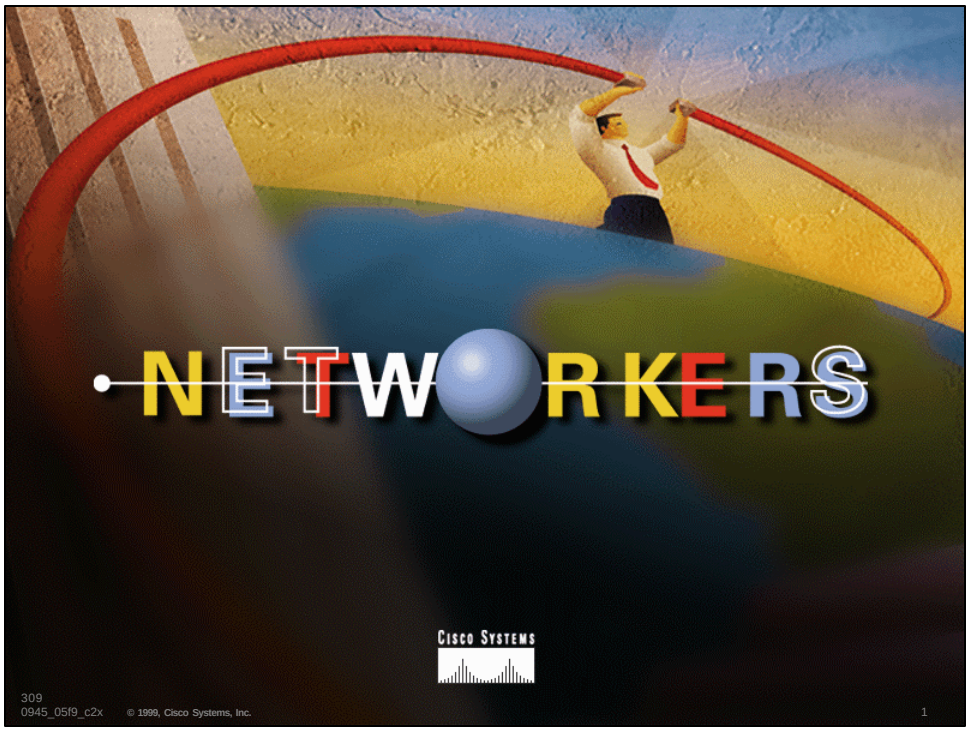


Overview

- Using BGP Attributes
- Deploying IBGP
- Deploying EBGP
- Focus on stability, scalability, and example configs

Bottom Line?

- **Implementation of routing policies that are:**
- ***Scalable***
- ***Stable***
- ***Simple (we hope!)***

More Details...

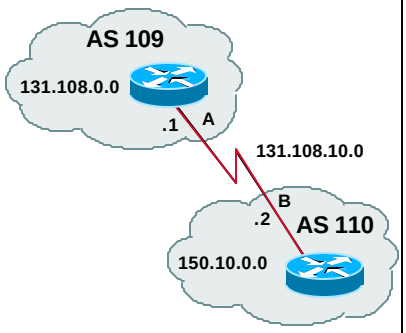
- **You need to scale your IGP**
- **You're a multi-homed ISP customer**
- **You need to transit full Internet routes**
- **You need to implement route policy, or augment QoS policy**

External BGP

Between BGP speakers
in different AS
Usually directly connected
Sets next-hop to self

Router B
router bgp 110
neighbor 131.108.10.1 remote-as 109

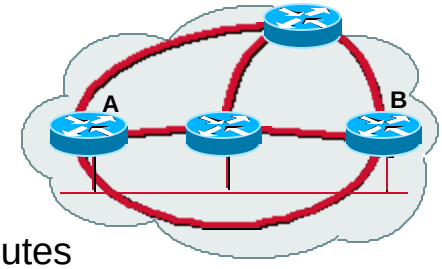
Router A
router bgp 109
neighbor 131.108.10.2 remote-as 110



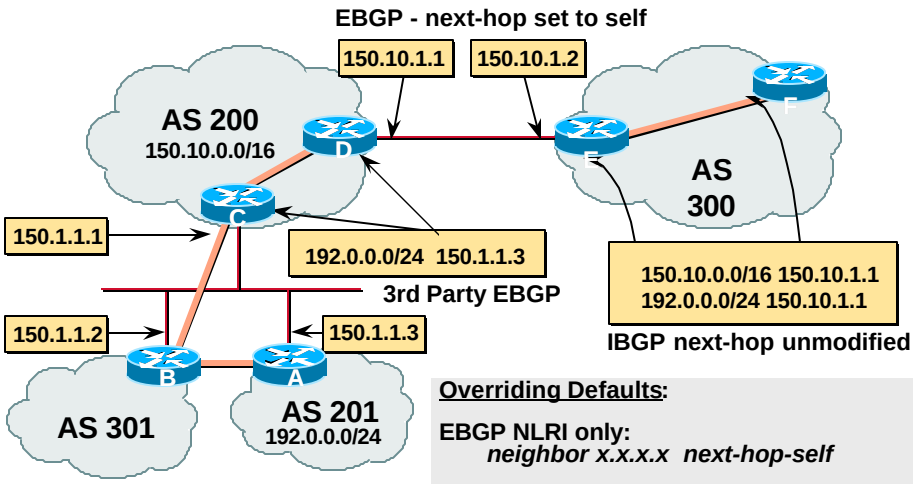
Internal BGP

Neighbor in same AS
next-hop unchanged
May be several hops away
Don't forward other IBGP routes

Router B
router bgp 109
neighbor 131.108.30.2 remote-as 109
Router A
router bgp 109
neighbor 131.108.20.1 remote-as 109



BGP Attributes: NEXT_HOP



Overriding Defaults:

EBGP NLRI only:
neighbor x.x.x.x next-hop-self

route-map override:
set ip next-hop { A.B.C.D | peer-address}

BGP Attributes
Used to shape routing policy

1: ORIGIN
2: AS-PATH
3: NEXT-HOP
4: MED
5: LOCAL_PREF
6: ATOMIC_AGGREGATE
7: AGGREGATOR
8: COMMUNITY
9: ORIGINATOR_ID
10: CLUSTER_LIST
14: MP_REACH_NLRI
15: MP_UNREACH_NLRI

