Wrapping up BGP and the IP header

Spring 2024
Sylvia Ratnasamy
CS168.io
Outline

- Wrapping up BGP
  - Context
  - Goals
  - Approach
    - Wrap up Gao-Rexford
  - Protocol design
  - Limitations
- Designing the IP header
Recall: computing paths on the AS graph
Recall: computing paths on the AS graph

- Nodes are Autonomous Systems (AS)
Recall: computing paths on the AS graph

- Nodes are **Autonomous Systems (AS)**
- Edges reflect physical connections & **biz relationships**
  - Customers pay providers
  - Peers don’t pay each other
Recall: computing paths on the AS graph

- Nodes are *Autonomous Systems (AS)*

- Edges reflect physical connections & *biz relationships*
  - Customers pay providers
  - Peers don’t pay each other

- Paths are selected based on *policy*
Recall: computing paths on the AS graph

- Nodes are **Autonomous Systems (AS)**
- Edges reflect physical connections & **biz relationships**
  - Customers pay providers
  - Peers don’t pay each other
- Paths are selected based on **policy**
- Policy reflects business goals (i.e., how money flows)
  - “Only carry traffic if you’re getting paid for it”
  - “Try and make/save money when sending traffic”
Recall: BGP

- Protocol that implements interdomain routing
Recall: BGP

- Protocol that implements interdomain routing
- Extends Distance-Vector
Recall: BGP

- Protocol that implements interdomain routing

- Extends Distance-Vector

- Basic idea
  - Destinations are prefixes
  - Each AS advertises its path to a prefix
  - Policy dictates which paths an AS selects ("import policy") and which paths it advertises ("export policy")
Recall: BGP

- Protocol that implements interdomain routing
- Extends Distance-Vector

Basic idea
- Destinations are prefixes
- Each AS advertises its path to a prefix
- Policy dictates which paths an AS selects ("import policy") and which paths it advertises ("export policy")

- Gao-Rexford rules tell us what import/export policies will achieve business goals
Gao-Rexford Rule: Import policy

- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider
Gao-Rexford Rule: Import policy

- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider.

- In practice, ASes use additional rules to break ties.
When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider.

In practice, ASes use additional rules to break ties.

Typical order of priority:
- First, make/save money (G-R rule)
- Then, maximize performance
- Then, minimize use of my network bandwidth
- ....

Gao-Rexford Rule: Import policy
Gao-Rexford Rule: Import policy

- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider.

- In practice, ASes use additional rules to break ties.

- Typical order of priority:
  - First, make/save money (G-R rule)
  - Then, maximize performance
  - Then, minimize use of my network bandwidth
  - ....
### Gao-Rexford Rules: Export policy

<table>
<thead>
<tr>
<th>Destination prefix advertised by...</th>
<th>Export route to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Everyone (providers, peers, other customers)</td>
</tr>
<tr>
<td>Peer</td>
<td>Customers</td>
</tr>
<tr>
<td>Provider</td>
<td>Customers</td>
</tr>
</tbody>
</table>

![Diagram showing network connectivity]
Gao-Rexford Rules: Implication

- Under **two assumptions** about the AS graph (coming up), if all ASes follow Gao-Rexford, we can guarantee:
  - **Reachability**: any two ASes can communicate
  - **Convergence**: all routers agree on paths
- The above hold in **steady state**
Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, etc.
Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, etc.
Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, etc.

\[ \text{Last change} \]
Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, etc.

\[ \text{Last change} \quad \text{convergence time} \quad \text{time} \]
Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, etc.

