CS 181AG
Lecture 11

Intro to Packet Classification

Arthi Padmanabhan

Oct 5, 2022

Reading: Named Data Networking

Questions from Monday?

Big Picture: Router Functionality

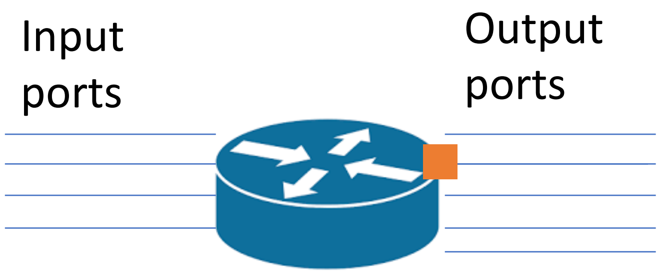
Switching:
move packet to correct port

Input ports

Output ports

Longest Matching Prefix to decide which output port

Big Picture: Router Functionality



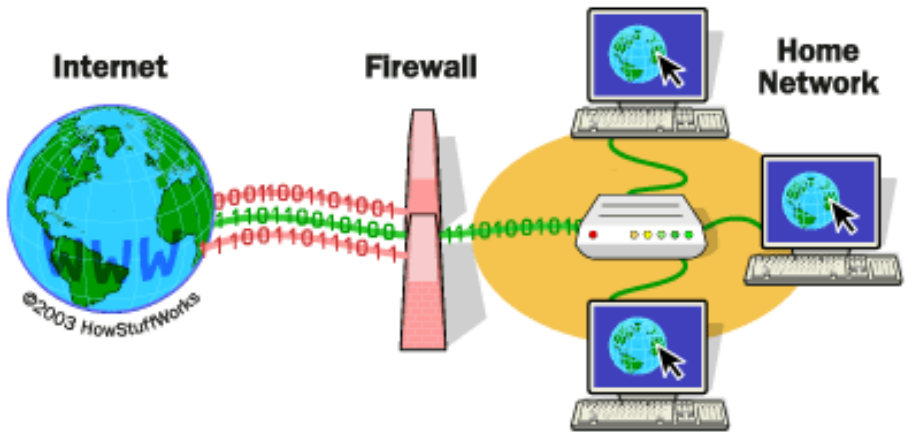
Schedule packets in output port queue

Big Picture: Router Functionality

- What if we need to look at parts of the packet beyond its destination, e.g., for a firewall?
  - Today’s topic!

Firewalls

- For routers than sit at edge of network, important job is to screen incoming packets for anything malicious
- Works based on pre-established rules
- ex/ look at packet header and remove packets that match certain threats
- ex/ let in packets only for certain applications OR packets that are responses to packets initiated from within network; otherwise drop

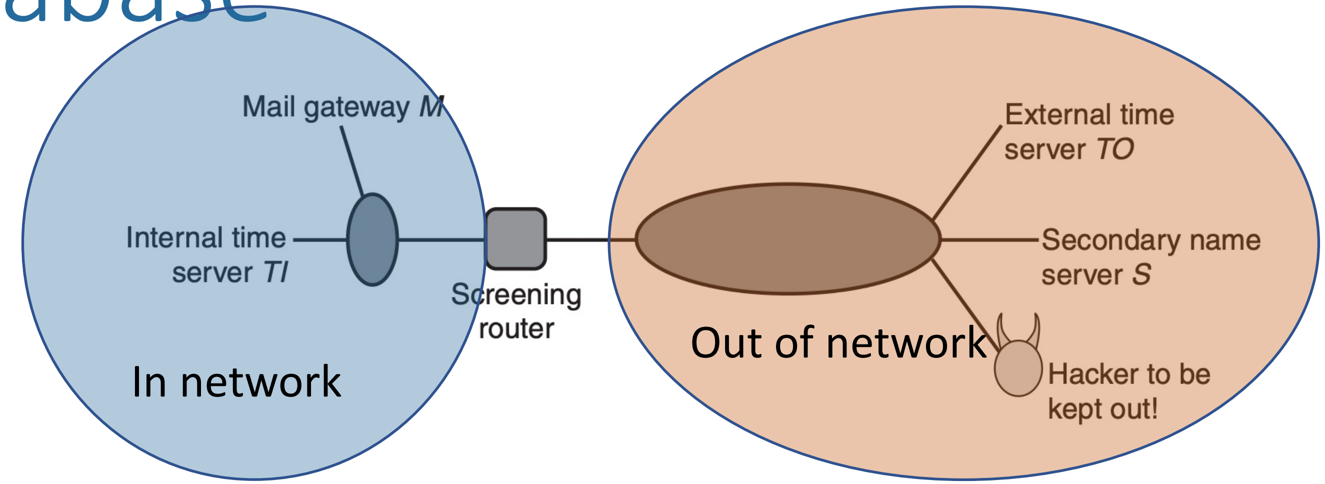


Fields for Screening

- Packets contain several parts to their headers that could be useful for screening, including:
  - Source/Dest IP – Who sent it? Who is it for?
  - Source/Dest Port – What type of traffic is it?
  - Flags – ex/ TCP ack: is it a response (acknowledgement) to a packet sent from within the network?

