TWO SCALABILITY ISSUES

- NETWORK ADDRESSES (SCALABLE BUT NOT ENOUGH IN TERM OF USE)
  - A NETWORK WITH 2 NODES USES AN ENTIRE C CLASS NETWORK, WASTING 253 ADDRESSES
  - A NETWORK WITH SLIGHTLY > 255 HOSTS WASTES OVER 64,000 ADDRESSES (NEED B CLASS)
- ROUTING (IN INTERNET)
  - BELLMAN-FORD (LOOPS & SLOW CONVERGENCE)
  - LINK STATE (TOO MUCH STATE INFO)
  - NO COMMON METRICS (e.g. WEIGHTS)

NETWORK ADDRESSES

class B Address	NETWORK #	HOST #
Subnet MASK e.g. 255.255.255.0	1 . . . 1	0.....0
SUBNET ADDRESS	SUBNET #	HOST ID

- MAKE PART OF HOST GO TO NETWORK # & CALL IT SUBNET
- CONFIGURE HOSTS WITH A SUBNET MASK
- HOST IP **AND** SUBNET MASK = SUBNET #
- ALL HOSTS ON SAME SUBNET GIVE SAME SUBNET #

MODIFICATIONS

- NOW SAME NETWORK ACTUALLY MEANS SAME SUBNET
- H1 & H2 ON SAME NETWORK $\Rightarrow$

$$IP_1 \wedge \text{SUBNET MASK} = IP_2 \wedge \text{SUBNET MASK}$$

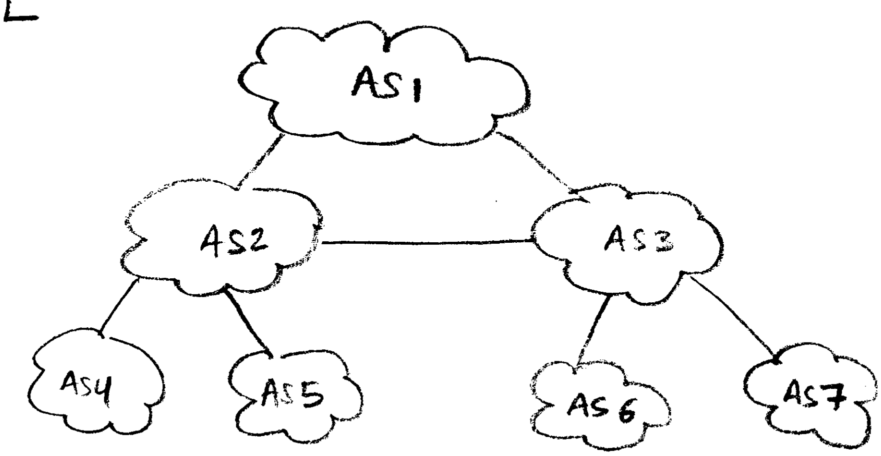
- ROUTER TABLE

	SUBNET #	SUBNET MASK	NEXT-HOP
eg.	128.96.34.0	255.255.255.128	Interface 0

- UPON RECEIVING A PACKET, AND ITS IP WITH EVERY SUBNET MASK & CHECK IF IT GIVES THE CORRESPONDING SUBNET #.

ROUTING

- EACH NETWORK IS REGARDED AS AUTONOMOUS SYSTEM AS, EACH NETWORK HAS AN AS # (16 bit #)
- AS is A NETWORK UNDER SAME ADMINISTRATIVE CONTROL

