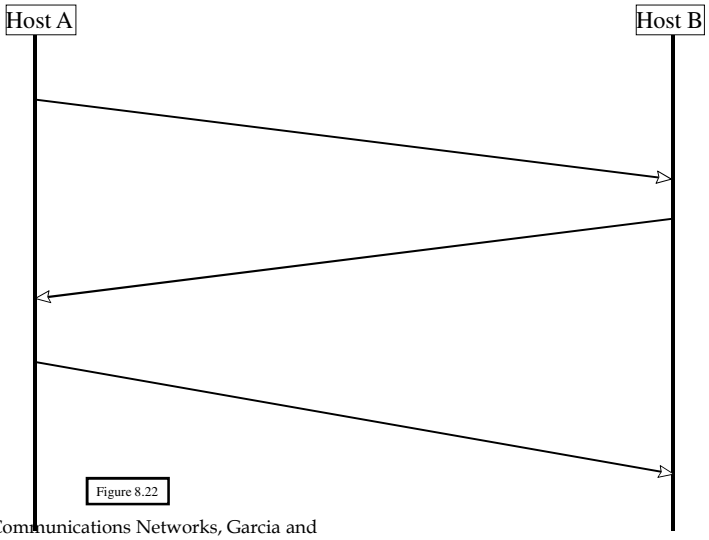


Figure 8.22

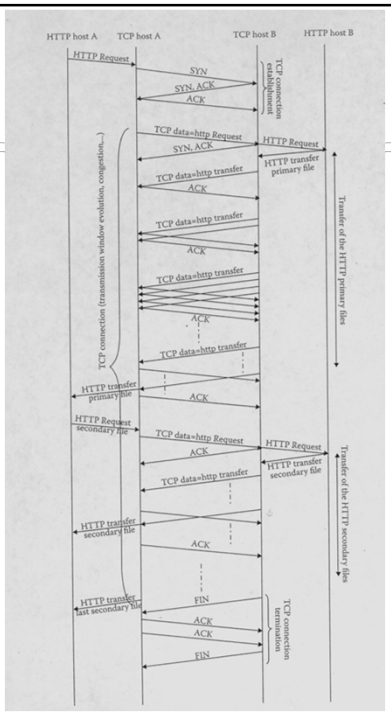
From: Communications Networks, Garcia and Widaja, McGraw Hill, 2000