Firewall: Sample Database

- Assume:
  - Network is 1010*
  - For simplicity, IPs shown as 8 bit
  - M: 10101111; S: 01001010
  - T1: 10101010; T0: 11110000



Destination	Source	Destination Port	Source Port	Flags	Instruction
10101111	*	25	*	*	Allow
10101111	*	53	*	UDP	Allow
10101111	01001010	53	*	*	Allow
10101010	11110000	123 - 125	*	UDP	Allow
*	1010*	*	*	*	Allow
1010*	*	*	*	TCP ack	Allow
*	*	*	*	*	Block

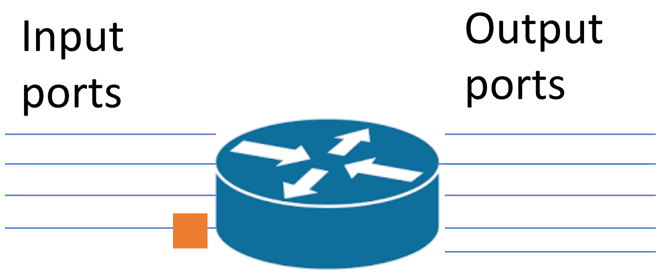
DiffServe

- Some organizations require that their traffic not be subverted by high traffic sent by other organizations (Quality of Service (QoS) guarantees), e.g., voice is more sensitive to slow packets
- DiffServe – reserve bandwidth between source and destination

Treating Different Traffic Differently?

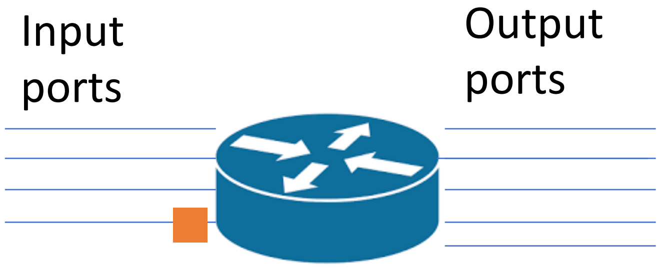
- Problem known as net neutrality: you will learn more about this in your reading

Big Picture: Router Functionality



Longest Matching
Prefix to decide which
output port

Big Picture: Router Functionality



Classify Packet, THEN Longest Matching Prefix to decide which output port

Packet Classification Problem

- Rules have costs – we want to find the lowest cost rule that matches
  - Makes sense to order rules by cost and find the first rule that matches
- Might have upwards of 1000 rules - similar to longest match prefix, this must be done *quickly*
- Depending on field, might be partial match, exact match, range match

Cost	Destination	Source	Destination Port	Source Port	Flags	Instruction
1	10101111	*	25	*	*	Allow
2	10101111	*	53	*	UDP	Allow
3	10101111	01001010	53	*	*	Allow
4	10101010	11110000	123 - 125	*	UDP	Allow
5	*	1010*	*	*	*	Allow
6	1010*	*	*	*	TCP ack	Allow
7	*	*	*	*	*	Block

Which rule is the least-cost match?

- Packet with header: (D, S, DP, SP, F)
  - (10101111, 01001010, 53, 64, -)
  - (10101111, 01001010, 53, 64, UDP)
  - (01010101, 10101011, 52, 65, TCP ack)
  - (10101111, 01001111, 53, 64, -)

Cost	Destination	Source	Destination Port	Source Port	Flags	Instruction
1	10101111	*	25	*	*	Allow
2	10101111	*	53	*	UDP	Allow
3	10101111	01001010	53	*	*	Allow
4	10101010	11110000	123 - 125	*	UDP	Allow
5	*	1010*	*	*	*	Allow
6	1010*	*	*	*	TCP ack	Allow
7	*	*	*	*	*	Block

Metrics

- Similar to prefix lookup:
  - Lookup Time
  - Memory
  - Insertion/Deletion time: many firewall rules do not change often, but we might have dynamic rules, e.g., when a packet leaves, create a rule for its response