Diagram:
- Last change
- Routes settle or “converge” (i.e., will not change from here on)
- convergence time

Axis:
- time
Two assumptions
Two assumptions

#1 The graph of customer-provider relationships is acyclic

- Cannot have A → B → ... → C and then C → A (customer → provider)
- Means one can arrange providers in a hierarchy
- Note: OK if peering relationships are cyclic (A-B, B-C, C-A)
Gao-Rexford Rules: Implication

- Under two assumptions about the AS graph, if all ASes follow Gao-Rexford, we can guarantee:
  - **Reachability**: any two ASes can communicate
  - **Convergence**: all routers agree on paths
- The above hold in steady state
Gao-Rexford Rules: Implication

- Under two assumptions about the AS graph, if all ASes follow Gao-Rexford, we can guarantee:
  - **Reachability**: any two ASes can communicate
  - **Convergence**: all routers agree on paths

- The above hold in steady state

- The above are **not** guaranteed for general policies!
  - (You’ll see an example of this in section)
Recap
Recap

- Policy is implemented by choosing which routes we import and which ones we export
Recap

- Policy is implemented by choosing which routes we import and which ones we export

- Gao-Rexford rules tell us which routes to import/export in order to make/save money
Recap

- Policy is implemented by choosing which routes we import and which ones we export

- Gao-Rexford rules tell us which routes to import/export in order to make/save money

- Good stuff happens when you follow G-R rules
Outline

● Wrapping up BGP
  ● Context
  ● Goals
  ● Approach
    ● Wrap up Gao-Rexford
  ● Protocol design
  ● Limitations

● Designing the IP header
So far: our model of the AS graph

An AS advertises routes to its neighbor ASes
In reality...

Border routers

Interior routers
Many design questions....

- How do we ensure the routers “act as one”?
  - The role of border vs. interior routers?
  - Interaction between BGP and IGP?
  - How does BGP implement all this?
Who “speaks” BGP?

- Border routers
- Interior routers
Who “speaks” BGP?

Border routers at an Autonomous System
What does “speak BGP” mean?
What does “speak BGP” mean?

- Advertise routes as specified by the BGP protocol standard
What does “speak BGP” mean?

- Advertise routes as specified by the BGP protocol standard

- Specifies what messages BGP “speakers” exchange
  - message types and syntax
What does “speak BGP” mean?

- Advertise routes as specified by the BGP protocol standard

- Specifies what messages BGP “speakers” exchange
  - message types and syntax

- And how to process these messages
  - e.g., “when you receive a BGP update, do….”
Note: Some Border Routers Don’t Need BGP

- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS
Note: Some Border Routers Don’t Need BGP

- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS
Note: Some Border Routers Don’t Need BGP

- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS
Note: Some Border Routers Don’t Need BGP

- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS

Provider

Customer

Install default routes pointing to Provider

130.132.0.0/16
Note: Some Border Routers Don’t Need BGP

- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS

Provider

Install routes 130.132.0.0/16 pointing to Customer

Customer

Install default routes pointing to Provider

130.132.0.0/16
BGP “sessions”
BGP “sessions”
BGP “sessions”
BGP "sessions"

Only border routers exchange messages with routers in external domains (hence, *external BGP* or "eBGP")
Border router speaks BGP with routers in its own AS (hence, *internal* BGP, or “iBGP”)

BGP “sessions”
BGP “sessions”

Border router speaks BGP with routers in its own AS (hence, *internal* BGP, or “iBGP”)
Border router speaks BGP with routers in its own AS (hence, internal BGP, or “iBGP”)

BGP “sessions”

“iBGP session”
eBGP, iBGP, IGP
eBGP, iBGP, IGP

- **eBGP**: BGP sessions between border routers in different ASes
  - exchange routes to different destination prefixes
eBGP, iBGP, IGP

- **eBGP**: BGP sessions between border routers in **different** ASes
  - exchange routes to different destination prefixes

- **iBGP**: BGP sessions between border routers and other routers within the **same** AS
  - distribute externally learned routes internally
eBGP, iBGP, IGP

- **eBGP**: BGP sessions between border routers in different ASes
  - exchange routes to different destination prefixes

- **iBGP**: BGP sessions between border routers and other routers within the same AS
  - distribute externally learned routes internally