tcpdump
http connection set up

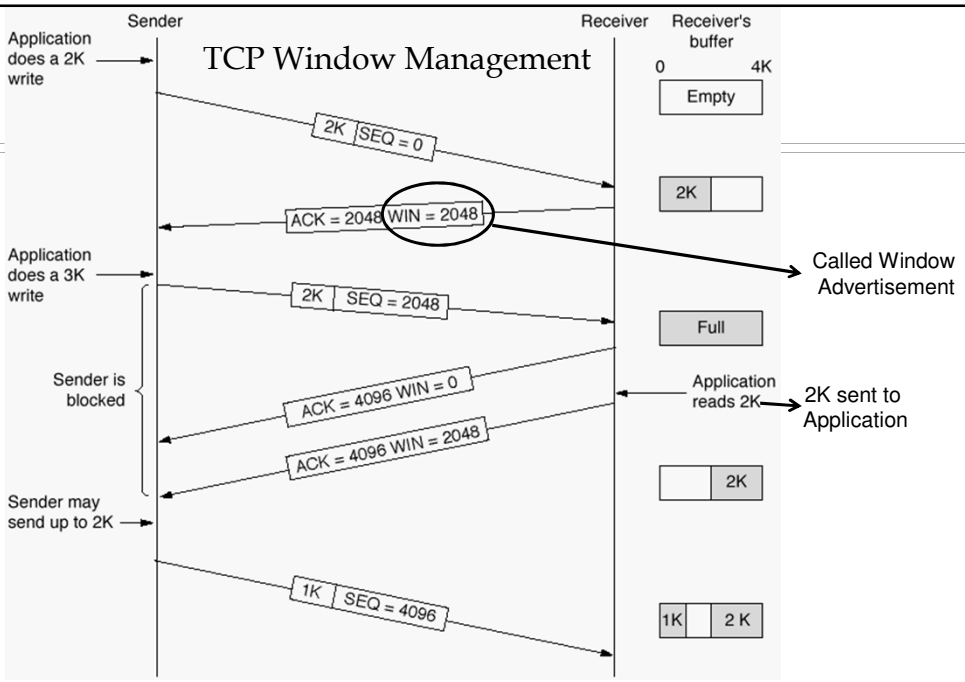
- Output columns are → Time SourceIP.SourcePort > DestIP.DestPort Flags ...
- 11:13:38.524046 x.x.x.x.3600 > 64.233.167.104.80: S 2021815674:2021815674(0) win 64240 <mss 1460,nop,nop,sackOK> (DF)
  - First packet is from client host x.x.x.x.
  - Client host is using 3600 as a source port.
  - Destination host is 64.233.167.104 on port 80 (that's Google's webserver).
  - The packet with flag S is a TCP SYN packet, means in words "i'd like to open a TCP connection with you"
  - Client host will have a temporarily opened port (3600) in order to receive data back from the server.
- 11:13:38.558668 64.233.167.104.80 > x.x.x.x.3600: S 3132749891:3132749891(0) ack 2021815675 win 8190 <mss 1460>
  - Second packet is sent from Google webserver. This packet comes from 64.233.167.104 source port 80, and contains SYN/ACK
  - TCP flags sent to client port 3600, means "ok you may open a connection with me".
- 11:13:38.559105 x.x.x.x.3600 > 64.233.167.104.80: . ack 1 win 64240 (DF)
  - Third packet is the client host sending a last ACK packet, which means "ok we are now connected". Source and dest ports must stay the same here.

Example

A TCP Session:
Start-to-Finish



From: Mohamad Assaad, Zeghlache Djamal TCP Performance Over Umts-Hsdpa Systems, CRC Press



From: "Computer Networks, 3rd Edition, A.S. Tanenbaum. Prentice Hall, 1996

Silly Window Syndrome

- Situation:
  - Transmitter sends large amount of data
  - Receiver buffer depleted slowly, so buffer fills
  - Every time a few bytes read from buffer, a new advertisement to transmitter is generated
  - Sender immediately sends data & fills buffer
  - Many small, inefficient segments are transmitted
- Solution:
  - Receiver does not advertize window until window is at least ½ of receiver buffer or maximum segment size
  - Transmitter refrains from sending small segments

From: Communication Networks: Fundamentals Concepts and Key Architectures Authors: A. Leon-Garcia and I. Widjaja

Delay-BW Product & Advertised Window Size

- Suppose RTT=100 ms, R=2.4 Gbps
  - # bits in pipe = 30 Mbytes
- If single TCP process occupies pipe, then required advertised window size is
  - RTT x Bit rate = 30 Mbytes
  - Normal maximum window size is 65535 bytes
  - With normal max window efficiency ~ 0.2%
- Solution: Window Scale Option
  - Window size up to 65535 x 2^{14} = 1 Gbyte allowed
  - Requested in SYN segment
  - Uses options Fields

From: Communication Networks:
Fundamentals Concepts and Key Architectures
Authors: A. Leon-Garcia and I. Widjaja

TCP: Retransmission Procedures

- TCP uses a positive acknowledgment
- Selecting timeout timer value
  - Delay unknown a-priori
  - Segments may be lost making measurements of the round-trip time (RTT) difficult, i.e., measurement of RTT can have a large variance

TCP: Timeout Interval

- Estimate the RTT for the $(i+1)^{th}$ segment, RTT(i+1)
- RTT(i+1) = aRTT(i) + (1-a)RTT(i-1)
  where 0<a<1
  [a typically = 7/8]
- Timeout interval = *b*RTT(i)