BGP UPDATES

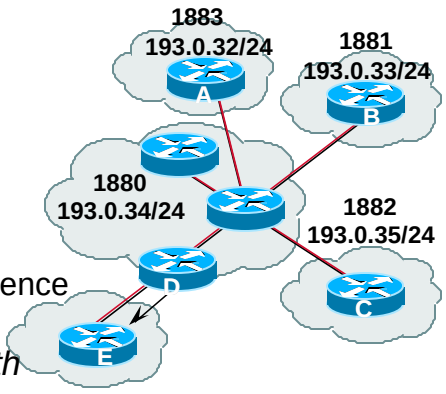
WITHDRAWS
ATTRIBUTES
PREFIXES (Next Layer Reachability Information)

Problem: Loop Detection, Policy Solution: AS-PATH

AS SEQUENCE
- List of AS's that a route has traversed

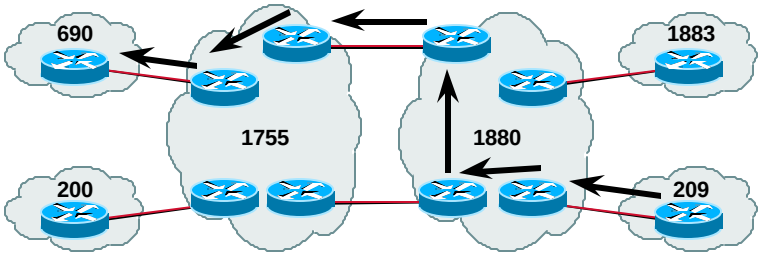
AS SET
- Summarizes contributing sequence
- Sequence ordering is lost

route-map prepend: *set as-path*



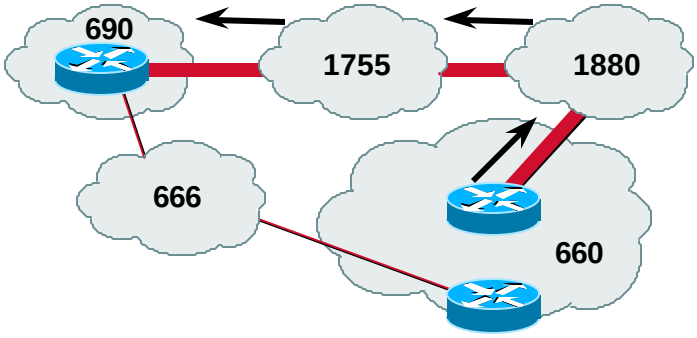
A: 193.0.33/24 1880 1881
B: 193.0.34/24 1880
C: 193.0.35/24 1880 1882
E: 193.0.32/22 1880 {1881,1882}

Problem: Indicate Best Path into AS
Solution: MED



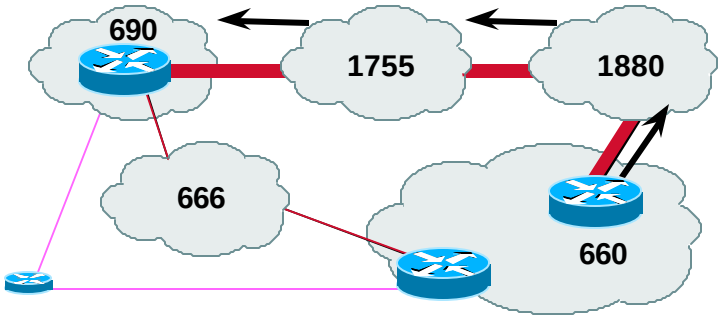
Non-transitive attribute
Conveys the relative preference of entry points
Comparable if paths from same AS
Unless "bgp always-compare-med" configured
route-map: *set metric { metric | internal}*

Problem: Override as-path/MED?
Solution: Local Preference



Attribute local to AS - mandatory for IBGP updates
route-map override: *set local-preference*

Problem: Override local preference
Solution: WEIGHT



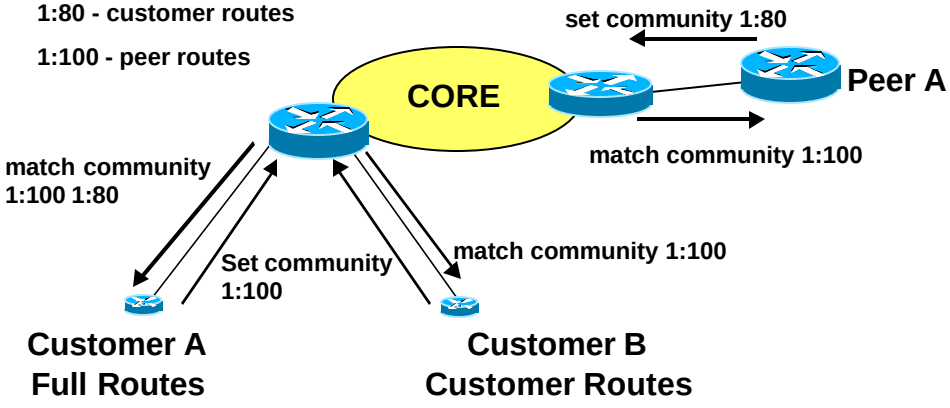
Local to router on which it's configured route-map: *set weight*
Highest weight wins over all valid paths

Problem: Scale routing policy
Solution: COMMUNITY

Communities:

1:80 - customer routes

1:100 - peer routes



BGP ATTRIBUTES: COMMUNITY

- Groups destinations to help scale policy application
- Typical communities:
  - destinations learned from customers
  - destinations learned from peers
  - destinations in VPN
  - destinations to receive preferential queuing

BGP ATTRIBUTES: COMMUNITY

- Activated per neighbor/peer-group*:*
  - *neighbor {peer-address | peer-group-name} send-community*
- Carried across AS boundaries
- Common convention is string of 4 bytes: <AS>:[0-65536]

BGP ATTRIBUTES: COMMUNITY

- **Each destination can be a member of *multiple* communities**
- **route-map: *set community***