Simple Solutions

- Linear:
  - Search through rules starting at least-cost
- Caching:
  - Low cache hit rates (short flows)
  - Could be combined with other methods

Two-Dimensional Schemes

- Let’s start by solving a simpler problem, having only 2 fields
- Notation: R1 -> apply Rule R1

Rule	Destination	Source
R1	D1 = 0*	S1 = 10*
R2	D2 = 0*	S2 = 01*
R3	D3 = 00*	S3 = 11*
R4	D4 = 00*	S4 = 1*
R5	D5 = 0*	S5 = 1*
R6	D6 = 10*	S6 = 1*
R7	D7 = *	S7 = 00*
R8	D8 = *	S8 = *

Two-Dimensional Schemes

- Why is this a “bad” set of rules?

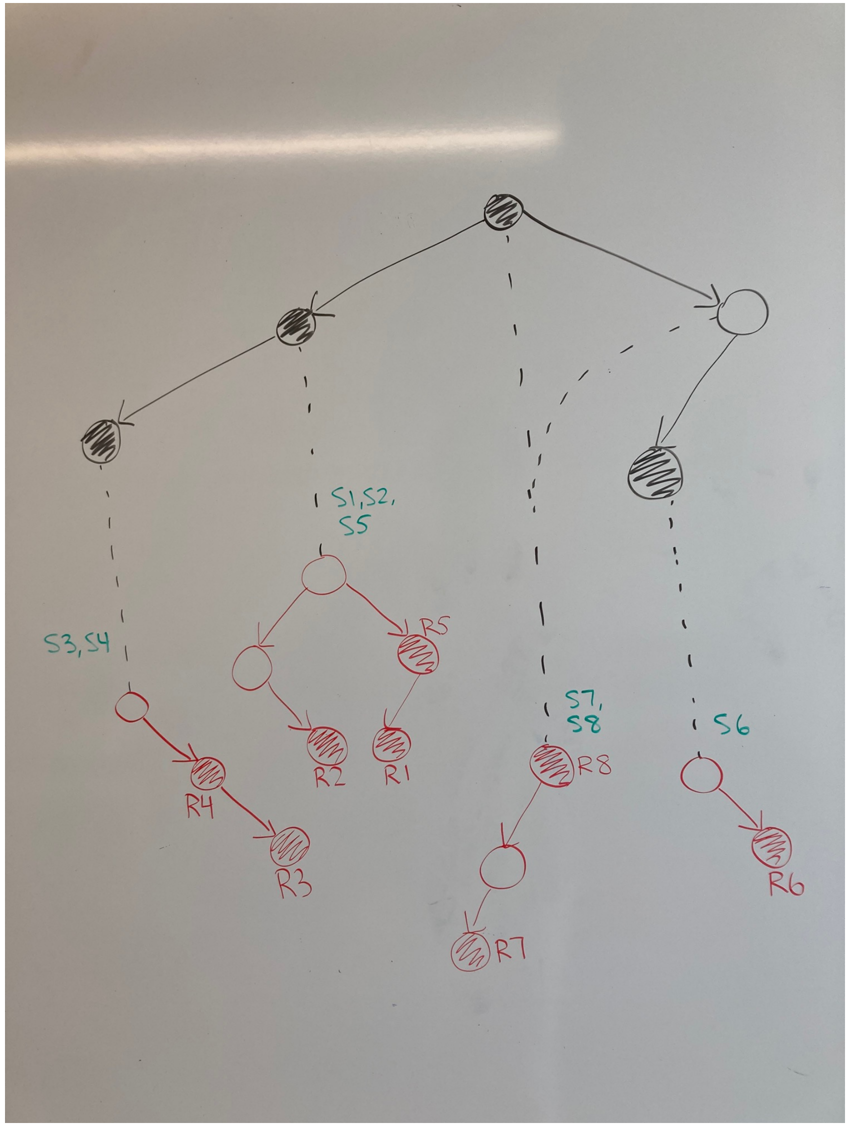
Rule	Destination	Source
R1	D1 = 0*	S1 = 10*
R2	D2 = 0*	S2 = 01*
R3	D3 = 0*	S3 = 1*
R4	D4 = 00*	S4 = 1*
R5	D5 = 00*	S5 = 11*
R6	D6 = 10*	S6 = 1*
R7	D7 = *	S7 = 00*
R8	D8 = *	S8 = *

Two-Dimensional Schemes

Rule	Destination	Source
R1	D1 = 0*	S1 = 10*
R2	D2 = 0*	S2 = 01*
R3	D3 = 00*	S3 = 11*
R4	D4 = 00*	S4 = 1*
R5	D5 = 0*	S5 = 1*
R6	D6 = 10*	S6 = 1*
R7	D7 = *	S7 = 00*
R8	D8 = *	S8 = *

Trie of Tries

- Construct a trie of destination prefixes
- Each valid destination prefix (D) points to a trie of source prefixes
    - The source trie contains source prefixes for all rules with a destination field exactly equal to D
- Problem?

