Network Layer

Study-Ready Notes

Compiled by Andrew Photinakis

November 15, 2025

Contents

Chapter 4: The Network Layer — Data Plane

1 4.1 Overview of Network Layer

1.1 4.1.1 Forwarding and Routing: The Data and Control Planes

Two key functions of the network layer, **forwarding** and **routing**, map directly to the **data plane** and **control plane**, respectively.

1.1.1 1. Concept Overview: The Two Core Functions

Welcome. In our last lecture, we completed our study of the transport layer, which provides a *logical* communication service between *processes* running on different hosts. Now, we move down the stack to examine the network layer, which provides the underlying *host-to-host* communication service that the transport layer relies on.

The network layer's primary role seems simple: to move packets, called **datagrams**, from a sending host to a receiving host. However, to accomplish this, the network layer must perform two distinct and critical functions: **forwarding** and **routing**.

1. Forwarding
    - Is a *local* action.
    - The process of taking a packet that has arrived on one of a router's input links and moving it to the appropriate output link.
2. Routing (Control Plane Function)
    - Is a *network-wide* process that determines the end-to-end paths that packets take from a source host to a destination host.

1.1.2 Forwarding: The "Data Plane"

1. Definition
    - Forwarding refers to the router-local action of transferring a packet from an input link interface to the appropriate output link interface.
    - Forwarding is the primary function of the network layer's **data plane**, which includes per-router operations performed on a packet as it moves through the router.
2. Mechanism
    - A packet arrives at a router's input link.
    - Router examines one or more fields in the packet's header.
    - Uses header values to index into its forwarding table.

- Value found in the forwarding table entry indicates the router's output link interface to which the packet should be forwarded.

3. Timescale & Implementation
   - Forwarding is a fast (nanosecond) action that must be performed for every packet.
   - Therefore, it is implemented in hardware.
4. Analogy
   - Forwarding is like navigating a single highway interchange or roundabout: the driver makes a local, split-second decision based on signs.

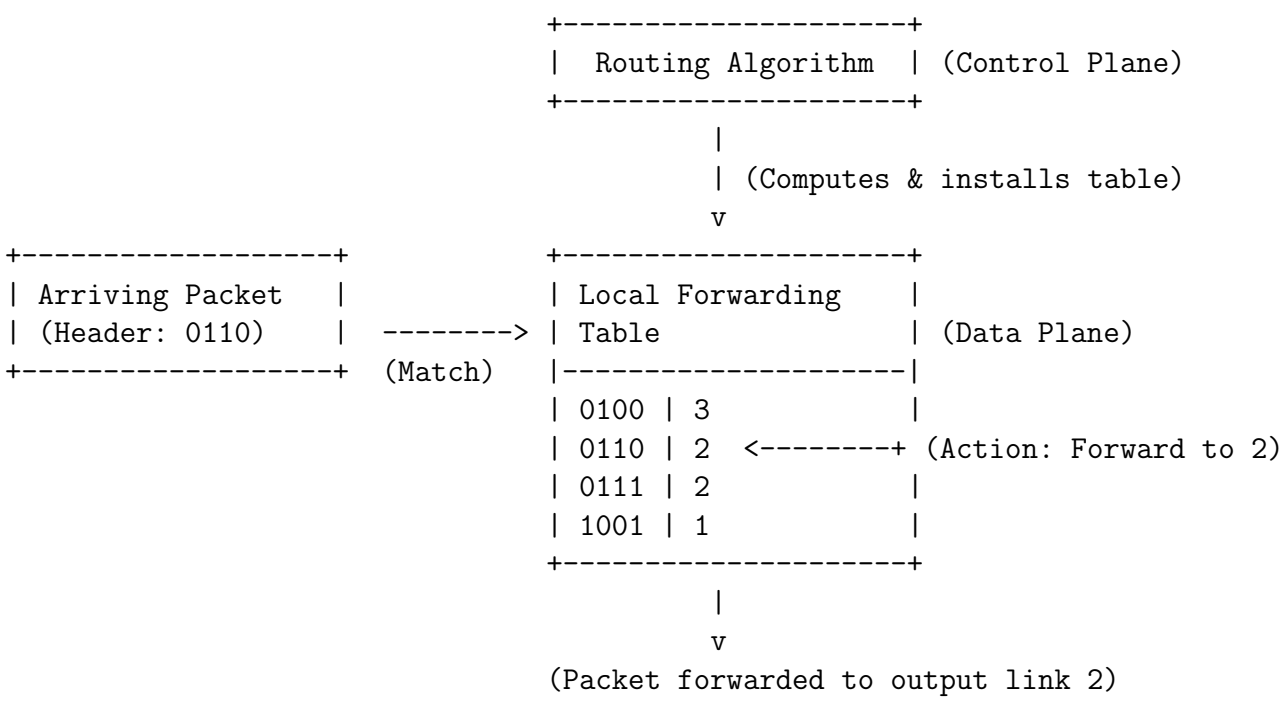
1.1.3 Routing (Control Plane Function)

1. Definition
   - Routing refers to the network-wide process of calculating and determining the end-to-end paths that datagrams follow from source to destination.
   - It is the primary function of the **control plane**, which governs network-wide routing logic.
2. Mechanism
   - Routing is accomplished by routing algorithms that calculate paths for datagrams.
   - Routing decisions are installed in the forwarding tables of routers.
3. Timescale & Implementation
   - Routing computations occur on much longer timescales (seconds to minutes).
   - Typically implemented in software due to their complexity.
4. Analogy
   - Routing is like planning an entire trip from Pennsylvania to Florida: the routing algorithm (driver) consults a map to select the best end-to-end path.
   - Each router on the path forwards packets according to the plan.

1.1.4 4.1.1.1 The Forwarding Table

The routing (control plane) controls forwarding (data plane) via the **forwarding table**.

- **Function:** Router examines header fields of incoming packets, indexes into the forwarding table, and forwards the packet according to the table entry.
- **Origin:** Entries are computed and installed by the control plane. Changes in topology or costs trigger updates.



1.1.5 4.1.1.2 Approaches to the Control Plane

Approach 1: Traditional Per-Router Control

- Control plane runs in each router.
- Routing components communicate via routing protocols to compute forwarding tables.
- Distributed architecture.

Approach 2: Software-Defined Networking (SDN)

- Control plane is physically separated from data plane.
- A remote controller computes forwarding tables and installs them on all routers.
- Routers act as pure data-plane devices.
- Controller runs in software, typically on a reliable data center, and communicates with routers using protocols like OpenFlow.

1.1.6 4.1.1 Summary

- **Forwarding:** Local, fast, hardware-level data-plane function.
- **Routing:** Network-wide, slower, software-level control-plane function.
- **Data Plane:** Handles forwarding using the forwarding table.
- **Control Plane:** Computes paths and installs forwarding table entries.

- **Forwarding Table:** Links control plane and data plane.
- **Per-Router Control:** Traditional distributed routing model.
- **Logically Centralized Control (SDN):** Centralized controller computes forwarding tables.

1.2 4.1.2 Network Service Model

1.2.1 1. Concept Overview: The "Contract" of the Network Layer

The network service model defines the characteristics of end-to-end delivery of packets between sending and receiving hosts. It represents the "contract" between the network and transport layers:

- Will the packet be delivered?
- Will it arrive intact?
- Will it arrive in order?
- Will it arrive within a certain time?
- Is there a minimum throughput guarantee?

1.2.2 2. Technical Mechanism: A "Menu" of Possible Services

Potential services a network layer could offer:

- **Guaranteed Delivery:** Every packet eventually arrives.
- **Guaranteed Delivery with Bounded Delay:** Delivery occurs within a specified time.
- **In-Order Packet Delivery:** Packets arrive in sending order.
- **Guaranteed Minimal Bandwidth:** Network behaves like a dedicated link with a specified bit rate.
- **Security:** Encrypt all datagrams for confidentiality.