<1-4294967295>	**community number**
aa:nn	**community number in aa:nn format**
additive	***Add to the existing community***
local-AS	***Do not send to EBGP peers (well-known community)***
no-advertise	***Do not advertise to any peer (well-known community)***
no-export	***Do not export outside AS/confed (well-known community)***
none	***No community attribute***

Least Useful Attribute Award: ORIGIN

- **IGP**

  network* statement under *router bgp

- **EGP**

  Redistributed from EGP

- **Incomplete**

  Redistribute IGP under *router bgp*

- **TIP: always use route-map override: *set origin igp***

Route Map "Set" Capabilities

as-path	Prepend string for a BGP AS-path attribute
comm-list	set BGP community list (for deletion)
community	BGP community attribute
dampening	Set BGP route flap dampening parameters
local-preference	BGP local preference path attribute
metric	Metric value for destination routing protocol
nlri	BGP NLRI type
origin	BGP origin code
weight	BGP weight for routing table
ip next-hop	{ A.B.C.D | peer-address }

BGP Attributes

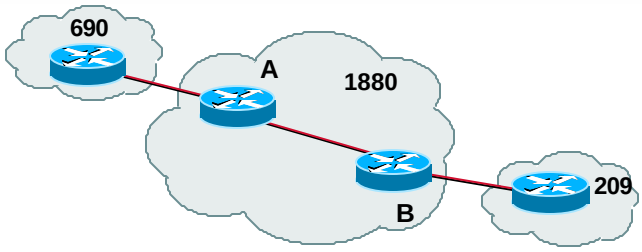
```
75k1#sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/24, version 139267814
Paths: (1 available, best #1)
  Not advertised to any peer
! AS-PATH                                    AS        ID
  65000 64000 {100 200},  (aggregated by 64000 16.0.0.2)
!   NEXT-HOP  IGP METRIC        PEER-IP  PEER-ID
    10.0.10.4       (metric 10)    from 10.0.0.1 (10.0.0.2)
      Origin IGP, metric 100, localpref 230, valid, aggregated
      internal (or external or local),
      atomic-aggregate, best
      Community: 64000:3 100:0 200:10
      Originator: 10.0.0.1, Cluster list: 16.0.0.4, 16.0.0.14
```

Basic Decision Algorithm

Consider only (synchronized) routes with no AS loops and valid next-hop, then prefer:

Highest WEIGHT
Highest LOCAL PREFERENCE
LOCALLY ORIGINATED (eg network/aggregate)
Shortest AS-PATH
Lowest ORIGIN (IGP < EGP < incomplete)
Lowest MED
EBGP
IBGP
Lowest IGP METRIC to next-hop

Synchronization



- **Rtr A won't advertise the prefixes from AS209 until the IGP converges.**
- **Turn synchronization off!**

  next-hop has to be known via IGP

  router bgp 1880
  no sync

General Issues

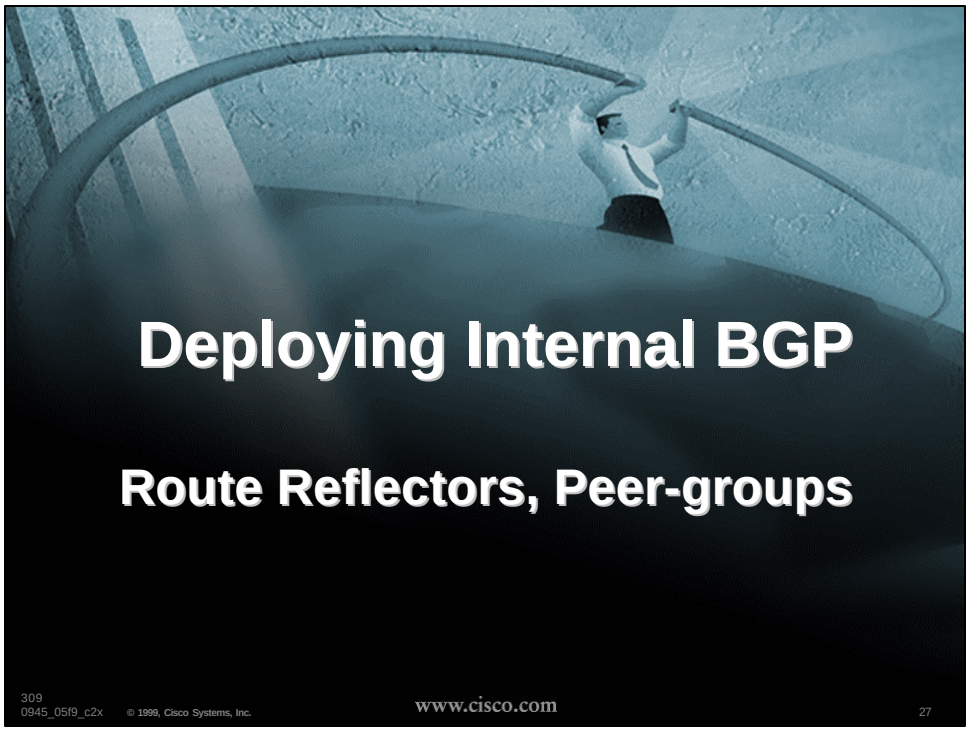
- **Synchronization: not required if you have a full IBGP mesh.**
- **=> Don't let BGP override IGP.**
- **auto-summary: use aggregation commands instead.**

```
router bgp 100
no synchronization
no auto-summary
distance 200 200 200
```


So Far....

Apply policy on a per AS basis
Group routes into even large communities
Choose exit and entry points for large policy groups using med/local preference

Will your policies scale????