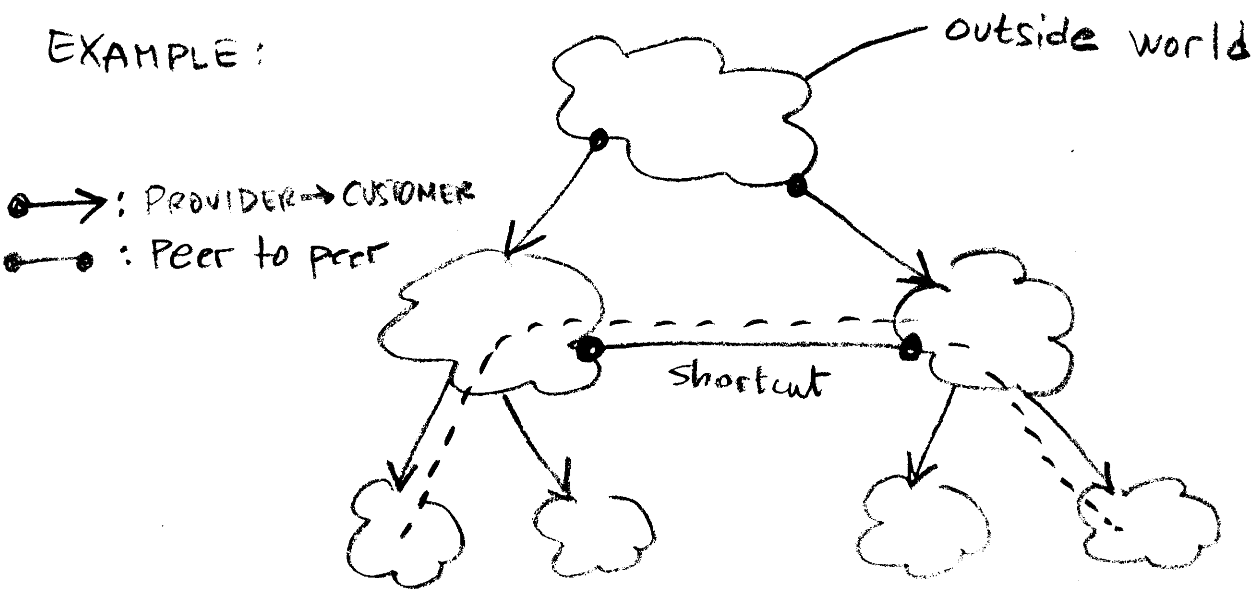


RELATIONSHIPS

- ASes USUALLY DEFINE RELATIONSHIPS AMONG THEM
  - CUSTOMER - PROVIDER
  - PEERS

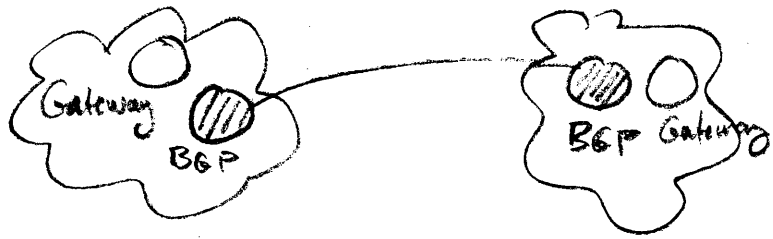
- ROUTING IS DONE DEPENDING ON POLICIES
  - AN AS MIGHT NOT WANT TO ROUTE THROUGH ANOTHER AS
  - AN AS MIGHT WANT TO ROUTE THROUGH A PEER (INSTEAD OF GOING TO PROVIDER) TO SAVE $$$ (PEERS USUALLY DON'T EXCHANGE $$$

EXAMPLE:



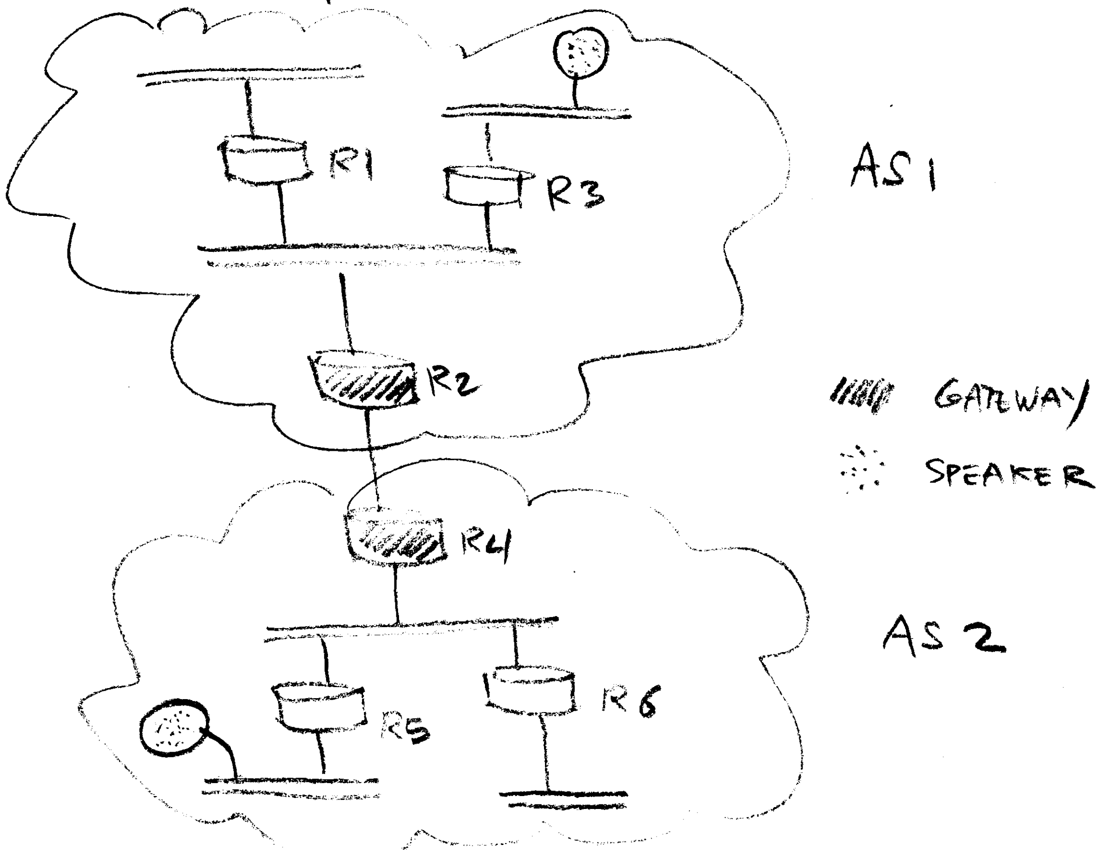
BGP
(BORDER GATEWAY PROTOCOL)

- EVERY AS HAS A BGP SPEAKER & BGP Gateway



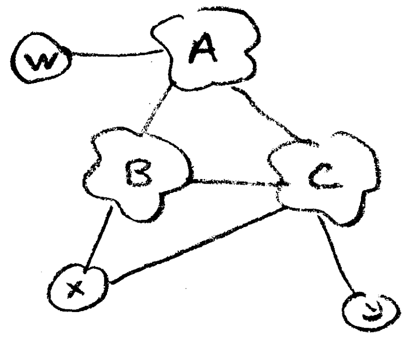
- BGP SPEAKERS EXCHANGE ROUTE INFORMATION
- GATEWAY : POINT OF ENTRY TO AS
- BGP & GATEWAY NEED NOT BE THE SAME, BUT THEY COULD BE

EXAMPLE



EXCHANGING ROUTES

- EVERY AS DECIDES TO ADVERTISE PATHS IT KNOWS TO OTHER AS (COMPLETE PATH)



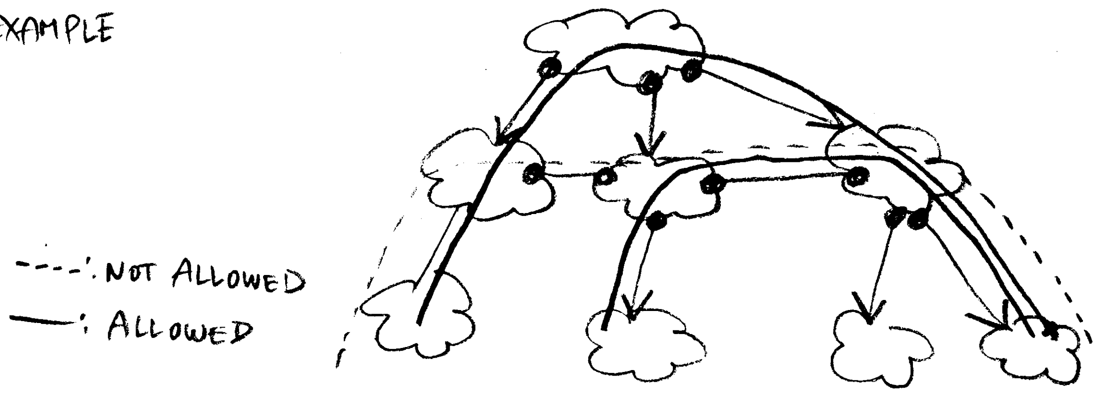
- FOR INSTANCE, C COULD ADVERTIZE TO B & A THAT IT CAN REACH Y & X
- EVERY AS ALSO DECIDES TO ADVERTIZE INFORMATION IT KNOWS FROM OTHER ASes.