TCP: Timeout Interval

- The 'a' parameter governs the time response to changes in RTT
- Standards recommend *b*=2
- *b* can be adaptive and proportional to Var[acktime]

TCP: Adaptive Congestion Control

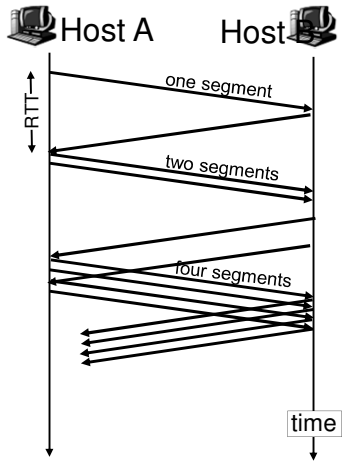
- If time out TCP **assumes** congestion caused loss
- If the network is congested then want to slow the source down to reduce congestion
- When the network congestion disappears then want to allow the source to send faster

TCP: Adaptive Congestion Control

- Turn efficiency calculation of data link control algorithms around
- Use window size to control the flow of traffic into the network

TCP: Adaptive Congestion Control

- Increase algorithm
  - If acknowledge received then increase the window size by one segment, i.e.,
  - new_window = old_window + 1 segment
  - This is called the slow start phase



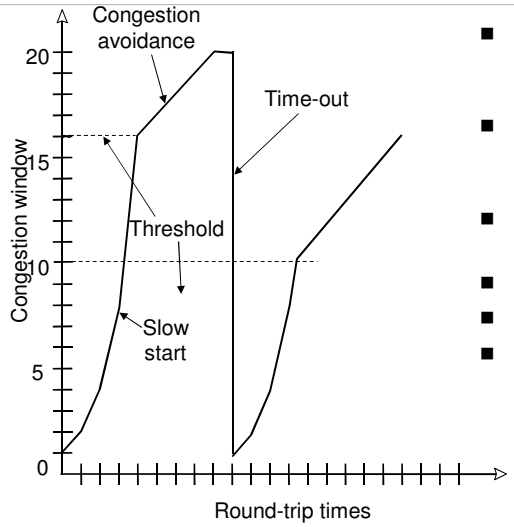
TCP: Adaptive Congestion Control

- If every packet is acknowledged in slow start then the window (and rate) doubles every RTT, Exponential increase.
- After the window reaches a threshold, it enters the congestion avoidance phase.
- In the congestion avoidance phase, upon receipt of an Ack it is increased by 1 segment every RTT, Linear increase

TCP: Adaptive Congestion Control

- Decrease Algorithm
  - If loss then set
    - –new_threshold = (1/2)current window
    - –Redo Slow Start from 1 Segment

TCP Congestion Control: Congestion



- Congestion is detected upon timeout or receipt of duplicate ACKs
- Assume current cwnd corresponds to available bandwidth
- Adjust congestion threshold = ½ x current cwnd
- Reset cwnd to 1
- Go back to slow-start
- Over several cycles expect to converge to congestion threshold equal to about ½ the available bandwidth

Modified from: Communication Networks: Fundamentals Concepts and Key Architectures Authors: A. Leon-Garcia and I. Widjaja

Variation of TCP Algorithms: Intertwined algorithms used commonly in TCP implementations

- TCP can use Go-Back-N or Selective Acknowledgements (SACK); SACK is most common
- **Slow Start** - Every *ack* increases the sender's window (*cwnd)* size by 1
- **Congestion Avoidance** - Reducing sender's window size by half at experience of loss, and increase the sender's window at the rate of about *one packet per RTT (NOTE: not per ack)*
- **Fast Retransmit** - Don't wait for retransmit timer to go off, retransmit packet if *3 duplicate acks* received
- **Fast Recovery** - Since duplicate *ack* came through, one packet *has left the wire*. Perform *congestion avoidance*, don't jump down to *slow start*