Guidelines for Stable IBGP

- **Peer using loopback addresses**

  neighbor { ip address | peer-group}
  update-source loopback0

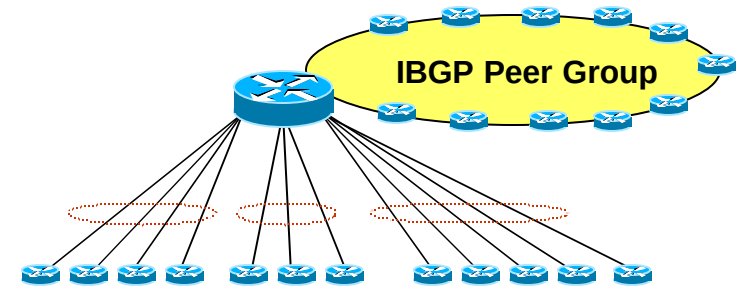
- **Independent of physical interface failure**
- **IGP performs any load-sharing**

Guidelines for Scaling IBGP

- Use peer groups and RRs
- Carry only next-hops in IGP
- Carry full routes in BGP only if necessary
- Do not redistribute BGP into IGP

Source: Placeholder for Notes, etc. 14 pt., bold

Using Peer Groups



Full Routes Peer Group | "Default" Peer Group | Customer Routes Peer Group

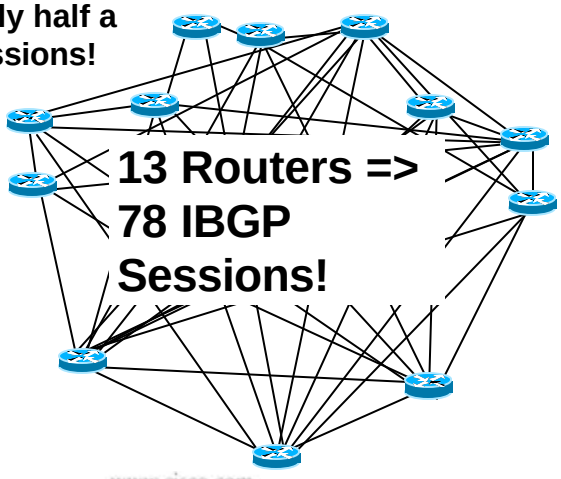
What Is a Peer Group?

- All peer-group members have a common outbound policy
- Updates generated once per peer-group
- Simplifies configuration
- Members can have different inbound policy

Why Route Reflectors?

Avoid n(n-1)/2 IBGP mesh

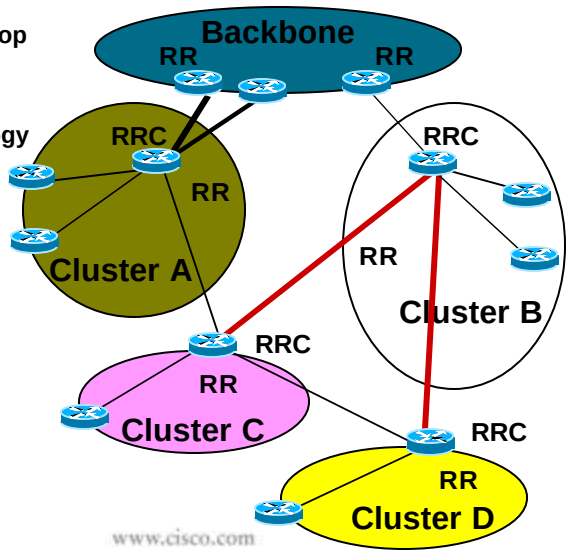
n=1000 => nearly half a million ibgp sessions!



Using Route Reflectors

Golden Rule of RR Loop Avoidance:

RR topology should follow physical topology



309
0945_05f9_c2x © 1999, Cisco Systems, Inc. www.cisco.com 33

What Is a Route Reflector?

- Reflector receives path from clients and no clients
- If best path is from a client, reflect to clients and non-clients
- If best path is from a non-client, reflect to clients

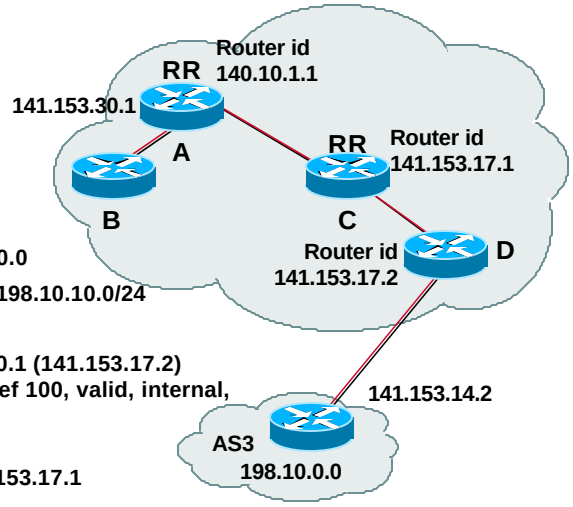
309
0945_05f9_c2x © 1999, Cisco Systems, Inc. www.cisco.com 34

0945_05f9_c2x.scr

Deploying Route Reflectors

- Divide backbone into multiple clusters
- Each cluster contains at least one RR (multiple for redundancy), and multiple clients
- RR's are fully meshed via IBGP
- Still use single IGP - next-hop unmodified by RR

Hierarchical RR



- **Example:**

RouterB>sh ip bgp 198.10.10.0

BGP routing table entry for 198.10.10.0/24

3

141.153.14.2 from 141.153.30.1 (141.153.17.2)
Origin IGP, metric 0, localpref 100, valid, internal, best