LOOPS

- C ADVERTISES Y TO A : (C, Y)
- A ADVERTISES Y TO B : (A, C, Y)
- B ADVERISES Y TO C : (B, A, C, Y)
- WHAT IF C DECIDES TO SEND TO Y THROUGH B ?
  ⟹ LOOP
- EVERY AS CHECKS IF IT IS PART OF THE PATH BEFORE ACCEPTING IT ⟹ DISCARD PATHS CONTAINING ID OF THE AS ITSELF

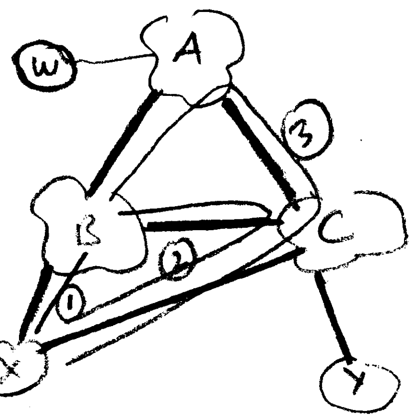
WHAT TO IMPORT, WHAT TO EXPORT?

- USUALLY WE WOULD LIKE "VALLEY-FREE" ROUTES
  - NUMBER LINKS (+1, 0, −1) FOR PROVIDER, PEER, CUSTOMER
  - IN ANY PATH WE WANT SEQUENCE OF +1, FOLLOWED BY AT MOST ONE 0, FOLLOWED BY SEQUECE OF −1.
- EXAMPLE



IMPORTING RULES

- IMPORT EVERYTHING BUT GIVE PREFERENCE TO
  - CUSTOMER FIRST
  - THEN PEER
  - THEN PROVIDER
- THEN AMONG DIFFERENT PATHS, CHOOSE SHORTEST.



B WILL EVENTUALLY LEARN

3 (A, C, X)
2 (C, X)
1 X

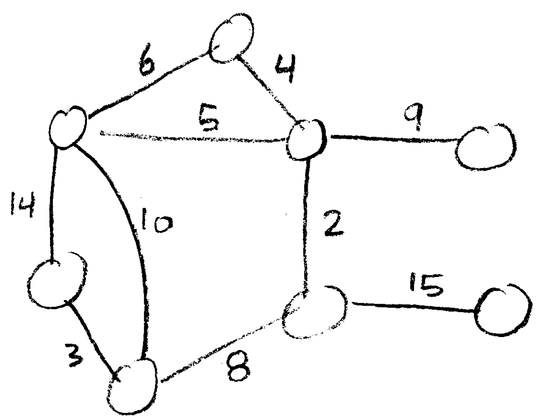
priority

EXPORTING RULES

- EXPORT CUSTOMERS ROUTES TO EVERYONE
  (NEED EVERYONE TO BE ABLE TO REACH YOUR CUSTOMERS)
- EXPORT ROUTES TO YOUR OWN ADDRESSES TO EVERYONE
  (NEED TO BE CONNECTED)
- DON'T EXPORT ROUTES ADVERTISED BY YOUR PROVIDER
  (MAY ADVERTISE TO CUSTOMERS) (VALLEY-FREE ROUTES)
- DON'T EXPORT ROUTES ADVERTISED BY PEER
  (MAY ADVERTISE TO CUSTOMERS) (VALLEY-FREE ROUTES)

MINIMUM SPANNING TREE

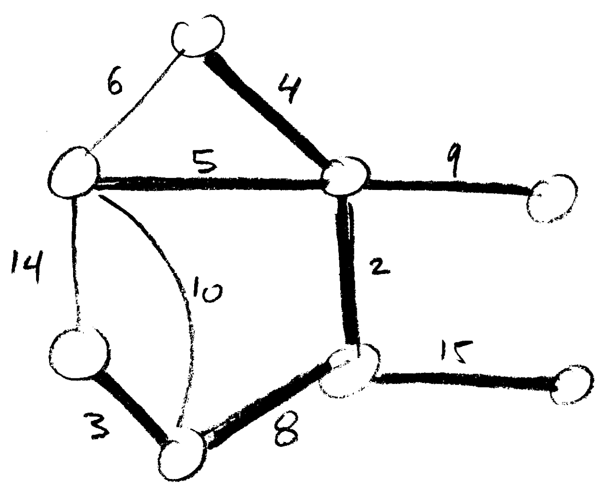
- ASSUME UNDIRECTED GRAPH
- EACH EDGE HAS WEIGHT