1.2.3 3. Practical and Intuitive View: The Internet's *Actual* Service

The Internet provides a single service: **best-effort service**:

- Delivery is not guaranteed; packets may be lost.
- In-order delivery is not guaranteed; packets may arrive out of order.
- Delay is not guaranteed; packets may experience long delays.

- Bandwidth is not guaranteed.

Design philosophy: *Keep the network core simple.* Reliability, ordering, and congestion control are pushed to the edges (transport/application layers).

1.2.4 4. Terminology Clarification: Switches vs. Routers

- **Packet Switch:** Generic device moving packets from input to output interfaces.
- **Link-Layer Switch:** Layer 2 device forwarding based on MAC addresses.
- **Router:** Layer 3 device forwarding based on network-layer addresses (IP addresses).

Focus of this chapter: **routers** (Layer 3 devices).

1.2.5 4.1.2 Summary

- **Network Service Model:** Defines network guarantees for transport layer.
- **Best-Effort Service:** Internet's minimalist model with no delivery, order, delay, or bandwidth guarantees.
- **Packet Switch, Link-Layer Switch, Router:** Key terminology distinctions.
- Minimalist model pushes intelligence to end systems.

1.3 Core Role and Architectural Placement

1. Network layer is third layer of the internet protocol stack
2. Resides between the transport layer and link layer
3. Fundamental role is to move network layer packets, called datagrams, from a sending host to a receiving host.
4. Host-to-Host Communication
    - While transport layer provides logical communication between processes, network layer provides logical communication between hosts
5. Universal Implementation
    - Transport and application layers run only on end systems (hosts)
    - Network layer runs in every host and router in network
    - It's esential b/c routers must examine datagram headers to perform their forwarding function
6. Router Protocol Stack

- Routers are network-layer (Layer 3) devices
- Consist of "truncated" protocol stack, implementing physical, link, and network layers
- Do not implement transport or application layers
- Sole purpose is to forward datagrams, not run end-user applications

7. At Each Stage
   (a) Sending Host
      - Network layer takes segments from transport layer, encapsulates them into datagrams, and sends these datagrams to its nearby router

   (b) Receiving Host
      - Network layer receives datagrams from its nearby router, extracts transport-layer segments, and delviers them up to transport layer

   (c) Routers
      - Core function to examine header of an arriving datagram and forward it to appropriate output link
      - Routers are 'Layer 3' devices
      - Typically have a 'truncated' protocol stack, implementing up to network layer but not transport or application layers

1.4 Planes

1.4.1 Data Plane

- Performs per-touer function of forwarding datagrams from a router's input link to its appropriate output link

1.4.2 Control Plane

- Performs network-wide logic that controls how datagrams are routed along an end-to-end path from source to destination host

2 4.2 What's Inside a Router?

2.1 Concept Overview: Unboxing the Data Plane

1. Primary function to transfer packets from incoming links to appropriate outgoing links
2. Task split smong four interconnected componenets
   (a) Input Ports (Hardware):
      i. Perform physical, link, and data-plane lookup for incoming packets

ii. Is entry point for packets into router

iii. Physically terminates incoming link, performs link-layer tasks to "unwrap" the packet, and perform lookup function to determine packet output port using forwarding table

iv. Control packets are passed "up" to routing processor

(b) Switching Fabric (Hardware):

i. "Heart" of router

ii. Purpose to connect input ports to output ports

Internal mechanism that moves packets from input ports to output ports

(c) Output Ports (Hardware):

i. Exit point for packets leaving router

ii. Receives packets from switching fabric, queues them, performs necessary link-layer and physical-layer functions to "wrap" packet into new frame and finally transmits frame onto outgoing link

Store, queue, and transmit packets received from switching fabric onto outgoing link

(d) Routing Processor (Software):

i. "Brain" of router

ii. Primary component of control plane.

iii. Implemented in software and runs on traditional CPU

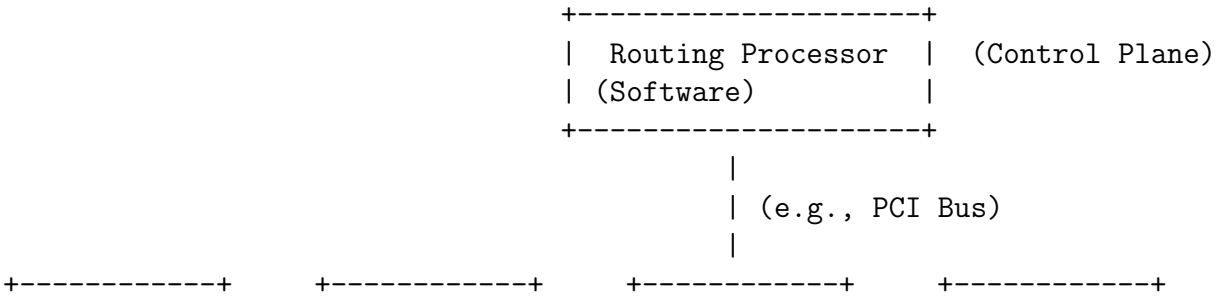
iv. Is not directly involved with main forwarding path

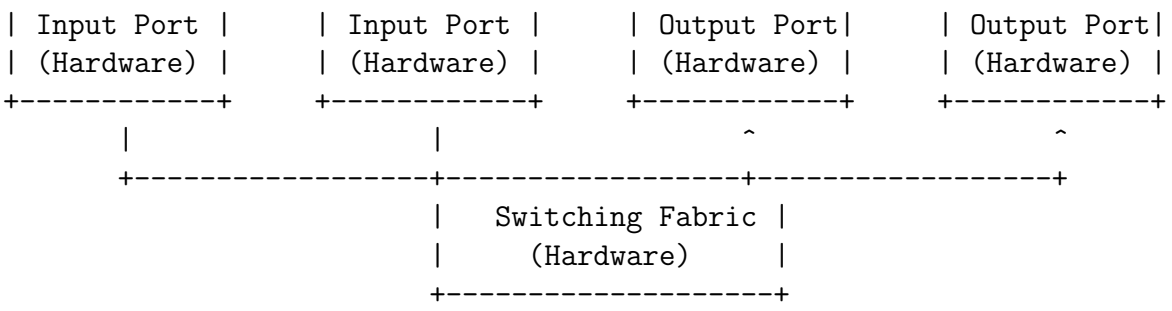
A. In traditional router, routing processor executes routing protocols, maintains routing tables, and uses info to compute and update forwarding table

B. In SDN router, routing processor is responsible for communicating with remote SDN controller, receiving forwarding table entries from it, and installing entriies into input ports

Executes control-plane functions like routing protocols and maintains/computers forwarding table

2.2 The Four Components of a Router





2.3 Data Plane (Hardware) vs Control Plane (Software)

Essential to understand why such components are built the way they are

- Data Plane
  1. Reason: Speed, executes in nanosecond timescale
  2. Example:
     (a) Consider 100 Gbps input link
     (b) New packet can arrive every few nanoseconds
     (c) For a min-sized 64-byte (512 bit) IP datagram, router only has 5.12 nanoseconds (512 bits / 100 Gbps) to process before next one arrives
  3. Per-packet forwarding action far too fast for software-based implementation
- Control Plane
  1. Reason: Complexity and Timescale, control-plane functions (like new route using Dijkstra's algo or commuinicating with SDN controller) are complex and happen on much slower timescale - milliseconds or even seconds
  2. Well suited for standard CPU running a software program.