Modified from : Paul D. Amer, University of Delaware
*www.cis.udel.edu/~amer/856/**tcpvariations**-Amer.ppt*

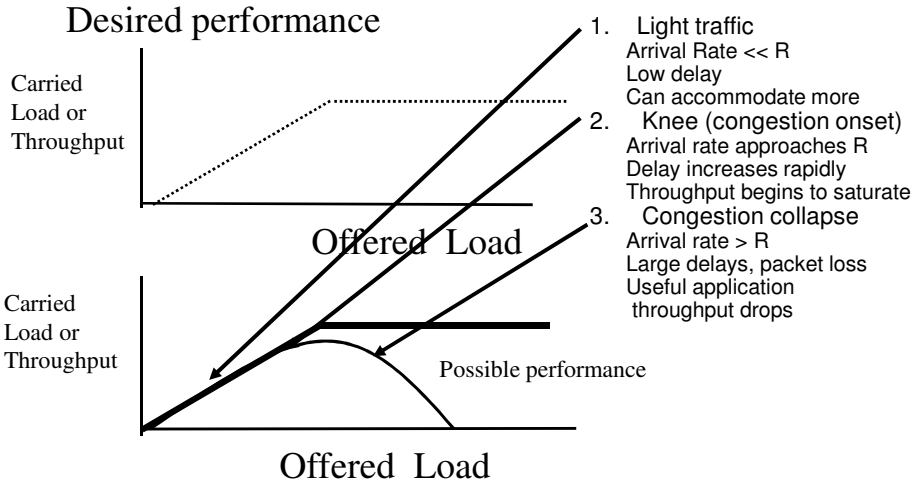
Flavors of TCP

- TCP is end-to-end so many variations can co-exist in the Internet.
  - TCP-Tahoe
  - TCP-Reno
  - TCP-Vegas
  - TCP-NewReno
  - Fast TCP (FastTCP)
  - BIC TCP (Binary Increase Congestion control)
  - CUBIC TCP
  - HighSpeed TCP (HSTCP)
  - Compound TCP (CTCP)
    - Microsoft algorithm that was introduced as part of the Windows Vista and Window Server 2008 TCP stack.

Congestion Control

- Global Issue
- Demand for network resources must be controlled, e.g. the number of packets or calls in the systems must be controlled

Congestion Control-Objective



Congestion Control

- Preventative
  - Call Admission Control (CAC)
  - VC switching
- Reactive
  - Packet Dropping
  - TCP is reactive→ End-to-End

A Congestion Control Method for the Internet

- Random Early Detection (RED)
  - RED is an example of Active Queue Management (AQM)
  - Monitor average **ROUTER** queue length
  - If average **ROUTER** queue length > threshold then Drop arriving packet with some probability p, (p=drop probability)
- This *implicitly* notifies the TCP source that there is congestion and the source then backs off
- In the Internet *"Random Early Detection"* (Red) gateways use this basic concept with some added complexity

MultiProtocol Label Switching (MPLS): Why?

- Provide a form a virtual circuit switching in the Internet for aggregates of flows not for individual hosts
- Label switching enables routing flexibility
- Virtual circuit switching enables QoS on aggregates of flows

MPLS: Why?

- Improve IP forwarding performance
- Decouple routing and forwarding components of IP
  - Routing - OSPF, IS-IS, BGP-4 to build and maintain forwarding tables
  - Forwarding - directs packet from input interface to output interface, based on forwarding table look-up
  - MPLS can use different routing protocols for flow aggerates.

MPLS: Why?