- FIND A SPANNING TREE T WITH MINIMUM WEIGHT

  i.e. $\sum_{(u,v)\in T} w(u,v)$ MINIMIZED

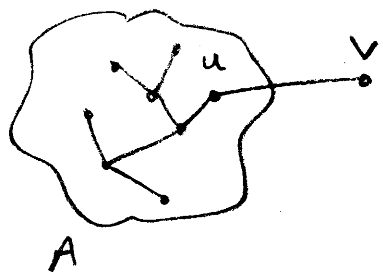
EXAMPLE



$w(T) = 3+8+2+15+9+5+4 = 46$

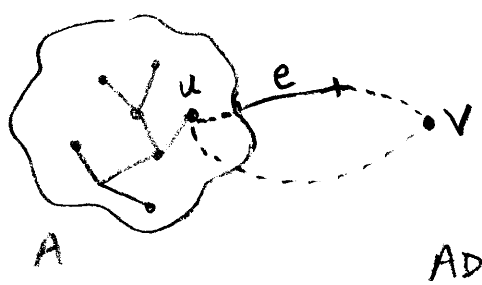
HOW DO WE FIND IT?

- LET T BE THE MST & LET $A \subseteq T$ BE A SUBTREE OF T.
- LET (u,v) BE MINIMUM WEIGHT EDGE THAT CONNECTS A TO $V-A$



THEN $(u,v) \in T$

PROOF



ASSUME T CONTAINS A BUT NOT (u,v)

ADD (u,v)
REMOVE e
OBTAIN T' WITH
$w(T') \leq w(T)$

PRIMS ALGORITHM

- START WITH ANY NODE
- BUILD THE TREE BY ALWAYS CHOOSING THE EDGE WITH SMALLEST WEIGHT THAT GOES OUTSIDE

THIS IS SIMILAR TO DIJKSTRA'S ALGORITHM

<u>INIT</u> $key(s) \leftarrow 0$
$key(u) \leftarrow \infty \quad u \neq s$

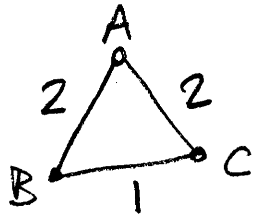
<u>ALG.</u> $S \leftarrow \phi$
while $(S \neq V)$
  do choose $u \notin S$ WITH SMALLEST key
    $S \leftarrow S \cup \{u\}$
    for every $(u,v) \in E$ & $v \notin S$
      if $w(u,v) < key(v)$
        then $key(v) \leftarrow w(u,v)$
          $p(v) \leftarrow u$

KRUSKAL'S ALGORITHM

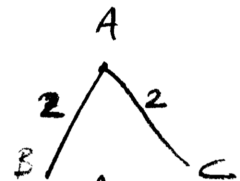
- REPEATEDLY CHOOSE THE SMALLEST WEIGHT EDGE THAT DOES NO CREATE A CYCLE & ADD IT
- ALSO CAN BE PROVED TO WORK

MINIMUM SPANNING TREE
&
SHORTEST PATH TREE

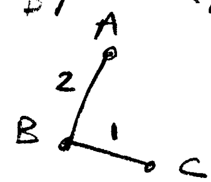
- THEY ARE NOT THE SAME.
- COUNTER EXAMPLE

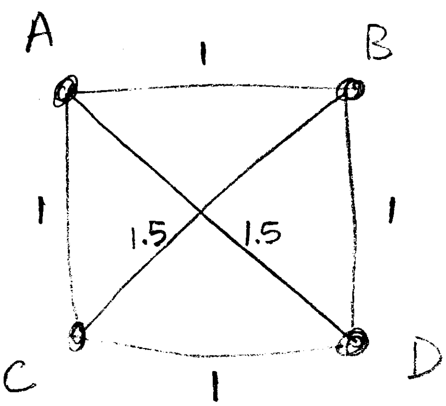


- SHORTEST PATH TREE FOR A AS SOURCE:



- MINIMUM SPANNING TREE STARTING WITH A:





NO SHORTEST PATH TREE IS A MINIMUM SPANNING TREE