2.4 Roundabout Analogy

- Input Port: Entry road, which includes an attendant (lookup function)
- Switching Fabric: roundabout itself
- Output Port: exit ramp
- Destination-based Forwarding: Tell attendant find destination (i.e. Florida) and attendant looks it up, tells you "take 3rd exit", and you're on the way
- Generalized Forwarding: Attendants decision is more complex
  1. "You're from NY? Take the slow road. You're from this state (license plate)? Take highway. Your car is not road-worthy? You're blocked"
- Bottlenecks: Analogy illustrates router's potential performance bottlenecks

1. Lookup bottleneck: Attendant is too slow. Cars pile up on entry road (input port)
2. Switching bottleneck: Roundabout is too small or slow. Cars get stuck in middle (switch fabric)
3. Output bottleneck: Too many cars want same exit ramp. Cars pil up on exit ramp (output port)

2.5 4.2.1 Input Port Processing and Destination-Based Forwarding

2.5.1 TBD

Input port performs pipeline of functions
[Line Term] –¿ [Link-Layer Processing] –¿ [Lookup, Fwd, Queuing] –¿ [To Switch Fabric]

Line Term –¿ [Link-Layer Processing] –¿ [Lookup, Fwd, Queuing] –¿ [To Switch Fabric]

1. Y
2. Line Termination: physical layer function, plus that connects physical wire to router
3. Data link processing: link layer function, de-encapsulates network-layer datagram from link-layer frame
4. Lookup, forwarding, and queueing: Core data-plan and network-layer function, lookup stage is most critical

- Lookup Function and Shadow Copy

  1. Concept: router needs to consult with forwarding table to decide output port for packet
  2. Mechanism:
     (a) Routing processor computes forwarding table
     (b) To avoid bottleneck, table is then copied to input ports
     (c) input port hardwarse uses local "shadow copy" to perform lookup line speed

- Destination-Based Forwarding and Longest Prefix Matching

  1. Problem: how do you implement this lookup? Brute-force forwarding table would need one entry for every possible 32-bit IP address ($2^{32} = 4{,}294{,}967{,}296$), which is impossible
  2. Solution: Prefix matching: Routers don't store entries for individual hosts, store entries for prefixes (subnets)

- Longest Prefix Matching: A major subtlety arises, what is dest address matches multiple entries?

1. Conflict: Given destination address can match multiple prefixes in table (e.g. it might match a /21 prefix and more specific /24 prefix)
2. Rule: router must use long matching entry in table to determine appropriate output link
3. Intuition: longest prefix is most specific route available for such destination
4. Mechanism: search must be performed in nanoseconds
   (a) Special hardware like Ternary Content Addressable Memories (TCAMs) often used for constant-time lookup, or high-speed data structures like heaps are employed
5. Result: 24-bit match is longer than 21-bit match, so packet is forwarded to Interface 1

- "Match Plus Action" Abstraction

1. Input port processing is specific example of powerful, general abstraction called "match plus action"
   (a) Match: look at header fields (i.e. dest IP address)
   (b) Action: do something (i.e. forward to an output port)
2. Same abstraction sued in many network devices
   (a) Routers: Match(dest ip) –¿ Action(forward)
   (b) Switches: Match(dest MAC) –¿ Action(forward)
   (c) Firewalls: Match(src ip, src port, dest ip, dest port) –¿ action(permit/deny)
   (d) NAT: match(src ip, src port) –¿ action(rewrite ip/port)

2.6 4.2.2 Switching Fabric

2.6.1 Three Switching Techniques

Switching fabric is component that actually moves packets from input to output ports via three ways below

- Switching via Memory

1. How it works:
   (a) Packets would arrive at input port, which would trigger an interrupt
   (b) CPU would copy packet into main system memory
   (c) CPU would perform lookup, find output port, and copy packet from memory to output port's buffer
2. Bottleneck
   (a) Routers total forwarding throughput is limited by memory bandwidth
   (b) Since packet must be written into memory and then read out, max throughput is B/2, where B is memory bus speed

   (c) Only one packet processed at a time
3. How it works (modern)
   (a) CPU no longer used
   (b) Instead, input port's hardware processor performs lookup and writes packet directory to output ports memory

- Switching via a Bus
  1. How it works:
     (a) All input ports and output ports hsare single, common bus
     (b) Input port receives a packet, performs lookup, and prepents a special "switch-internal label" to packet (like saying "this is for output 2"), and sents it onto the bus
  2. All output ports receive packet, but only port that matches label will keep and transmit it
  3. Bottlenecks
     (a) Bus is shared resource, only one packet can cross bus at a time
     (b) Routers total throughput limited to bus speed
     (c) Fine for small enterprice or home router, but not for high-speed core routers
- Switching via an Interconnection Network (Crossbar)
  1. How it works:
     (a) This is highest-performance solution
     (b) Crossbar switch is a grid of 2N buses (N horizontal, N vertical) connecting N input ports to N output ports
     (c) A "crosspoint" (a transistor switch) exists at every intersection
  2. When packet at input port A needs to go to output port Y, switch fabric controller closes crosspoint connecting A's horizontal bus to Y's vertical bus
  3. Key Property (parallelism)
     (a) A packet from input B can simutaneously be forwarded to output X by closing (B, X) crosspoint
     (b) Allows multiple packets to be transferred in parallel, as long as they are going to different output ports
  4. Non-Blocking
     (a) Crossbar switch is non-blocking – a packet detined for an idle output port is inever prevented from reaching it
     (b) However, if two packets from A and B both want to go to port Y, one will have to wait
     (c) This is output port contention, not a limitation of fabric itself

2.7 4.2.3 Output Port Processing

2.7.1 Overview

- Output port procesing is mirror image of input port procesing. Takes packets that have been transferred across switch fabric and stores them in its own output port queue. Functions are:

    1. Queueing: Stores packets recevied from fabric. Where packet scheduling (FIFO, priority, etc.) is performed to decide which packet to send next
    2. Link-Layer Processing: Encapsulate IP datagram into link-layer frame (i.e. add Ethernet header and trailer)
    3. Line Termination: physical-layer function of transmitting bits onto outgoing link

2.8 4.2.4 Where Does Queuing Occur?

2.8.1 Overview

This is a critical topic in router performance. Packet queues form when the packet arrival rate exceeds the forwarding rate. This can happen at both input and output ports. These queues are where packet delay is incurred and packet loss (when a queue overflows) occurs.

- Output Queueing

    1. Occurs when switching fabric is fast, but rate of packets arriving at output port (from multiple input ports) exceeds line speed of single output port
    2. Even a very fast switching fabric can feed packets to an output port faster than output link can transmit them, leading to a bottleneck at output port buffer
    3. Packet Loss and Buffer Management
        (a) If arrival rate to toutput queue persists, buffer will fill up
            i. Drop-Tail: default policy, when queue is full, newly arriving packet is dropped
            ii. Active Queue Management (AQM)
                A. More intelligent policy
                B. Proactively drop or mark (e.g. with ECN) packets before buffer is full
                C. Provides early congestion signal to senders (like TCP)
                D. Random Early Detection is classic example

- Input Queueing

    1. Occurs when switching fabric is slower than combined speed of input ports (e.g. $R_{\text{switch}} < N \cdot R_{\text{line}}$)
    2. Packets must wait in a queue at input port for their turn to cross fabric
    3. Head-of-the-Line Blocking

   (a) Specific problem in input-queued switches
   (b) Packet at head of an input queue is blocked from moving across fabric because desired output port is occupied by another packet (or because fabric is too slow)
   (c) Forces all subsequent packets in that input queue to wait, even if their desired output port is free
   (d) Can drastically limit throughput

- Route Buffer Sizing: "How much buffering is 'enough'"?