- Traffic Engineering - using Explicit routes and Constraint-based routing for better load balancing.
  - Traffic Engineering is moving the traffic to where the bandwidth is; establish separate paths to meet different performance requirements of aggregated traffic flows
  - Network Engineering is configuring the network to support the traffic.
- Improve scaling of IP overlay networks
  - Edge routers can peer with nearby MPLS nodes, rather than peering with N-1 other routers

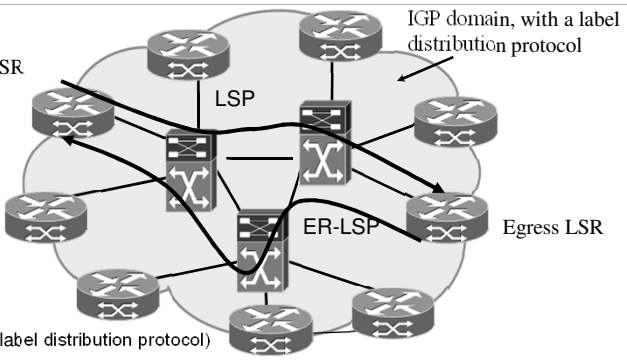
MPLS: Why

- MPLS provides a tunneling mechanism to interconnect VPN sites
- MPLS can be generalized to provide
  - Control plane for optical cross-connects
  - Automatic protection switching, without SONET overheads
  - Generalized MPLS (GMPLS)
    - Time Slot→ Label
    - Wavelength → Label
    - MPLS (IP) → Label
    - All can use the same infrastructure

MultiProtocol Label Switching (MPLS) concepts-How?

- Just like Virtual Circuit Switching (but with different terms)
- Forwarding Equivalence Class **(FEC)** - group (***Aggregate***) of IP packets that are forwarded in the same manner
- Label - assigned per FEC
- Label Switch Router **(LSR)** –
  Here a routers acts *Like a VC switch*
- Edge LSRs assign/remove labels, can perform packet classification
- Core LSRs switch packets based on label value
- Existing IP routing protocols used to exchange routing info
- All LSRs use some kind of label distribution protocol **(LDP)**
  a ***signaling protocol***
- Label Switched Path **(LSP)** - sequence of LSRs through which labeled packets go through to reach the egress LSR

MPLS Concepts-How?



- LSP's are unidirectional
- Route selection can use:
  - Hop-by-hop routing
    (using IGP and a label distribution protocol)
  - Explicit routing - ingress LSR specifies all LSR nodes that are in the path (statically (source routing), or using link-state topology information)
    - May be signaled using RSVP-TE or CR-LDP
    - May be different from IGP-shortest path
    - Explicit routing useful for traffic engineering

MPLS Concepts- How?: Packet Header

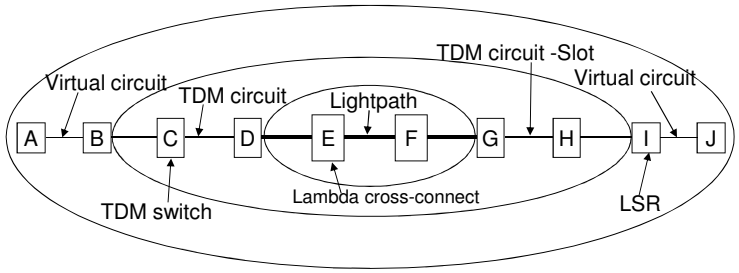
Label
- Value to determine next hop of the packet

Experimental (EXP)
- Used as CoS field - Limited QoS parameters, derived from IP header, diffserv, etc.