Originator : 141.153.17.2
Cluster list: 144.10.1.1, 141.153.17.1

BGP ATTRIBUTES: ORIGINATOR_ID

- **ORIGINATOR_ID**

  Router ID of IBGP speaker that reflects RR client routes to non-clients

  Overridden by: *bgp cluster-id x.x.x.x*

- **Useful for troubleshooting and loop detection**

BGP ATTRIBUTES: CLUSTER_LIST

- **CLUSTER_LIST**

  String of ORIGINATOR_IDs through which the route has passed

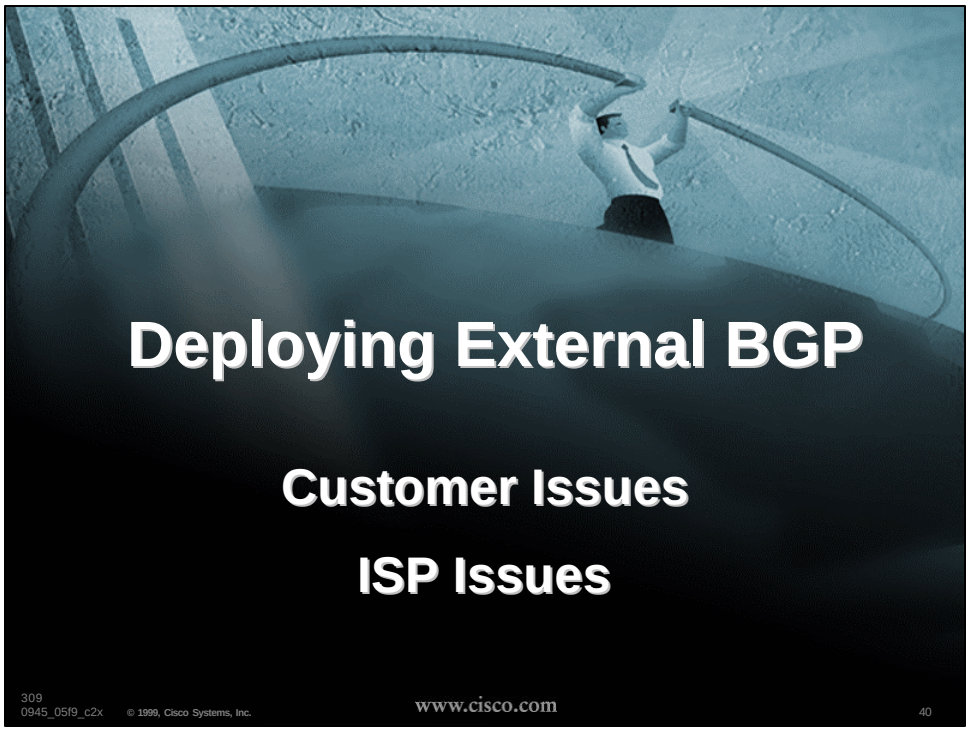
- **Useful for troubleshooting and loop detection**

So Far….

Is IBGP peering Stable?
- Use loopbacks for peering

Will it Scale?
- Use peer groups
- Use Route Reflectors

Simple, hierarchical config?



Customer Issues

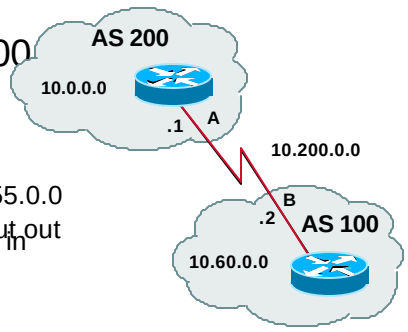
- **Steps:**
  - **Configure BGP**
  - **Generate a stable aggregate**
  - **Set inbound policy**
  - **Set output policy**
  - **Configure loadsharing/multihoming**

Connecting to an ISP

AS100 is a customer of AS200
Usually directly connected

Router B:

```
router bgp 110
aggregate-address 10.60.0.0 255.255.0.0
summary-only
neighbor 10.200.0.2 remote-as 109
neighbor 10.200.0.2 route-map ispout out
neighbor 10.200.0.2 route-map ispin in
ip route 10.60.0.0 255.255.0.0 null0
```



Source: Placeholder for Notes, etc. 14 pt., bold

What Is Aggregation?

- **Summarization based on specifics *from the BGP routing table***

  10.1.1.0 255.255.255.0

  10.2.0.0 255.255.0.0

  -> 10.0.0.0 255.0.0.0

How to Aggregate

- **Aggregate-address 10.0.0.0 255.0.0.0 {as-set} {summary-only} {route-map}**
- **Use *as-set* to include path and community info from specifics**
- ***summary-only* suppresses specifics**
- **route-map sets other attributes**

Why aggregate?

- Reduce number of Internet prefixes
- Increase stability - aggregate stays even specifics come and go.
- Consider using: aggregate/network/null route combination to lock-down all routes

Source: Placeholder for Notes, etc. 14 pt., bold

BGP Attributes ATOMIC AGGREGATE

- Indicates loss of AS-PATH information
- Must not be removed once set
- Set by: *aggregate-address x.x.x.x*
- Not set if as-set keyword is used, however, AS-SET and COMMUNITY then carries information about specifics

BGP Attributes: Aggregator

- AS number and IP address of router generating aggregate
- Useful for troubleshooting

Aggregate Attributes

NEXT_HOP = local

WEIGHT = 32768

LOCAL_PREF = best

AS_PATH = AS_SET or nothing

ORIGIN = worst

MED = none

Why Inbound Policy?

Apply a recognizable community to use in outbound filters or other policy

Set local-preference to override default of 100

multi-homing loadsharing - more later

```
route-map ISPin permit 10
set local-preference 200
set community 100:200
```


Why Outbound Policy?

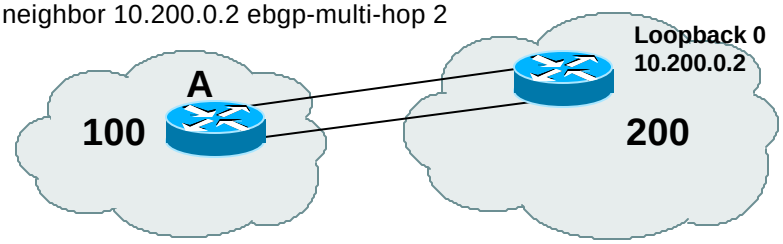
Outgoing prefix filter helps protect against mistakes (Can also apply community and as-path filters)

Send community based on agreements with ISP

```
route-map ISPout permit 10
    match ip prefix
    match community 100:0
    set community 100:1 additive
```

Load-sharing - single path

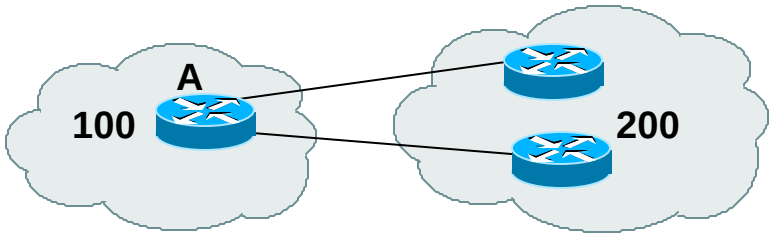
Router A:
interface loopback 0
ip address 10.60.0.1 255.255.255.255
!
router bgp 100
neighbor 10.200.0.2 remote-as 200
neighbor 10.200.0.2 update-source loopback0
neighbor 10.200.0.2 ebgp-multi-hop 2



Source: Placeholder for Notes, etc. 14 pt., bold

Load-sharing - multiple paths from same AS

Router A:
router bgp 100
neighbor 10.200.0.1 remote-as 200
neighbor 10.300.0.1 remote-as 200
maximum-paths 6



Note:A still only advertises one "best" path to ibgp peers

Source: Placeholder for Notes, etc. 14 pt., bold

What is Multihoming?