- **IGP**: “Interior Gateway Protocol” = Intradomain routing protocol
  - provide internal reachability
  - e.g., OSPF, RIP
Putting the pieces together
1. Provide internal reachability (IGP)
Putting the pieces together

1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
4. Travel shortest path to egress (IGP)
Putting the pieces together

1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
4. Travel shortest path to egress (IGP)
Putting the pieces together

1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
4. Travel shortest path to egress (IGP)
Putting the pieces together

1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
4. Travel shortest path to egress (IGP)
Putting the pieces together

1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
4. Travel shortest path to egress (IGP)
Putting the pieces together

1. Provide internal reachability (IGP)
2. Learn routes to external destinations (eBGP)
3. Distribute externally learned routes internally (iBGP)
4. Travel shortest path to egress (IGP)
Short Summary
Short Summary

- Every router in AS has two routing tables:
  - From IGP: next hop router to all *internal* destinations
  - From iBGP: egress router to all *external* destinations
Short Summary

- Every router in AS has two routing tables:
  - From IGP: next hop router to all *internal* destinations
  - From iBGP: egress router to all *external* destinations

- For internal addresses, just use IGP
  - Entry <internal destination, internal next hop>
Every router in AS has two routing tables:
- From IGP: next hop router to all *internal* destinations
- From iBGP: egress router to all *external* destinations

For internal addresses, just use IGP
- Entry <internal destination, internal next hop>
Short Summary

• Every router in AS has two routing tables:
  • From IGP: next hop router to all *internal* destinations
  • From iBGP: egress router to all *external* destinations

• For internal addresses, just use IGP
  • Entry <internal destination, internal next hop>

• For external locations: use iBGP to find egress
  • Use IGP to find next hop to egress router
Note: In reality, there are a few different ways to integrate inter- and intra-domain routing
Note: In reality, there are a few different ways to integrate inter- and intra-domain routing

- Our option: run iBGP between all routers in domain
  - Requires NxB iBGP connections. Could be a scaling issue.
  - This is what we will assume
Many design questions....
Many design questions....

- How do we ensure the routers in an AS “act as one”?
  - The role of border vs. interior routers?
  - Interaction between BGP and IGP
- How is all this implemented?
  - Route updates and attributes
BGP protocol message types

- Many different message types
  - Open
  - Keepalive
  - Notification
  - ...
  - Update
    - Inform neighbor of new routes
    - Inform neighbor of updates to old routes
    - “Withdraw” a route that’s now inactive
Route Updates
Route Updates

- Format `<IP prefix: route attributes>`
  - attributes describe properties of the route
Route Attributes

- General mechanism used to express properties about routes
  - Used in route selection/export decisions
Route Attributes

- General mechanism used to express properties about routes
  - Used in route selection/export decisions

- Some attributes are local to an AS
  - Not propagated in eBGP advertisements
Route Attributes

- General mechanism used to express properties about routes
  - Used in route selection/export decisions
- Some attributes are local to an AS
  - Not propagated in eBGP advertisements
- Others are propagated in eBGP route advertisements
Route Attributes

- General mechanism used to express properties about routes
  - Used in route selection/export decisions

- Some attributes are local to an AS
  - Not propagated in eBGP advertisements

- Others are propagated in eBGP route advertisements

- There are many standardized attributes in BGP
  - We will discuss four important ones
Attributes (1): **ASPATH**

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements
Attributes (1): **ASPATH**

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements
Attributes (1): AS PATH

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements
Attributes (1): **ASPATH**

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements
Attributes (1): ASPATH

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements
Attributes (1): **ASPATH**

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements

**IP prefix = 128.112.0.0/16**
**AS path = 88**

**Princeton, 128.112.0.0/16**
**AS 88**

**AS 7018 ATT**

**AS 25 Berkeley**
Attributes (1): ASPATH

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements
Attributes (1): ASPATH