   1. Too little buffer = high packet loss, too much buffer = high queueing delay
   2. Rule of Thumb 1 (tranditional)
      (a) B = RTT * C, where B = buffer size, RTT = average round-trip time, C = link capacity
      (b) Example: 250ms RTT, 10 Gbps Link –¿ B = 0.25s * 20 Gbps = 2.5 Gbits of buffer, which is huge and expensive amount of memory
      (c) Purpose to fully utilize TCP conncetion pipeline without ricking premature packet drops
   3. Rule of Thumb 2 (modern)
      (a) B = (RTT * C) / sqrt(N), where N = number of indepdent TCP flows
      (b) When large number of flows (N) are passing through a link, their traffic bursts tend to average out (statistical multiplexing)
      (c) Required buffer size to achieve good throughput and loss performance scales down with square root of N

2.9 4.2.5 Packet Scheduling

2.9.1 Overview

if an output port queue has mutliple packets waiting, packet schedulign discipline decides which packet to transmit next

- FIFO

   1. Simplest
   2. packets are transmitted in exact ordert they arrived
   3. FIFO provides no way to give priority to important packets (like VoIP) over less-important packets (like email)

- Priority Queueing

   1. packets are classified into priority classes upon arrival, each class has its own queue

2. Scheduler always transmits a packet from highest priority queue that has packets in it
3. Packets within same priority class are served FIFO
4. Preemption: typically non-preemptive, once a packet has started transmitting, it's not interrupted
5. Problem: Starvation, high priority flow can completely block all low-priority flows

- Round Robin (RR) and Weighted Fair Queuing (WFQ)

1. "taking-turns" scheduler, packets are classified, schedular cycles through classes: send one packet from class 1, then one from class 2, then one from class 3, then back to class 1
2. Work conserving, if a class's queue is empty, scheduler doesn't wait, it just skips to next class
3. Weighted Fair Queueing
   (a) Each class i is assigned a weight, w_i
   (a) scheduler is still work conserving and round-robin
   (b) Service guarantee
      i. WFQ guarantees that each class i will receive a fractice of total bandwidth equal to: R * $(w_i/\sigma(w_j))$, $where sum is over all classes that' recurrently active (have packet$

3 4.3 The Internet Protocol (IP): IPv4, Addressing, IPv6, and More

3.1 Overview

- Delves into foundational protocols and addressing schemes of the Internet Protocol (IP) that enable the Internet's network layer.
- IP Protocol provides essential, unreliable, best-effort delviery service that underpins all higher-layer communcation
- IP Protocol is hte only network-layuer protocol used in the Internet

3.1.1 IP Protocol

- Is the only network-layer protocol in the internet
- Is the "narrow waist" of the hourglass, the single protocol that everything else - every transport protocol (TCP, UDP) and every applciaiton (HTTP, DNS) - must run on top of
- In turn must be able to run over any link-layer technology (Ethernet, WiFi, 4G, etc)

3.2 4.3.1 IPv4 Datagram Format

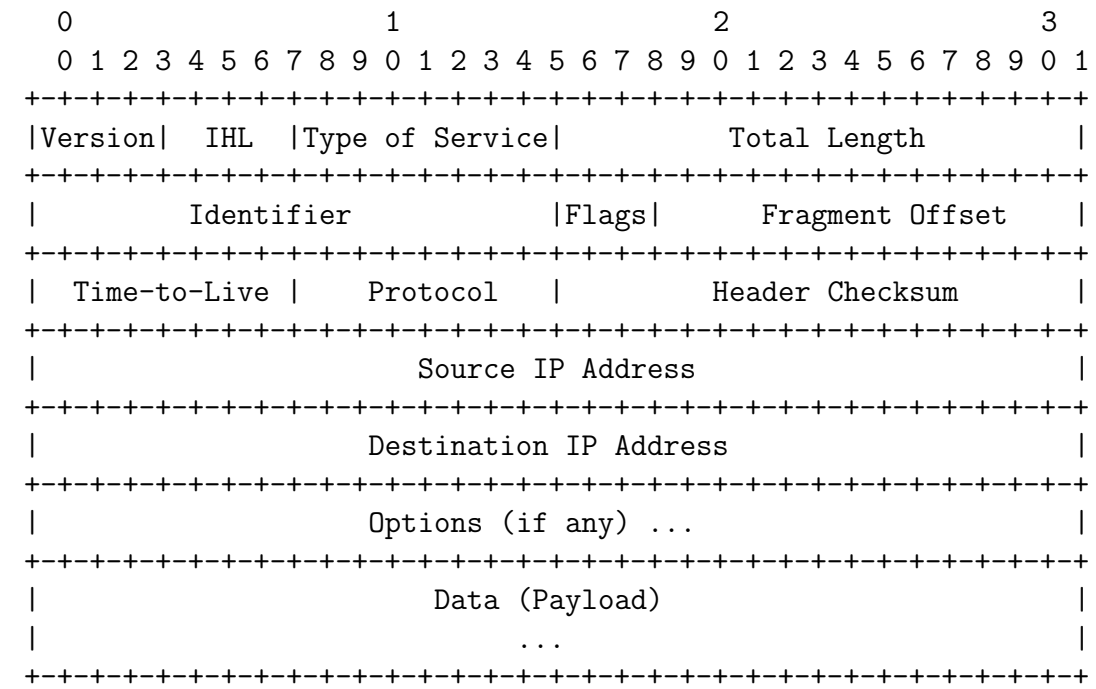
3.2.1 Concept Overview

Datagram:

1. The Internet's network-layer packet is known as a **datagram**.
2. Fundamental to understanding IP's operations.
3. Format is defined in **RFC 791**.
4. IPv4 datagram has a variable-length header:
   - Typically 20 bytes long when no options are present.
   - Followed by a payload (data field).
5. Composed of two main parts:
   (a) **Header:** Contains all information a router needs to make forwarding decisions.
   (b) **Data (Payload):** Contains the data being transported, typically a transport-layer segment (like TCP or UDP).

3.2.2 Technical Mechanism: The Datagram Format

- A textual representation of the IPv4 header:



- Key Header Fields and Their Purpose:

1. **Version (4 bits):**
   (a) Specifies IP protocol verstion
   (b) For IPv4, this is always 4.

   (c) Route examines this field first to know how to parse the rest of the header (e.g. to know that it should expect an IPv4 header, not IPv6 header)

2. **IHL (Internet Header Length) (4 bits):**
   (a) Specifies length of IP header in 32-bit words
   (b) Field is necessary because 'Options' field can make header variable-length
   (c) Required to find stasrt of a data/payload fields
   (d) Typical heder length is 20 bytes (5 words)
3. **Type of Service (TOS) (8 bits):** Indicates priority and routing preferences.
   (a) Included to allow different types of IP datagrams (e.g. real-time vs non-real time) to be distinguished and serviced differently (e.g. with priority or low-delay)
   (b) Bits are also used in Explicit Congestion Notification (ECN)
4. **Datagram Length (16 bits):** Length of the entire datagram (header + payload).
   (a) Total length of IP datagraam (header + payload) in bytes
   (b) Practial View:
      i. A 16-bit field allows for theoretical maximum datagram size of 65,535 bytes
      ii. However, datagrams are rarely larger than 1,500 bytes because they must fit within the Maximum Transmission Unit (MTU) of udnerlying link-layer to avoid fragmentation
5. **Identifier, Flags, Fragmentation Offset (32 bits total):** Identifies fragments of the original datagram.
   (a) Fields are all used for IP fragmentation
   (b) If router needs to forward a datagram onto a link with an MTU smaller than datagram's size, it can "fragment" datagram into multiple smaller datagrams
   (c) Such smaller datagreams are reassembled only at final destination
   (d) This is a slow, complex process, and IPv6 gets rid of it entirely for routes
6. **Time-to-Live (TTL) (8 bits):** Limits the datagram's lifetime to prevent infinite loops.
   (a) Critical safety mechanism to ensure datagrams don't circulate forever in network (e.g. in a routing loop)
   (b) TTL field is decremented by one at every router is passes through
   (c) If router receives a datagram with a TTL of 1, it decrements it to 0, discards the datagram, and sends an ICMP error message back to source
7. **Protocol (8 bits):** Specifies the transport protocol in the payload (e.g., TCP, UDP).
   (a) Binds network layer to transport layer

   (b) Used only at final destination host to indicate which transport-layer protocol should receive datagram's payload
      i. 6 = TCP
      ii. 17 = UDP
      iii. 1 = ICMP (Internet Control Message Protocol)

8. **Header Checksum (16 bits):** Error-checking for the header.
   (a) Performs error detection on header only, not on payload
   (b) Computed using 1s complement arithmetic over 2-byte units of the header
   (c) Must be recomputed at every router because TTL field changes
   (d) Why only header?
      i. Transport layer (TCP/UDP) performs its own checksum on payload
      ii. TTL field changes at every route. Means header must be altered. Checksum must be recomputer and restored at every single router, which is time consuming
9. **Source and Destination IP Address (32 bits each):** Addresses of originating and final destination interfaces
   (a) Fields contain 32-bit IP addresses of original src and final dest
   (b) Src address inserted by sending host
   (c) Dest address obtained from a DNS lookup
10. **Options (if any) (variable length):** Optional fields for control or security.
   (a) Field allows IP header to be extended, but rarely used
   (b) Can be problematic because makes header length variable, complkicating and slowing down router processing
11. **Data (Payload):** Actual transported data (e.g., TCP/UDP segment).