Connecting to two or more ISPs to increase:

Reliability: one ISP fails, still OK

Performance: better paths to common Internet destinations

Source: Placeholder for Notes, etc. 14 pt., bold

Types of Multihoming

Three common cases:

- default from all providers
- customer+default routes from all
- full routes from all

Default from all providers

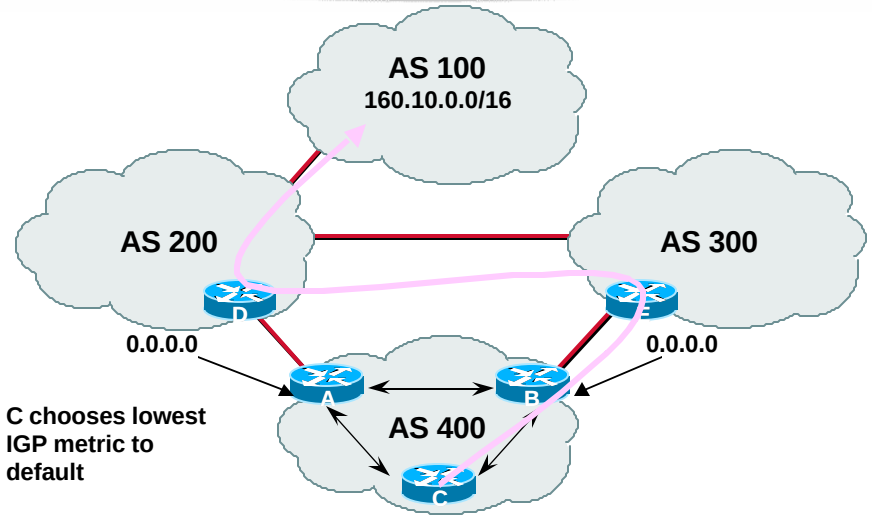
Low memory/CPU solution

Provider sends BGP default => provider decided by IGP metrics to reach default

You send all your routes to provider =>inbound path decided by Internet

- you can influence using AS-path

Default from all providers



Customer+default from all providers

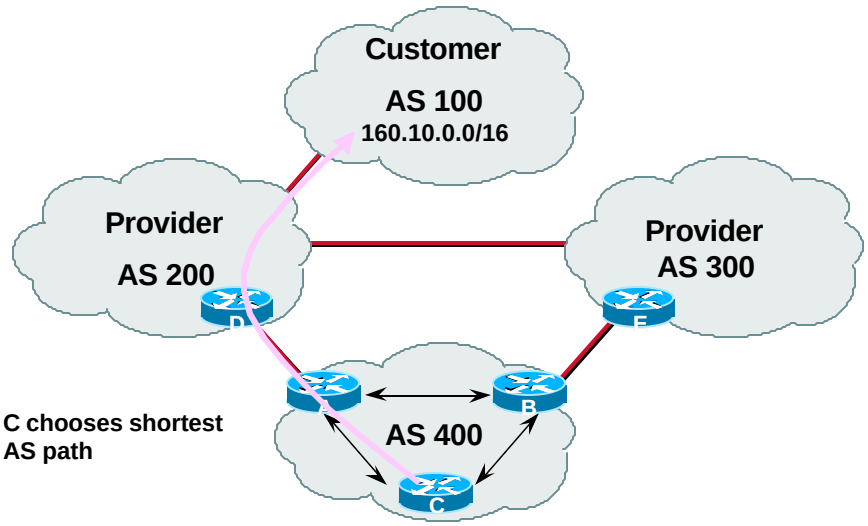
Medium memory and CPU

“best” path - usually shortest AS-path.

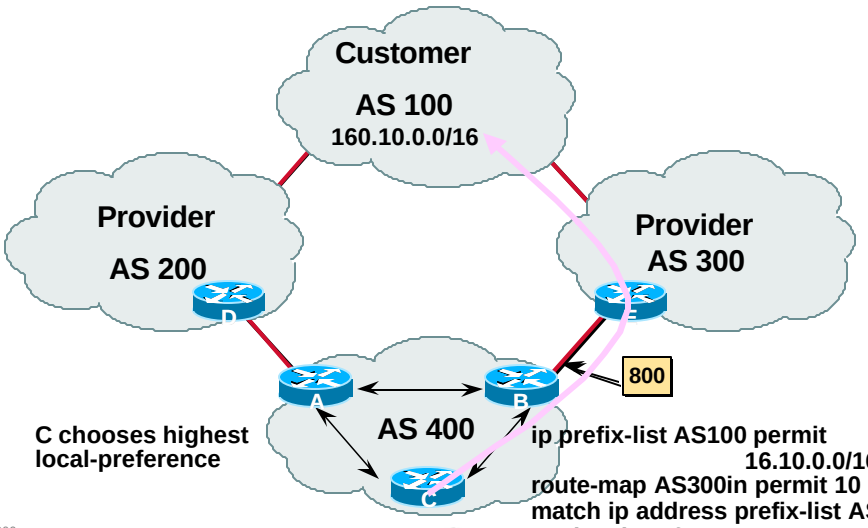
Use local-preference to override based on prefix, as-path, or community