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements

IP prefix = 128.112.0.0/16
AS path = 88

IP prefix = 128.112.0.0/16
AS path = 7018 88

Princeton, 128.112.0.0/16
AS 88

AS 7018
ATT

AS 25
Berkeley
Attributes (2): **LOCAL PREFERENCE**

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred
Attributes (2): LOCAL PREFERENCE

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred
Attributes (2): LOCAL PREFERENCE

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred

IP prefix = 128.112.0.0/16
AS path = 100 88
**Attributes (2): LOCAL PREFERENCE**

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred

**IP prefix = 128.112.0.0/16**
**AS path = 100 88**

**BGP table at AS 7018:**

<table>
<thead>
<tr>
<th>destination</th>
<th>AS PATH</th>
<th>LocPref</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.112.0.0/16</td>
<td>100, 88</td>
<td>3000</td>
</tr>
</tbody>
</table>
Attributes (2): LOCAL PREFERENCE

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred

BGP table at AS 7018:

<table>
<thead>
<tr>
<th>destination</th>
<th>ASPATH</th>
<th>LocPref</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.112.0.0/16</td>
<td>100, 88</td>
<td>3000</td>
</tr>
</tbody>
</table>

IP prefix = 128.112.0.0/16
AS path = 200 88

AS 100

AS 88

AS 200

AS 7018
Attributes (2): LOCAL PREFERENCE

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred

<table>
<thead>
<tr>
<th>destination</th>
<th>ASPATH</th>
<th>LocPref</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.112.0.0/16</td>
<td>100, 88</td>
<td>3000</td>
</tr>
<tr>
<td>12.112.0.0/16</td>
<td>200, 88</td>
<td>1000</td>
</tr>
</tbody>
</table>

BGP table at AS 7018:
In reality...

Princeton

AT&T

Verizon

Berkeley
In reality...
In reality...

Princeton

Berkeley

AT&T

Verizon
In reality...

Note: both routes follow the same AS path!
In reality...

Which route does Verizon prefer?

Note: both routes follow the same AS path!
Attributes (3) : MED

- MED = “Multi-Exit Discriminator”
- Used when ASes are interconnected via 2 or more links to specify how close a prefix is to the link it is announced on.
Attributes (3) : MED

Princeton

Verizon

AT&T

Berkeley
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
- AS receiving prefix (optionally!) uses MED to select link

Princeton

AT&T

Verizon

IP prefix = ...
AS path = ...
MED = 10

Berkeley

IP prefix = ...
AS path = ...
MED = 50
More reality...

Which route does AT&T prefer?
Attributes (4): IGP cost

- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. “hot potato” routing
Attributes (4): IGP cost

- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. “hot potato” routing
Attributes (4): IGP cost

- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. “hot potato” routing

![Diagram of network with distances and labels]
Attributes (4): IGP cost

- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. “hot potato” routing
Attributes (4): IGP cost

- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. “hot potato” routing

![Diagram showing network with labels AT&T and Verizon]
Attributes (4): IGP cost

- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. “hot potato” routing

![Diagram showing network connectivity between AT&T and Verizon with IGP costs labeled.]
Note: IGP may conflict with MED
Note: IGP may conflict with MED
IGP-MED conflicts pretty common
IGP-MED conflicts pretty common
IGP-MED conflicts pretty common
IGP-MED conflicts pretty common

![Map of the United States with points A and B and a path labeled D_{sf}]

A B
IGP-MED conflicts pretty common

Can lead to asymmetric paths!
Closing the loop...

Typical Selection Policy

- In decreasing order of priority
  - make/save money
  - maximize performance
  - minimize use of my network bandwidth
  - ...
  - ...
Closing the loop...