3.3 4.3.2 IPv4 Addressing

3.3.1 Concept Overview: Interfaces and Subnets

- IP Addresses and Interfaces

1. IP addr is 32 bits long
2. IP addr associated with a router or host interface, not device itself
3. Router has multiple interfaces (one per link), and thus multiple IP addrs
4. Hosts typically have one interface (and one IP address) connecting them to network
5. Addresses are writting in dotted-decimal notation

3.3.2 Technical Mechanism: The IP Subnet

- IP addresses not assigned randomly
- Portion of address is determined by network interface connected to
- /24 notation is subnet mask
    1. Indicates that leftmost 24 bits define subnet, and remaining (32-24) = 8 bits identify specific interfaces on that subnet
- Recipe for Identifying Subnets
    1. Detach every interface from its host or source
    2. This creates "islands" of isolated networks
    3. Each island is a subnet
- Subnets
    1. IP subnet is a network that connects multiple host interfaces and router interfaces,
    2. Forms isolated network without intervening routers
- Subnet Addressing
    1. Interfaces on given subnet share same high-order bits of their IP addresses
    2. Notation (CIDR)
        (a) A subnet address is denoted by form a.b.c.d/x, where /x indicates number of high-order bits that constitute network portion of address

3.3.3 Concept: CIDR (Classless Inter-Domain Routing)

1. Is a flexible IP address assignment strategy that replaced older class-based system
2. Introduced /x notation, where x specifies how many bits of the address form network prefix

- Mechanism
    1. CIDR generalizes traditional subnet and hsot division
        (a) In address a.b.c.d/x
        (b) x most significant bits are prefix
        (c) Remaining 32 - x bits are host portion
    2. Allows networks to have variable-length prefixes, making address allocation more flexible
- Scalability via Address Aggregation

1. Organizations are assigned contiguous blocks of IP addresses with a common prefix
2. Routers outside of org only need one entry in their forwarding tables.
   (a) "To reach any address startingt with 200.23.16.0/20, forward to that orgs ISP"
3. Ability to use single prefix to advertise many networks = address aggregation (or route aggregation)
4. Essential for keeping global routing tables small and manageble

3.3.4 Obtaining and Managing IP Addresses

1. Getting a block of addresses (for an ISP or large org)
   (a) Global authority is ICANN (Internet Corporation for Assigned Names and Numbers)
   (b) ICANN allocates address blocks to Regional Internet Registries (RIRs)
   (c) An ISP (like Comcast, or Verizon) gets its address blocks from its RIR
   (d) An org gets its address blocks from its ISP
   (e) Hierarchical allocation allows for route aggregation
2. Getting a host address (DHCP)
   (a) Once org has blocks of addresses (e.g. 68.85.2.0/24) it needs to assign individual addresses to its hosts
   (b) Done automatically by Dynamic Host Configuration Protocol (DHCP)
   (c) DHCP - is a plug and play protocol. When your laptop connects to a network, uses DHCP to automatically get
      i. Its IP address (e.g. 68.85.2.101)
      ii. Its subnet mask (e.g. /24)
      iii. IP address of default gateway (first hop router)
      iv. IP address of local DNS server
   (d) How DHCP works?
      i. DHCP Discover
         A. New host client sends broadcast message (Dest IP: 255.255.255.255, Souce IP: 0.0.0.0)
         B. Asks "is there a DHCP server out there?"
      ii. DHCP Offer
         A. A DHCP server (typically on the router) receives the discover and replies with offer message
         B. Includes proposing an IP address, lease time, etc.

iii. DHCP Request
   A. Client formally requests offered address by sending a broadcast "request" message
   B. This tells any other servers that sent offers that they weren't chosen
iv. DHCP ACK
   A. Server confirms allocation with a "DHCP ACK" message
   B. Client can now use IP address

3.4 4.3.3 Network Address Translation (NAT)

3.4.1 Concept Overview

DHCP and CIDR helped, but IPv4 address space still faced exhaustion

- NAT
  1. Designed for SOHO (Small Office/Home Office) networks where many internal devices share single public IP address
- NAT-enabled Router
  1. To the Internet: Entire home network looks like single device with a single IP address
  2. To the Home: Home network is a private network, using a private address space (e.g. 10.0.0.0/24)
     (a) These addresses are not routable on public internet and are reused by millions of homes

3.4.2 Technical Mechanism: NAT Translation Table

How does NAT map one public address to many private addresses? Uses port numbers.

NAT router maintains NAT Translation table. This table mapes (Private IP, Private Port) pairs to (Public IP, Public Port) pairs

1. Step-by-Step Operations
   (a) Outbound packet
      i. Your laptop (10.0.0.1) sends packet to google.com (128.119.40.186)
      ii. Your OS picks a source port (e.g. 3345)
      iii. Packet arriving at NAT router: (Src IP: 10.0.0.1, Src Port: 3345), (Dest IP: 128.119.40.186, Dest Port: 80)
      iv. NAT Router
         A. Saves this mapping, then creates a new, unique public source port (e.g. 5001)
         B. Adds it to table: (10.0.0.1, 3345) ¡–¿ (138.76.29.7, 5001)

v. Router sends modified packet to Internet. Web server (google.com) thinks request came from (138.76.29.7)

vi.

(b) Inbound Packet

i. google.come sends a replay, which is addressed to source of packet it received

ii. Packet arriving at NAT router

A. Src IP: 128.119.40.186, Src Port: 80

B. Dest IP: 138.76.29.7, Dest Port: 5001

iii. NAT Router

A. Looks up destination (138.76.29.7, 5001) in translation table.

B. Finds matching internal address: (10.0.0.1, 3345).

C. Rewrites the packet header again:

D. Src IP: 128.119.40.186, Src Port: 80

E. Dest IP: 10.0.0.1, Dest Port: 3345

F. The router forwards this packet into the home network, and it arrives at your laptop (10.0.0.1) on the correct port (3345).

3.4.3 Practical and Intuitive View: Controversy

NAT is highly controversial among network purists

- Pros

1. Saves IP Addresses: A single public IP can be shared by thousands of private IPs
2. Security:
   (a) Internal devices are not directly addressable from outside world
   (b) Incoming packet is dropped unless there's an existing entry in translation table
3. Easy Management: Can add devices to home network without needing a new public IP from your ISP

- Cons

1. Breaks End-to-End Principle:
   (a) Port numbers are a Layer 4 (transport) concenpt meant to identify processes
   (b) NAT is Layer 3 (network) device that is reading and rewriting Layer 4 fields
   (c) Violation of layered architecture
2. Breaks P2P and Servers
   (a) External users can't connect to you since no public, routable address
   (b) Breaks P2P apps (like file sharing or games) and makes it harder to run a server from home

   (c) Requires special "NAT Traversal" techniques or manual "port forwarding" configurations

3. Middlebox
   (a) NAT is a "middlebox", a device that does more than just routing
   (b) Is complex, stateful device in middle of network, which violdates original internet design of simple core and smart edges

3.5 4.3.4 IPv6

3.5.1 Concept Overview

IPv6 was developed primarily to address the impending exhaustion of the 32-bit IPv4 address space. It solves the IPv4 address exhaustion problem by introducing 128-bit addressing and a simplified, fixed-length header.