IGP metric to default used for all other destinations

Customer routers from all providers



Customer routes from all providers



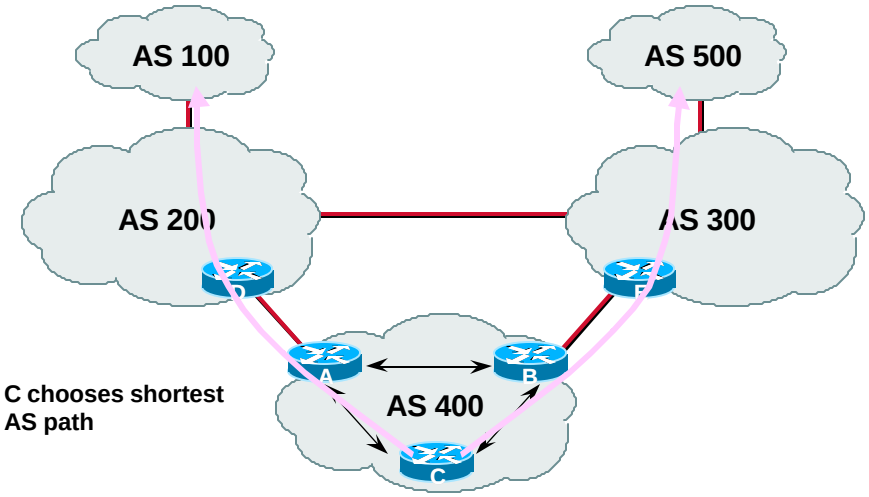
Full routes from both providers

Higher memory/CPU solution

Reach *all* destinations by “best” path - usually shortest AS path

Can still manually tune using local-pref and as-path/community/prefix matches

Full routes from all providers



Controlling Inbound Traffic?

Inbound is very difficult due to lack of transitive metric.

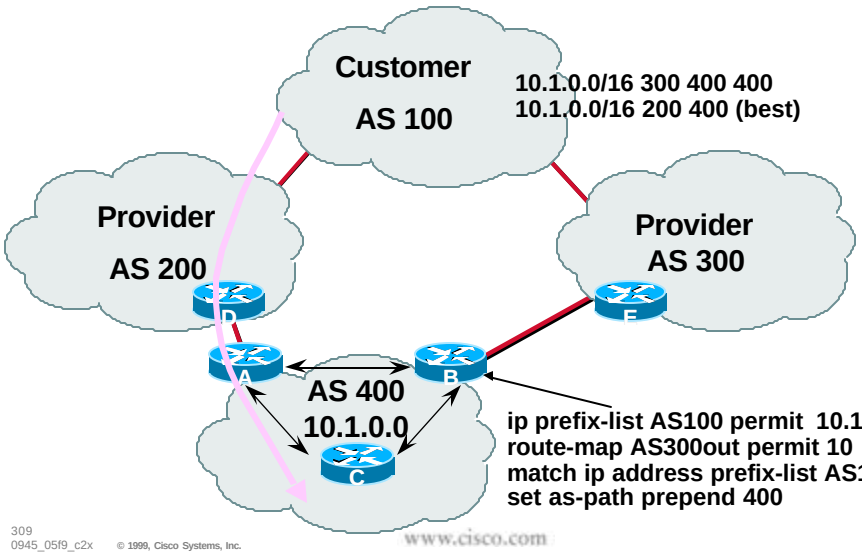
Can divide outgoing updates across providers, but what happens to redundancy?

Controlling Inbound Traffic?

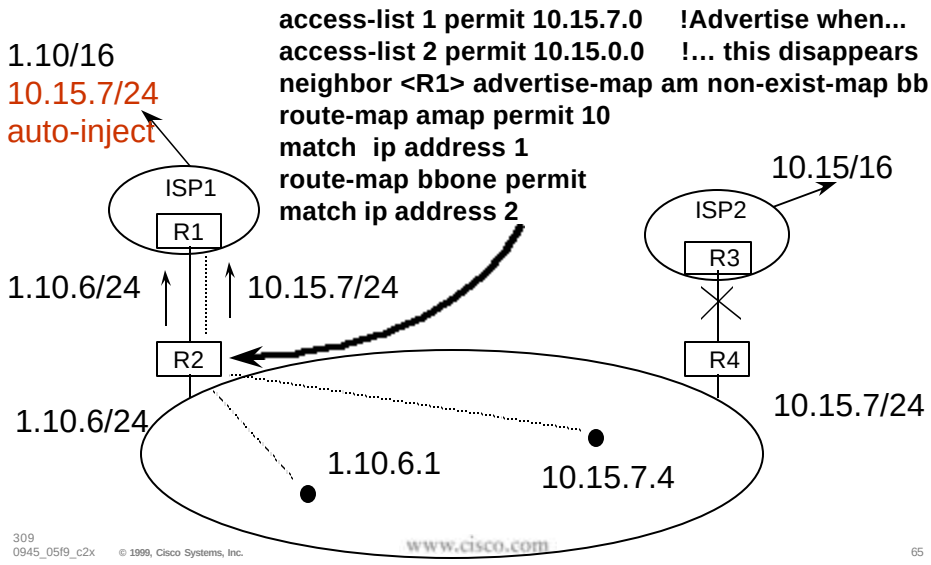
Bad Internet Citizen:
- Divide address space
- “set as-path prepend

Good Internet Citizen
- Divide address space
- use “advertise maps”

Using AS-PATH prepend



Using an advertise-map



So Far....

Stability through:
- Aggregation
- Multihoming
- Inbound/Outbound Filtering

Scalability of memory/CPU:
- default, customer routes, full routes

Simplicity using “standard” solutions

ISP Issues

- Scale BGP customer aggregation
- Offer a choice of route-feeds
- Peer with other providers
- Minimize BGP activity and protect against customer mis-configurations
- Provide a backup service
- Propagate QoS Policy

Customer Aggregation Guidelines

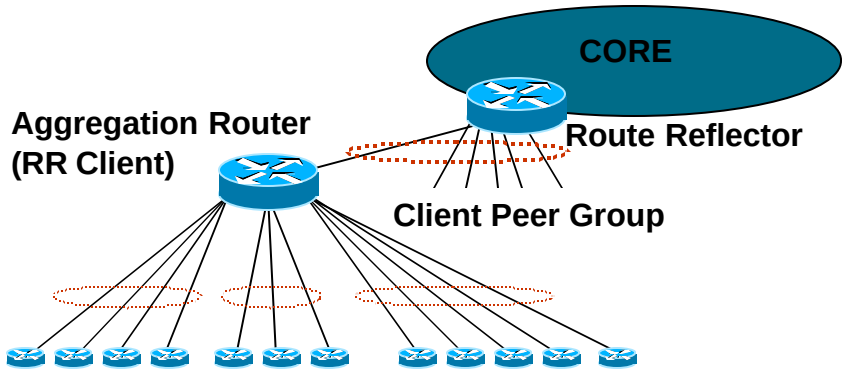
Define at least three peer groups:

- cust-default - send default route only
- cust-cust - send customer routes only
- cust-full - send full Internet routes

Identify routes via communities

Apply passwords and an inbound prefix-list on a *per neighbor* basis:

Customer Aggregation



TIP: apply passwords and inbound prefix-list to *each* customer

cust-default peer-group

```
Neighbor cust-full peer-group
neighbor cust-full description Send full Routes
neighbor cust-full remove-private-AS
neighbor cust-full version 4
neighbor cust-full route-map cust-in in
neighbor cust-full prefix-list cidr-block out
neighbor cust-full route-map full-routes out
.
ip prefix-list cidr-block seq 5 deny 10.0.0.0/8 ge 9
ip prefix-list cidr-block seq 10 permit 0.0.0.0/0 le 32
```

cust-full outgoing route-map

```
route-map full-routes permit 10
! matches routes from customers & other ISPs
match community 1 80     ;
! send MED equal to IGP metric
set metric-type internal
! ensure next-hop IP our own address
set ip next-hop peer-address
```


cust-full incoming route-map

```
route-map cust-in permit 10
! Ignore MED
set metric 4294967294
set ip next-hop peer-address
set community 2:100 additive
```

cust-cust peer-group

```
Neighbor cust-cust peer-group
neighbor cust-cust description customer routes
neighbor cust-cust remove-private-AS
neighbor cust-cust version 4
neighbor cust-cust route-map cust-in in
neighbor cust-cust prefix-list cidr-block out
neighbor cust-cust route-map cust-routes out
```


cust-routes route-map

```
route-map cust-routes permit 10
! matches routes from customers only
match community 1
! send MED equal to IGP metric
set metric-type internal
! ensure next-hop IP our own address
sey ip next-hop peer-address
```

default route peer-group

```
neighbor cust-default peer-group
neighbor cust-default description Send default
neighbor cust-default default-originate
                              route-map default-route
neighbor cust-default remove-private-AS
neighbor cust-default version 4
neighbor cust-default route-map cust-in in
neighbor cust-default prefix-list deny-all out

ip prefix-list deny-all seq 5 deny 0.0.0.0/0 le 32
```


default-route route-map

```
route-map default-route permit 10
! Send MED equal to IGP metric
set metric-type internal
! ensure next-hop IP our own address
set ip next-hop peer-address
```

Peer groups for NAPs

- Similar to EBGP customer aggregation except inbound prefix filtering is rarely used (lack of registry)
- Use maximum-prefix and prefix sanity checking instead.
- Still use per-neighbor passwords!

Peer groups for NAPS

```
neighbor nap peer-group
neighbor nap description for peer ISPs
neighbor nap remove-private-AS
neighbor nap version 4
neighbor nap prefix-list sanity-check in
neighbor nap prefix-list cidr-block out
neighbor nap route-map nap-out out
neighbor nap maximum prefix 30000
```

Peer groups for NAPS

```
route-map nap-out permit 10
! matches customers only
match community 1
! Send MED equal to IGP metric
set metric-type internal
! ensure next-hop IP our own address
set ip next-hop peer-address
```


Peer groups for NAPS: sanity-check prefix-list

```
# FIRST - FILTER OUT YOUR IGP ADDRESS SPACE!!
ip prefix-list sanity-check seq 5 deny 0.0.0.0/32
# deny the default route
ip prefix-list sanity-check seq 10 deny 0.0.0.0/8 le 32
# deny anything beginning with 0
ip prefix-list sanity-check seq 15 deny 0.0.0.0/1 ge 20
# deny masks > 20 for all class A nets (1-127)
ip prefix-list sanity-check seq 20 deny 10.0.0.0/8 le 32
# deny 10/8 per RFC1918
ip prefix-list sanity-check seq 25 deny 127.0.0.0/8 le 32
# reserved by IANA - loopback address
ip prefix-list sanity-check seq 30 deny 128.0.0.0/2 ge 17
deny masks >= 17 for all class B nets (129-191)
ip prefix-list sanity-check seq 35 deny 128.0.0.0/16 le 32
# deny net 128.0 - reserved by IANA
ip prefix-list sanity-check seq 40 deny 172.16.0.0/12 le 32
# deny 172.16 as RFC1918
```

Peer groups for NAPS: sanity-check prefix-list

```
ip prefix-list sanity-check seq 45 deny 192.0.2.0/24 le 32
# class C 192.0.20.0 reserved by IANA
ip prefix-list sanity-check seq 50 deny 192.0.0.0/24 le 32
# class C 192.0.0.0 reserved by IANA
ip prefix-list sanity-check seq 55 deny 192.168.0.0/16 le 32
# deny 192.168/16 per RFC1918
ip prefix-list sanity-check seq 60 deny 191.255.0.0/16 le 32
# deny 191.255.0.0 - IANA reserved (I think)
ip prefix-list sanity-check seq 65 deny 192.0.0.0/3 ge 25
# deny masks > 25 for class C (192-222)
ip prefix-list sanity-check seq 70 deny 223.255.255.0/24 le 32
# deny anything in net 223 - IANA reserved
ip prefix-list sanity-check seq 75 deny 224.0.0.0/3 le 32
# deny class D/Experimental
```


Summary

- **Scalability:**
  - **Use attributes, especially community**
  - **Use peer-groups and route-reflectors**
- **Stability:**
  - **Use loopback addresses for IBGP**
  - **Generate aggregates**
  - **Apply passwords**
  - **Always filter inbound and outbound**

Summary

- **Simplicity - standard solutions:**
  - **Three multi-homing options**
  - **Group customers into communities**
  - **Apply standard policy at the edge**
  - **Avoid “special configs”**
  - **Script your config generation**

For Further Reference:

- **Case Studies on WWW.CISCO.COM**
- **Sam Halabi. Internet Routing Architectures. Cisco Press.**
- **John Stewart. BGP4. Addison Wesley**

Source: Placeholder for Notes, etc. 14 pt., bold

Thank You!

- **Related presentations:**

  Advanced BGP and Troubleshooting

- **Questions?**