Typical Selection Policy

In decreasing order of priority

- make/save money: LOCAL PREF (cust > peer > provider)
- maximize performance: length of ASPATH
- minimize use of my network bandwidth: “hot potato”, MED
- ...
  ...
  ...
  ...
Using Attributes

- Rules for route selection in priority order

<table>
<thead>
<tr>
<th>Priority</th>
<th>Rule</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LOCAL PREF</td>
<td>Pick highest LOCAL PREF</td>
</tr>
<tr>
<td>2</td>
<td>ASPATH</td>
<td>Pick shortest ASPATH length</td>
</tr>
<tr>
<td>3</td>
<td>IGP path</td>
<td>Lowest IGP cost to next hop (egress router)</td>
</tr>
<tr>
<td>4</td>
<td>MED</td>
<td>MED preferred</td>
</tr>
<tr>
<td>5</td>
<td>Router ID</td>
<td>Smallest next-hop router’s IP address as tie-breaker</td>
</tr>
</tbody>
</table>
Questions?
Outline

- Context
- Goals
- Approach
- Detailed design
- Limitations
Issues with BGP
Issues with BGP

- Security

No guarantee that an AS owns the prefixes it advertises!
No guarantee that an AS will follow the path it advertises.
Issues with BGP

- Security
- Performance (non?)issues
Issues with BGP

- Security
- Performance (non?)issues
- Prone to misconfiguration
Issues with BGP

- Security
- Performance (non?)issues
- Prone to misconfiguration
- Reachability and Convergence
Issues with BGP

- Security
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises
- Performance (non?)issues
Issues with BGP

- **Security**
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises

- **Performance (non?)issues**
  - Policy-based paths not necessarily shortest/least-cost
  - AS path length can be misleading

- **Prone to misconfiguration**
Issues with BGP

- **Security**
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises

- **Performance (non?)issues**
  - Policy-based paths not necessarily shortest/least-cost
  - AS path length can be misleading

- **Prone to misconfiguration**
  - Many attributes; configuration often manual and ad-hoc
  - BGP misconfigurations a major source of Internet outages!

- **Reachability and Convergence**
Issues with BGP

- Security
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises

- Performance (non?) issues
  - Policy-based paths not necessarily shortest/least-cost
  - AS path length can be misleading

- Prone to misconfiguration
  - Many attributes; configuration often manual and ad-hoc
  - BGP misconfigurations a major source of Internet outages!

- Reachability and Convergence
  - Not guaranteed if Gao-Rexford doesn’t hold
  - Example of policy oscillations in discussion section
Questions?
Taking Stock: We’ve done...

- An end-to-end overview of the Internet arch.
- How L3 works
  - IP addressing and routers
  - Intra-domain routing
  - Inter-domain routing
- Last topic: the IP header
  - At which point, you’ll know how L3 works!
Let’s design the IP header

- **Syntax**: format of an IP packet
  - Nontrivial part: header
  - Rest is opaque payload

- **Semantics**: meaning of IP header fields
  - How they’re processed
Recall: Layering

- **L4 (transport)**: Add Transport header (e.g., TCP)
- **L3 (network)**: Add Network header (e.g., IP)
- **L1+L2**: Add L2 header (e.g., Ethernet)
- **L7 (app)**: Take data, add app header (e.g., HTTP)
Recall: Layering

- L1+L2: Add L2 header (e.g., Ethernet)
- L3 (network): Add Network header (e.g., IP)
- L4 (transport): Add Transport header (e.g., TCP)
- L7 (app): Take data, add app header (e.g., HTTP)

Host A to Host B
Recall: Hosts vs. Routers

- **IP**
- **TCP**
- **HTTP**
- **Ethernet interface**
- **SONET interface**

**HTTP messages**

**TCP bytestreams**

**IP packets**

**Ethernet frames**

**SONET frames**
Recall: Hosts vs. Routers

HTTP messages

TCP bytestreams

IP packet

Ethernet frames

SONET frames

Ethernet interface

SONET interface
Designing the IP header

- Think of the IP header as an interface
  - between the source and network (routers)
  - between the source and destination endhosts
Designing the IP header

- Think of the IP header as an interface
  - between the source and network (routers)
  - between the source and destination endhosts

- Designing an interface
  - what task(s) are we trying to accomplish?
  - what information is needed to do it?
Designing the IP header

- Think of the IP header as an interface
  - between the source and network (routers)
  - between the source and destination endhosts

- Designing an interface
  - what task(s) are we trying to accomplish?
  - what information is needed to do it?