Bottom of Stack (S)
- Set to 1 if bottom of label stack, otherwise 0

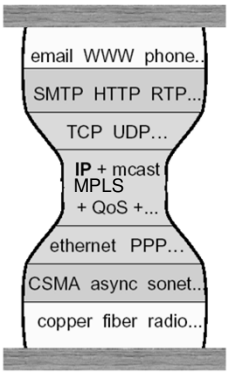
Time to Live (TTL)
- Eliminates loops and prevents packets from remaining in the network indefinitely

GMPLS & Hierarchical LSPs

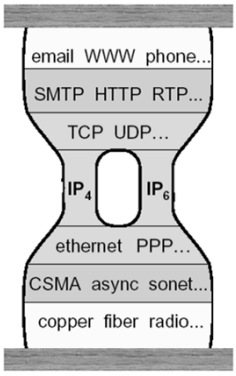


- GMPLS allows node with multiple switching technologies to be controlled by one control component
- Notion of "label" generalized:
  - TDM slot, WDM wavelength, optical fiber port
- LSP Hierarchy extended to generalized labels"
  - MPLS LSP over SONET circuit over wavelength path over fiber

IP Hourglass Architecture: Revisited

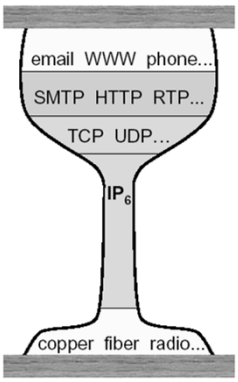


New protocols require more functionality from underlying networks



1) Doubles number of service interfaces
2) Requires changes above & below
3) Creates interop problems

Goal



Modified from: Steve Deering
http://www.iab.org/Documents/hourglass-london-ietf.pdf

Future Internet – The Challenge

- Society's needs for an IT infrastructure may no longer be met by the present trajectory of incremental changes to the current Internet
- Society needs the technical community to create the trustworthy Future Internets that meet the needs and challenges of the 21st Century.
  - In the 1960's the telephone industry saw little need for packets→ packets gave society the Internet
    - What is the next breakthrough?

Opportunity

- New ground is being broken in wide range of core networking areas such as:
  - Identities, naming, addressing, network management, high-speed deep packet inspection, access and transport technologies, sensing, content and media delivery, network applications, and services
- Still far from the understanding needed to identify the coherent architectural alternatives from among emerging ideas

Future Internet Architectures

- FIA's needs to be:
  - Trustworthy – broadly defined
    - Security, privacy, robust, reliability, and usability
  - Economic viable
  - Configurable/manageable
  - Scalable – broadly defined
    - New applications, new technologies, and policies
- FIA's need to be understood:
  - Based on architectural principles
  - Defined architectural invariants
  - Exposed component interactions
  - Designed for predictable performance
  - Identified metrics for architectural evaluation and a path to the comparison of alternative architectures
- FIA's need to recognize technology trends
  - Ubiquitous high speed wireless (10's, 100 Mb/s to Gb/s)→ mobility
  - Cloud computing

Future Internet Architecture Projects

- **Named Data Networking**
- Underlying architectural principles
  - Packets indicate what (content)
    not who (IP address)
  - Packet is a <name, data, signature>
  - Securing named data potentially allows trust to be more user-centric.
  - http://www.named-data.net/index.html

Future Internet Architecture Projects

- **MobilityFirst**
- Underlying architectural principles
  - Mobility is the norm
  - The architecture uses generalized delay-tolerant networking to provide robustness even in presence of link/network disconnections. GDNT integrated with the use of self-certifying public key addresses provides an inherently trustworthy network.
  - http://mobilityfirst.winlab.rutgers.edu

Future Internet Architecture Projects

- **eXpressive Internet Architecture** (XIA)
- Underlying architectural principles
  - XIA offers support for communication between current communicating principals--including hosts, content, and services--while accommodating unknown future entities.
  - For each type of principal, XIA defines a narrow waist that dictates the application programming interface (API) for communication and the network communication mechanisms.
  - XIA enables flexible context-dependent mechanisms for establishing trust between the communicating principals.
  - http://www.cs.cmu.edu/~xia/

Challenges

- Challenges
  - Trust
  - Network and configuration management
  - Scalability and control of system complexity
  - Predictable performance
  - Performance evaluation and comparison of different architectures
- Approaches and mechanisms are now being woven together into coherent, overarching candidate designs for a future Internet.