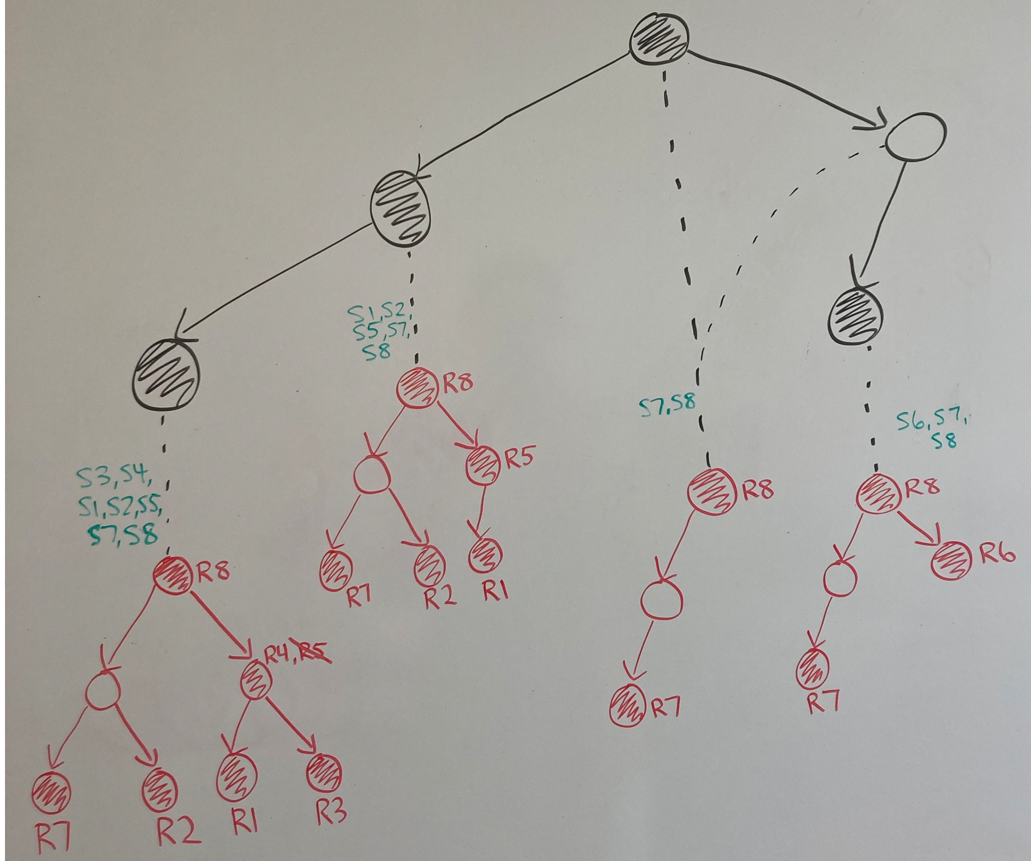

S1, S2, S5
S3, S4
R5
R2
R1
R4
R3
S7, S8
R8
R7
S6
R6

Trie of Tries: Backtracking

- Construct a trie of destination prefixes
- Each valid destination prefix (D) points to a trie of source prefixes
  - The source trie contains source prefixes for all rules with a destination field exactly equal to D
- Problem?
  - Best rule might not be in the trie corresponding to longest matching destination
  - For now, ignore the lookup time and find any solution that keeps trie as is
  - Solution: use backtracking to traverse each source trie corresponding to a destination that's a prefix of the longest matching destination

Trie of Tries: No Backtracking

- Construct a trie of destination prefixes
- Each valid destination prefix (D) points to a trie of source prefixes
  - The source trie contains source prefixes for **all** rules where D matches the destination field
  - That is, once we get to a source trie, we do not leave that trie
- Problem?


S1,S2,
S5,S7,
S8
R8
R5
R7
R2
R1
S7,S8
R8
R7
S6,S7,
S8
R8
R6
R7
S3,S4,
S1,S2,S5,
S7,S8
R8
R4, R5
R7
R2
R1
R3

Trie of Tries: No Backtracking

- Construct a trie of destination prefixes
- Each valid destination prefix (D) points to a trie of source prefixes
  - The source trie contains source prefixes for **all** rules where D matches the destination field
  - That is, once we get to a source trie, we do not leave that trie
- Problem?
  - Memory explosion