3.5.2 IPv6 Datagram Format

- **Key Changes from IPv4**

  1. **Expanded Addressing (128 bits)**
     (a) Address size increased from 32 to 128 bits.
     (b) Provides 2^{128} possible addresses.
  2. **Streamlined 40-byte Fixed Header**
     (a) IPv6 header is a fixed 40 bytes.
     (b) Simplifies router processing.
  3. **Flow Labeling**
     (a) New field for identifying flows (e.g., a video stream).
     (b) Allows routers to provide special treatment (QoS/routing) to packets belonging to the same flow.

- **IPv6 Header Fields**

  1. **Version (4 bits):** Always set to 6.
  2. **Traffic Class (8 bits):** Like IPv4 TOS; supports priority/QoS.
  3. **Flow Label (20 bits):**
     (a) Labels packets belonging to a specific flow.
     (b) Routers can process these packets as a group.
  4. **Payload Length (16 bits):** Length of data after the fixed 40-byte header.
  5. **Next Header (8 bits):**
     (a) Identifies the protocol of the payload (e.g., TCP, UDP).
     (b) Key to streamlined header structure.

   (c) If IPv6 options are included, this field points to an *options header*.

6. **Hop Limit (8 bits):**
   (a) Same as IPv4 TTL.
   (b) Decrements at each router; packet dropped when it reaches 0.
7. **Source and Destination Addresses (128 bits each):**
   (a) Much larger address space.
   (b) Supports unicast, multicast, and anycast addressing.
8. **Data (Payload)**

3.5.3 IPv6 Datagram Format (Structured View)

Bits 0-3	Bits 4-11	Bits 12-31
Version (6)	Traffic Class	Flow Label
Payload Length		Next Header
Hop Limit		
Source Address (128 bits)		
Destination Address (128 bits)		
Data (Payload)		

3.5.4 Fields Dropped or Changed from IPv4

- **Fragmentation/Reassembly**
  - Routers no longer fragment packets.
  - If a packet is too large, router sends a *Packet Too Big* ICMP message.
  - Sender performs Path MTU discovery.
- **Header Checksum**
  - Removed—redundant due to link-layer and transport checksums.
  - Removal speeds up per-hop processing (TTL/Hop Limit changes no longer require recalculating checksum).
- **Options**
  - Removed from base header.
  - Now included via extension headers pointed to by the Next Header field.

3.5.5 Transitioning from IPv4 to IPv6

The Deployment Problem IPv4-only systems cannot handle IPv6 datagrams. A global simultaneous "flag day" upgrade is impossible.

Tunneling (Widely Used Transition Method)

- **Mechanism:** Encapsulate an IPv6 datagram inside an IPv4 datagram.
- **Process:**
  1. Node B sends an IPv6 datagram to node E.
  2. If an IPv4 network lies in between, B encapsulates the IPv6 packet inside an IPv4 header.
  3. Outer IPv4 header uses IPv4 addresses of the tunnel endpoints (B and E).
  4. IPv4 routers forward normally, unaware of inner IPv6 datagram.
  5. Tunnel endpoint E removes IPv4 header and processes the IPv6 packet.
- **Purpose:** Allows IPv6 connectivity through existing IPv4 infrastructure.

3.5.6 Key Terms and Definitions

- **Datagram:** Network-layer packet (IPv4 or IPv6).
- **Prefix (Network Portion):** High-order bits of an IP address, specified by the /x mask.
- **Subnet:** Devices sharing the same prefix and connected without a router.
- **CIDR:** Classless Interdomain Routing—current addressing system using variable-length prefixes.
- **DHCP:** Protocol for automatic assignment of IP addresses and configuration.
- **NAT:** Translates private addresses to a public address using ports.
- **Tunneling:** Encapsulation of one protocol inside another (e.g., IPv6 inside IPv4).
- **TTL / Hop Limit:** Prevents loops by decrementing at each hop.
- **IPv6:** Modern Internet Protocol using 128-bit addressing with simplified headers.

3.5.7 Core Relationships

- **IP Address vs. Interface:** IP addresses are assigned to interfaces, not devices.
- **IP Addressing vs. Fragmentation:** IPv4 allows fragmentation; IPv6 does not (sender must handle MTU).
- **DHCP vs. NAT:**
  - DHCP assigns local addresses.
  - NAT maps private addresses to a public one for external communication.

3.5.8 Key Insights / Takeaways

- IPv4 uses a 20-byte header with fields that require per-hop processing (TTL, checksum).
- IP addressing is hierarchical to enable scalable routing via CIDR.
- NAT conserves addresses but breaks the end-to-end principle.
- IPv6 provides:
  - vast address space (128 bits),
  - simplified 40-byte header,
  - no router fragmentation,
  - flow label support,
  - and relies on tunneling for transition from IPv4.

3.6 Template Example

- X
  1. Y

4 4.4 Generalized Forwarding and SDN

4.1 Concept Overview

- Traditional Model of Fowarding
  1. Router makes forwarding decision based on single field in packet header, dest IP address
  2. Router sole job to perform lookup on dest IP address (using longest prefix matching) and send packet to correct output port
- Generalized Forwarding
  1. Describes a "match-plus-action" paradigm where a packet switch (which could be a router or a switch) makes decision based on wide range of fields in packet header, spanning multiple protocol layers
  2. Match-plus-action abstraction
     (a) Instead of just matching dest IP, switch can match on any combination of fields, such as src IP, dest port, source MAC addresses, VLAN tag, etc.
     (b) Action:
        i. Instead of just forwarding, switch can perform variety of actions, such as

A. Forward packet to one or more output ports (e.g. unicast, multicast, or broadcast)
B. Load balance packet across mutliple output ports
C. Rewrite head values
D. Block/drop packet
E. Send packet to special server or controller for further processing (e.g. deep packet inspection)

3. Flow Table
   (a) Generalizes simple forwarding table into a more complex flow table, flow table is "program" for packet switch's data plane
4. Connection to SDN
   (a) Generalized forwarding model is data-plane foundation of Software-Defined Networking
   (b) Because "match" and "action" rules can be complex and interdepedent, they're not computed by switch itself
   (c) Instead computed, installed, and updated by a remote, logically centralized controller

4.2 OpenFlow Flow Table

- match plus action table is called flow table. Each entry in table has three key components

  1. Set of header field values (the "match"): Patterns used to match against incoming packet. Matching performed very fast, by TCAMs
  2. Set of counters (the "statistics"): Counters are updated as packets match rule. Crucial for network management and analystics. Coutners track number of packets matched by rule and time since rule was last updated
  3. Set of Actions (the "action"): Instructiosn to execute when a packet matches a rule. Can be a list of actions (e.g. "rewrite source port to 80, then forward to port 3")