- Header reflects information needed for basic tasks
What are these tasks?
(at a router, at the destination host)
What are these tasks? (at a router, at the destination host)

- Parse packet (*router, dst host*)
What are these tasks? (at a router, at the destination host)

- Parse packet *(router, dst host)*
- Forward packet to the L3 destination *(router)*
What are these tasks? (at a router, at the destination host)

- Parse packet *(router, dst host)*
- Forward packet to the L3 destination *(router)*
- Tell destination what to do next *(dst host)*
What are these tasks?
(at a router, at the destination host)

- Parse packet (router, dst host)
- Forward packet to the L3 destination (router)
- Tell destination what to do next (dst host)

Next: what information do we need?
What are these tasks? (at a router, at the destination host)

- Parse packet *(router, dst host)*
- Forward packet to the L3 destination *(router)*
- Tell destination what to do next *(dst host)*
- Get responses back to the source *(dst host, router)*

Next: what information do we need?
What are these tasks? (at a router, at the destination host)

- Parse packet \((router, dst \ host)\)
- Forward packet to the L3 destination \((router)\)
- Tell destination what to do next \((dst \ host)\)
- Get responses back to the source \((dst \ host, router)\)
- Deal with problems along the way \((router, dst \ host)\)

Next: what information do we need?
What are these tasks? (at a router, at the destination host)

- Parse packet *(router, dst host)*
- Forward packet to the L3 destination *(router)*
- Tell destination what to do next *(dst host)*
- Get responses back to the source *(dst host, router)*

- Deal with problems along the way *(router, dst host)*
- Specify any special handling *(router, dst host)*

Next: what information do we need?
Parse Packet Correctly
Parse Packet Correctly

- What version of IP?
Parse Packet Correctly

- What version of IP?
- Where does header end?
Parse Packet Correctly

- What version of IP?
- Where does header end?
- Where does packet end?
Deliver packet to the L3 destination
Deliver packet to the L3 destination

- Provide destination address (duh!)
Tell the destination how to handle packet
Tell the destination how to handle packet
Tell the destination how to handle packet.
Tell the destination how to handle packet

- Indicate which protocol should handle packet next
Tell the destination how to handle packet

- Indicate which protocol should handle packet next
- **Protocol** field: identifies the higher-level protocol
  - Important for *de-multiplexing* at receiving host
Tell the destination how to handle packet

- Protocol field that identifies the L4 protocol for this packet
- Common examples
  - “6” for the Transmission Control Protocol (TCP)
  - “17” for the User Datagram Protocol (UDP)
Get responses back to the source
Get responses back to the source

- Source IP address
Where are we ...

- Parse packet → *version, header length, packet length*
- Forward packet to the L3 dst → *destination address*
- Tell destination what to do next → *protocol field*
- Get responses back to the source → *source address*

- Deal with problems along the way
- Specify any special handling
What problems?
What problems?

- Loops
What problems?

- Loops
- Corruption
What problems?

- Loops
- Corruption
- Packet too large (> MTU)
Preventing Loops
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity

Means header must include source IP address
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity

Means header must include source IP address
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity

Means header must include source IP address
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity

Means header must include source IP address
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity

Means header must include source IP address
Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity

- Time-to-Live (TTL) field
  - decremented at each hop, packet discarded if reaches 0
  - ...and “time exceeded” message is sent to the source

Means header must include source IP address.
Header Corruption
Header Corruption

- Checksum
Header Corruption

- **Checksum**
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header
Header Corruption

- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header

- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information
Header Corruption

- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header

- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information

- Checksum updated at every router
Header Corruption

- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header

- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information

- Checksum updated at every router
  - Why?
Header Corruption

- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header

- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information

- Checksum updated at every router
  - Why?
  - Why include TTL?
Header Corruption

- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header

- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information

- Checksum updated at every router
  - Why?
  - Why include TTL?
  - Why only header?
Every link has a “Maximum Transmission Unit” (MTU)
- largest number of bits it can carry as one unit
**Fragmentation**

- Every link has a “Maximum Transmission Unit” (MTU)
  - largest number of bits it can carry as one unit

- A router can split a packet into multiple “fragments” if the packet size exceeds the link’s MTU

```
hdr 3980
```

```
hdr 1480
```

```
hdr 1200
```

```
hdr 1300
```
Fragmentation

- Every link has a “Maximum Transmission Unit” (MTU)
  - largest number of bits it can carry as one unit
- A router can split a packet into multiple “fragments” if the packet size exceeds the link’s MTU
- Must reassemble to recover original packet
Fragmentation

- Every link has a “Maximum Transmission Unit” (MTU)
  - largest number of bits it can carry as one unit

- A router can split a packet into multiple “fragments” if the packet size exceeds the link’s MTU

- Must reassemble to recover original packet

Details of fragmentation will be covered in section
Where are we ...

- Parse packet → version, header length, packet length
- Forward packet to the L3 dst → destination address
- Tell destination what to do next → protocol field
- Get responses back to the source → source address
- Deal with problems along the way
  → TTL, source address, checksum, frag. fields (TBD)
- Specify any special handling
What forms of special treatment?
What forms of special treatment?

- Don’t treat all packets the same (“Type of Service”)
  - Idea: treat packets based on app/customer needs
What forms of special treatment?

- Don’t treat all packets the same (“Type of Service”)
  - Idea: treat packets based on app/customer needs

- “Options”
  - Request advanced functionality for this packet
“Type of Service” (ToS)
“Type of Service” (ToS)

- Originally: multiple bits used to request different forms of packet delivery
  - Based on priority, delay, throughput, reliability, or cost
  - Frequently redefined, never fully deployed
  - Only notion of priorities remained
“Type of Service” (ToS)

- Originally: multiple bits used to request different forms of packet delivery
  - Based on priority, delay, throughput, reliability, or cost
  - Frequently redefined, never fully deployed
  - Only notion of priorities remained

- Today:
  - Differentiated Services Code Point (DSCP): traffic “classes”
  - Explicit Congestion Notification (ECN): a later lecture
Options
Options

- Optional directives to the network

Examples

- Record Route, Source Route, Timestamp, ...
Options

● Optional directives to the network

● Examples
  ● Record Route, Source Route, Timestamp, ...

● More complex implementation
  ● Leads to variable length headers
  ● Often leads to higher processing overheads
Where are we ...

- Parse packet → *version, header length, packet length*
- Forward packet to the L3 dst → *destination address*
- Tell destination what to do next → *protocol field*
- Get responses back to the source → *source address*
- Deal with problems along the way
  → *TTL, source address, checksum, frag. fields (TBD)*
- Specify any special handling → *ToS, options*
IP Packet Structure

- 4-bit Version
- 4-bit Header Length
- 8-bit Type of Service
- 16-bit Total Length (Bytes)
- 16-bit Identification
- 3-bit Flags
- 13-bit Fragment Offset
- 8-bit Time to Live (TTL)
- 8-bit Protocol
- 16-bit Header Checksum
- 32-bit Source IP Address
- 32-bit Destination IP Address
- Options (if any)
- Payload

32 bits
IP Packet Structure

- 4-bit Version
- 4-bit Header Length
- 16-bit Total Length (Bytes)
- Options (if any)
- Payload

32 bits
IP Packet Structure

- 4-bit Version
- 4-bit Header Length
- 16-bit Total Length (Bytes)
- 32-bit Destination IP Address
- Payload

Options (if any)

Payload: 32 bits
IP Packet Structure

- 4-bit Version
- 4-bit Header Length
- 16-bit Total Length (Bytes)
- 8-bit Protocol
- 32-bit Destination IP Address
- Options (if any)
- Payload

32 bits
## IP Packet Structure

<table>
<thead>
<tr>
<th>4-bit Version</th>
<th>4-bit Header Length</th>
<th>16-bit Total Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8-bit Protocol</td>
<td>32-bit Source IP Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32-bit Destination IP Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Payload</td>
</tr>
</tbody>
</table>

32 bits
IP Packet Structure

- **4-bit Version**
- **4-bit Header Length**
- **16-bit Total Length (Bytes)**
- **8-bit Time to Live (TTL)**
- **8-bit Protocol**
- **16-bit Header Checksum**
- **32-bit Source IP Address**
- **32-bit Destination IP Address**
- **Payload**

32 bits
### IP Packet Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4-bit</td>
</tr>
<tr>
<td>Header Length</td>
<td>4-bit</td>
</tr>
<tr>
<td>Total Length (Bytes)</td>
<td>16-bit</td>
</tr>
<tr>
<td>Identification</td>
<td>16-bit</td>
</tr>
<tr>
<td>Flags</td>
<td>3-bit</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>13-bit</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>8-bit</td>
</tr>
<tr>
<td>Protocol</td>
<td>8-bit</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16-bit</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>32-bit</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>32-bit</td>
</tr>
<tr>
<td>Payload</td>
<td>32-bit</td>
</tr>
</tbody>
</table>

---

**Example:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Version</td>
<td>4-bit</td>
</tr>
<tr>
<td>IP Header Length</td>
<td>4-bit</td>
</tr>
<tr>
<td>IP Total Length (Bytes)</td>
<td>16-bit</td>
</tr>
<tr>
<td>IP Identification</td>
<td>16-bit</td>
</tr>
<tr>
<td>IP Flags</td>
<td>3-bit</td>
</tr>
<tr>
<td>IP Fragment Offset</td>
<td>13-bit</td>
</tr>
<tr>
<td>IP TTL</td>
<td>8-bit</td>
</tr>
<tr>
<td>IP Protocol</td>
<td>8-bit</td>
</tr>
<tr>
<td>IP Header Checksum</td>
<td>16-bit</td>
</tr>
<tr>
<td>IP Source Address</td>
<td>32-bit</td>
</tr>
<tr>
<td>IP Destination Address</td>
<td>32-bit</td>
</tr>
<tr>
<td>IP Payload</td>
<td>32-bit</td>
</tr>
</tbody>
</table>
## IP Packet Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4-bit</td>
</tr>
<tr>
<td>Header Length</td>
<td>4-bit</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8-bit</td>
</tr>
<tr>
<td>Total Length (Bytes)</td>
<td>16-bit</td>
</tr>
<tr>
<td>Identification</td>
<td>16-bit</td>
</tr>
<tr>
<td>Flags</td>
<td>3-bit</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>13-bit</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>8-bit</td>
</tr>
<tr>
<td>Protocol</td>
<td>8-bit</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16-bit</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>32-bit</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>32-bit</td>
</tr>
<tr>
<td>Payload</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

**Diagram:**

- The diagram shows the structure of an IP packet, with each field and its corresponding length indicated.
- The payload consists of 32 bits.
# IP Packet Structure

<table>
<thead>
<tr>
<th>4-bit Version</th>
<th>4-bit Header Length</th>
<th>8-bit Type of Service</th>
<th>16-bit Total Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>16-bit Identification</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>3-bit Flags</strong></td>
<td>13-bit Fragment Offset</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>8-bit Protocol</td>
<td>16-bit Header Checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 bits
Two remaining topics (next time)

- IPv4 $\rightarrow$ IPv6
- Security implications of the IP header
Questions?