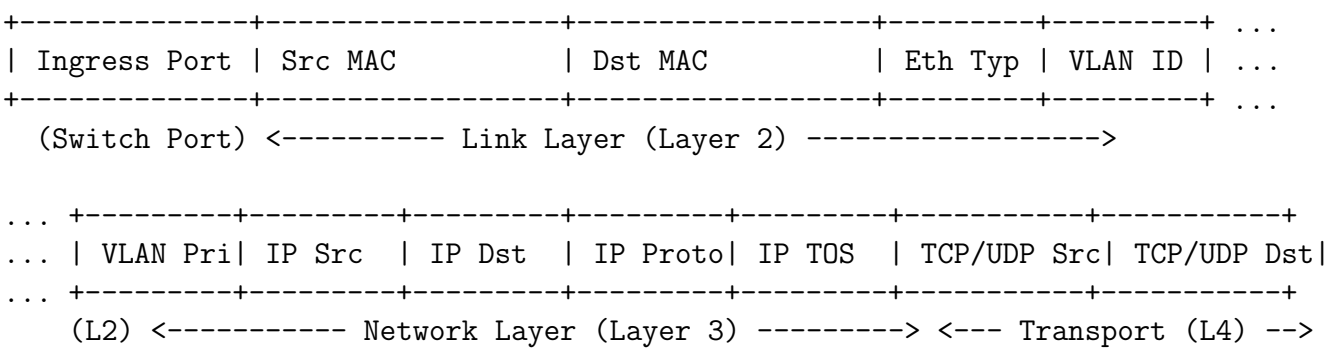
- If packet arrives and matches no entry in flow table, can either be dropped, or more commonly, forward to remote controller for a decision. Controller can decide what to do and potentially install a new flow rule for this packet's flow

4.3 4.4.1 Match

4.3.1 Concept Overview

- "Match" component of OpenFlow is what makes it powerful. Defines what parts of a packet switch can look at

4.3.2 Technical Mechanism



- OpenFlow 1.0 defines 11 header fields (plus ingress port) that can be matched
    1. Ingress Port: physical port on which packet arrived at swtich
    2. Link-layer (L2) fields:
        (a) Src and Dest MAC Address: allows switch to operate as a Layer 2 switch
        (b) Ethernet Type: identifies protocol in payload
        (c) VLAN fields: allows for handling of Virtual LANs
    3. Network-Layer (L3) Fields:
        (a) IP Src and Dest Address: allows switch to operate as a Layer 3 router
        (b) IP Protocol: Identifies transport protocol
        (c) IP TOS: allows matching on quality-of-service bits
    4. Transport-Layer (L4) Fields
        (a) TCP/UDP Src and Dst Port: Allows for fine-grained, application aware rules (e.g. trading port 80 traffic differently from port 25 traffic)

4.3.3 Practical and Intuitive View: Wildcards and Priorities

Made even more flexible by two additional features

- Wildcards
    1. Flow table entries can have wildcards for any field
    2. Example: a match entry with IP Dst = 128.119.*.* and TCP Dst Port = 80, would match all web traffic destined for any host on 128.119.0.0/16 subnet
- Priority
    1. Packet might match multiple rules
    2. Example: one rule might match IP Dst = 128.119.*.*, while a second, more specific rule matches IP Dst = 128.119.40.1
    3. Flow table includes a priority field for each entry, and switch must apply action correspeonding to highest-priority matching rule

4.4 4.4.2 Action

4.4.1 Concept Overview

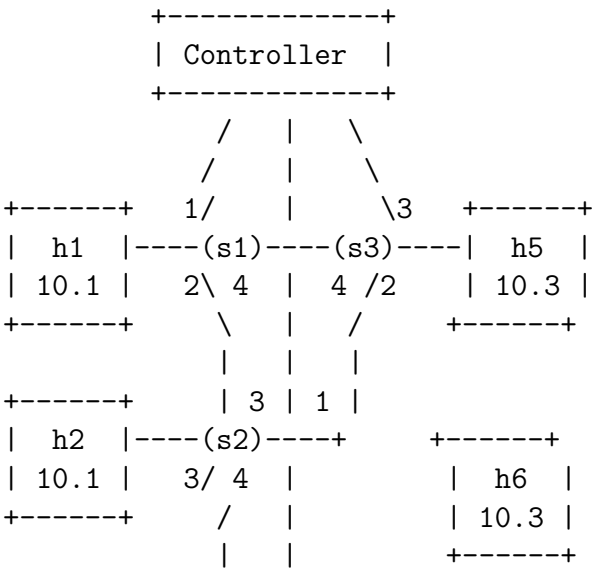
- Action is a list of zero or more operations that're applied to matched packet in sequence.
- Once a packet has been "matched" to a high-priority rule, switch executes "action" associated with rule

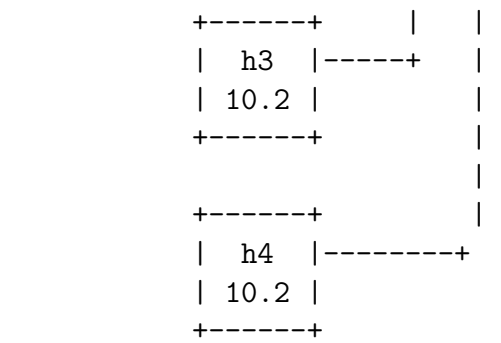
4.4.2 Technical Mechanism: Key Actions

Most important actions defined in OpenFlow

- Forwarding
  1. Forward to specific physical output port
  2. Broadcast: foward to all output ports (except ingress port)
  3. Multicast: forward to a select set of output ports
  4. Forward to controller: encapsulate packet and send it up to SDN controller for decision. How control plane learns about new, unknown flows
- Dropping
  1. Rule has no specified action, so packet is dropped. Basic mechanism for a firewall
- Modify-Field
  1. Switch can rewrite values in any of 10 header fields
  2. Basic mechanism for a NAT or Layer 3 router

4.5 4.4.3 OpenFlow Examples of Match-plus-action in Action





This section illustrates how combining matches and actions implements core network functionality using the sample network in Figure 4.30 (hosts $h1$ through $h6$ connected via switches $s1$, $s2$, and $s3$, under the control of a central controller).

4.5.1 Example 1: Simple Forwarding (Policy-Based Routing)

- Goal: Forward packets from subnet $10.3.*.*$ (hosts $h5, h6$) to subnet $10.2.*.*$ (hosts $h3, h4$) along the policy-selected path $s3 \rightarrow s1 \rightarrow s2$.
- Resulting switch behavior:
  1. **s3**: Match packets with IP Src = $10.3.*.*$ and IP Dst = $10.2.*.*$; action: *Forward(3)*.
  2. **s1**: Match packets arriving on port 1 with IP Src = $10.3.*.*$ and IP Dst = $10.2.*.*$; action: *Forward(4)*.
  3. **s2**:
     - If IP Dst = 10.2.0.3, action: *Forward(3)* (toward $h3$).
     - If IP Dst = 10.2.0.4, action: *Forward(4)* (toward $h4$).
- Insight: Matching on both ingress port and multiple IP fields enables forwarding behavior more specific than destination-based IP routing.

4.5.2 Example 2: Load Balancing (Source-Specific Forwarding)

- Goal: Load-balance traffic from subnet $10.2.*.*$ (hosts $h3, h4$) destined to $10.1.*.*$ by splitting paths:
  - Traffic from $h3$ uses path $s2 \rightarrow s1$.
  - Traffic from $h4$ uses path $s2 \rightarrow s3 \rightarrow s1$.
- Resulting switch behavior at **s2**:
  1. Match: Ingress Port = 3 (from $h3$), IP Dst = $10.1.*.*$; action: *Forward(2)* (toward $s1$).
  2. Match: Ingress Port = 4 (from $h4$), IP Dst = $10.1.*.*$; action: *Forward(1)* (toward $s3$).

- Insight: This demonstrates source-specific routing, impossible with traditional destination-only forwarding.

4.5.3 Example 3: Firewalling (Policy Enforcement)

- Goal: At switch $s2$, only allow packets originating from subnet $10.3.*.*$ (hosts $h5, h6$) to reach $h3$ or $h4$.
- Resulting switch behavior at **s2**:
  1. Match: IP Src = $10.3.*.*$, IP Dst = 10.2.0.3; action: *Forward(3)*.
  2. Match: IP Src = $10.3.*.*$, IP Dst = 10.2.0.4; action: *Forward(4)*.
  3. No other rules: default action = *Drop*.
- Insight: Switch $s2$ effectively acts as a firewall, blocking all traffic from $h1, h2$ destined for $h3, h4$.

4.5.4 Connection to Programmable Hardware

- OpenFlow provides a fixed match-plus-action abstraction.
- Next-generation systems such as **P4** allow defining new header formats and custom match-action logic, enabling protocol-independent programmability of the data plane.

4.5.5 Key Terms and Definitions

- **Generalized Forwarding**: Forwarding based on matching multiple header fields across layers, not only destination IP.
- **Flow Table**: Data structure containing match fields, counters, and actions.
- **OpenFlow**: Standard API enabling centralized controller-managed flow tables.
- **Modify-Field**: Action modifying packet headers (e.g., VLAN, IP, port).
- **P4**: Language for protocol-independent programmable data-plane processing.

4.5.6 Core Relationships

- SDN uses generalized forwarding to implement controller-driven network-wide behavior.
- Flow tables combine cross-layer match fields (port, MAC, IP, transport).
- Forward/Drop/Modify actions enable routing, firewalling, NAT, QoS, and load balancing in one device.

4.5.7 Key Insights / Takeaways

- Generalized forwarding enables rich, policy-driven behavior far beyond traditional IP routing.
- SDN relies on centralized controllers to ensure coherent network-wide configurations.
- Matching on source fields enables source-specific paths, load balancing, and security policies.

4.5.8 4.4 Section-Wide Summary

Key Terms

- **Generalized Forwarding**: A "match-plus-action" paradigm where forwarding decisions are based on multiple fields across multiple protocol layers (L2, L3, L4).
- **Match-Plus-Action**: The core abstraction of SDN data planes. First match on header fields, then apply an action (forward, drop, modify, send to controller).
- **Flow Table**: The data structure inside an SDN switch holding the match-plus-action rules.
- **SDN (Software-Defined Networking)**: An architecture that separates the data plane (switch hardware) from the control plane (remote controller software).
- **OpenFlow**: The standard protocol defining the match-plus-action abstraction and how an SDN controller programs a switch.
- **Match Fields**: The 11+ OpenFlow header fields (MAC addresses, IP addresses, TCP/UDP ports, etc.) used for matching.
- **Action Types**: The operations a switch can perform (Forward, Drop, Modify-Field, Send-to-Controller).
- **P4**: A domain-specific language for protocol-independent definition of packet-processing logic.

Core Relationships

- Generalized Forwarding is the enabling data-plane mechanism for SDN.
- OpenFlow implements the generalized forwarding abstraction, letting the SDN controller program flow tables in switches.
- The *match* component allows a single device to function simultaneously as a switch (L2), router (L3), firewall (L3/L4), and NAT (L3/L4).
- The *action* component (forward, drop, rewrite) makes the device truly programmable and policy-driven.

Key Insights / Takeaways

- "Match-plus-action" is the unifying abstraction that merges routing, switching, firewalling, and NAT into one programmable device.
- SDN breaks the traditional monolithic router by separating the "brain" (control plane) from the "brawn" (data plane), placing the brain in a logically centralized controller.
- The abstraction crosses layers: By matching on L2, L3, and L4 fields, SDN can implement holistic network-wide logic that traditional layered architectures could not support.
- SDN makes the network programmable: network-wide behaviors (load balancing, access control, traffic engineering) are implemented via controller software that automatically installs the appropriate low-level match-plus-action rules throughout the network.

5 4.5 Middleboxes

5.1 Concept Overview: Beyond the Router

- Middlebox defined as "any intermediary box performing functions apart from normal, standard functions of an IP router on data path between source host and destination host"
- Sit on forwarding path but modify, inspect, or filter packets based on policies often related to security, performance, op application-specific reqs
- Often violate end-to-end principle by processing information up to application layer (layer 7)or modifying network layer (layer 3) headers, thereby breaking strict separation of layers

5.2 A Taxonomy of Middlebox Services

Functions categorized into three categories

- NAT Translation
    1. NAT boxes are middleboxes that perform private network addressing
    2. Rewrite datagrams header's source/dest IP addresses and, transport-layers src/dest port numbers
- Security Services
    1. Firewalls
        (a) Most common security middlebox
        (b) Block traffic based on wide range of header-field values (IP addresses, port numbers, protocol type)

2. Instrusion Detection/Prevention System (IDS/IPS)
   (a) Devices perform "deep packet inspection", examining packets payload
   (b) Can redirect packets for further analysis, detect known attack patterns ("signatures"), and filter malicious packets
3. Application-Level Filters
   (a) Email gateways are prime example
   (b) Set on email delivery path and filter messags based on content, blocking spam, phishing attempts, and security threats

- Performance Enhancement

1. Load Balancers: Devices intercept incoming service requests and distribute them across a farm of backend servers to balance computational load
2. Content Caches: Devices store copies of popular content close to users to reduce latency and save network bandwidth
3. Traffic Compressors: Compress data streams on the fly to save bandwidth, especially over expensive or slow links

5.3 The Problem with Middleboxes: Cost and Complexity

- Capital Cost: each new function (firewall, NAT, load balancer) has required its own specialized, often proprietary, hardware "box"
- Operational Cost: each box runs its own separate software stack and requires separate management, configuration, and upgrades. Admin needs special skills to manage such

1. Y

5.4 The Solution: Network Function Virtualization (NFV)

- NFV's goal is to "unbundle" function from dedicated hardware
- Mechanism

1. Instead of buying firewall appliance, network operator buys commodity hardware (standard servers, switches, and storage)
2. Run middlebox functions as software on top of generic hardware, using virtual machines or containers

- Conenction to SDN

1. NFV is logical extension of SDN
2. SDN unbundles router, separating control-plane software from data-plan hardware
3. NFV unbundles all other network functions (middleboxes) in same way

5.5 Architectural Principles of the Internet (A Sidebar)

- The "Architectural Abomination" View
  1. Original Sin: Middleboxes violate clean separation between network layer and layers above/below it
  2. Layer-Breakers
     (a) Original Internet had a "simple" network core (routers just forward IP packets) and "smart" edges (hosts run transport and application logic)
     (b) NAT box is a router (L3) that peeks into and rewrites port numbers (L4)
     (c) Firewall is a router (L3) that blocks packets based on TCP flags (L4) or even HTTP URLs (L7)
     (d) Email gateway (L3) filters packets based on application-layer content (email body)
- The "Pragmatic" View
  1. Counter argument is middleboxes "exist for important and permanent reasons"
  2. Fill critical needs (security, address translation) that original architecture failed to provide
  3. Argument from this camp is that we will ahve more, not fewer, middleboxes in future, so must learn how to build and manage them properly (e.g. using NFV and SDN)
- The IP Hourglass (The "Narrow Waist")
  1. Concept: Internet protocol stack has narrow waist at network layer
  2. Above Waist: Innumerable application-layer protocols and transport protocols
  3. Below Waist: Innumerable link-layer technologies
  4. Wait: One single, universisal protocol: IP

5.6 4.5 Section-Wide Summary

Key Terms

- **Middlebox**: In-network device performing non-standard IP forwarding functions such as NAT, firewalling, load balancing.
- **Network Function Virtualization (NFV)**: Implementing middlebox functions in software on commodity hardware.
- **IP Hourglass**: The Internet architecture with a narrow waist at the IP layer.
- **End-to-End Argument**: Idea that intelligence should reside at end hosts, while the network core remains simple.

Core Relationships

- Middleboxes address practical needs (security, address scarcity) that original Internet architecture did not solve.
- NFV improves cost, flexibility, and deployment by separating middlebox software from hardware.
- Middleboxes challenge the IP Hourglass (by adding complexity to the core) and the End-to-End Argument.

Key Insights / Takeaways

- The IP core is no longer simple: it is filled with middleboxes essential for modern networking.
- SDN (match-plus-action) provides a way to program and manage middlebox-like behaviors centrally.
- The tension between architectural purity (End-to-End) and practical necessity (Middleboxes) will define the future of networking.