Internet Engineering

241-461

Robert Elz

kre@munnari.OZ.AU
kre@coe.psu.ac.th

http://fivedots.coe.psu.ac.th/~kre

Contents

- Why Fragmentation?
- Why Reassembly?
- When to Fragment
- When to Reassemble
- How to Fragment
- How to Reassemble (cont)
- Why not fragment
- How to avoid fragmentation (PMTUD)

Reassembly Issues

- What happens if a fragment is lost ??
  - That is, never arrives at destination
- Does reassembly queue wait forever ?
  - No - we hope anyway!
- What prevents waiting forever ?

TTL and Reassembly

- Time To Live
  - Exists in every IP packet header
  - Including in headers of fragments
    - which are IP packets anyway!
  - Including fragments on reassembly queue
- TTL is upper limit (in seconds)
  - for how long packet can exist
- Every second
    - Or more often sometimes
  - TTL must be decremented
    - decreased by one
- When TTL reaches 0
  - packet must be discarded
  - Even if at destination in reassembly queue

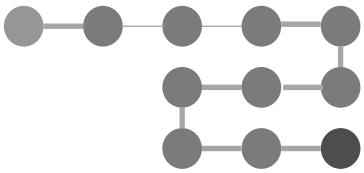
TTL and Reassembly (2)

- TTL cleans up reassembly queues
  - eventually the TTLs reach 0
  - fragments on queue discarded
- Also note TTL procedure
    - when packets merged
  - Take minimum of TTL values from fragments
  - TTL must NEVER increase
    - not in any packet or fragment
- Be aware of the effect of lost fragments
  - Entire packet ends up lost
    - Fragments not lost discarded at destination
    - Because the reassembly queue never merges
      - Back into the original packet
    - The missing piece leaves a hole
- So...

Contents

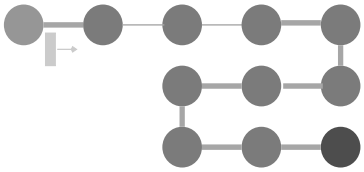
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



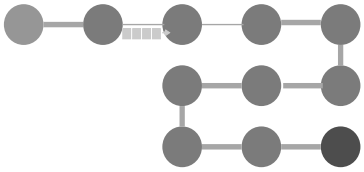
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



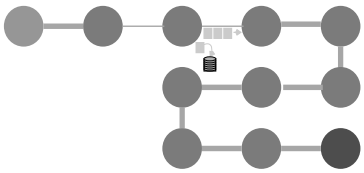
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



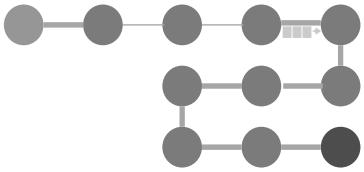
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



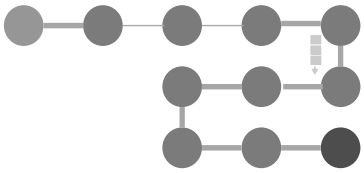
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



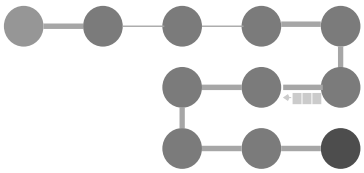
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



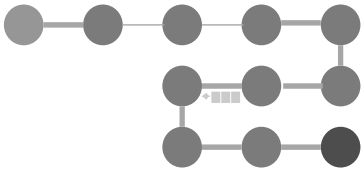
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



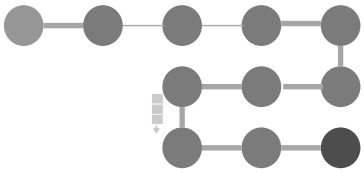
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



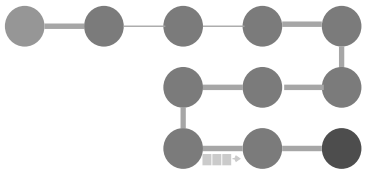
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



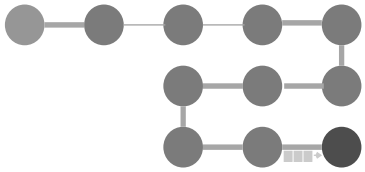
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



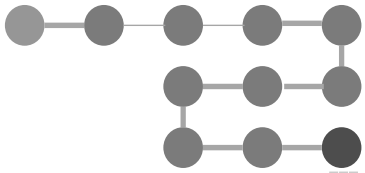
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



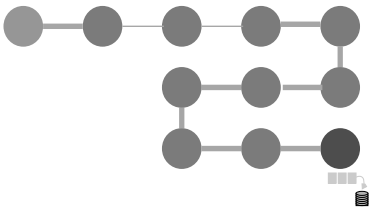
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



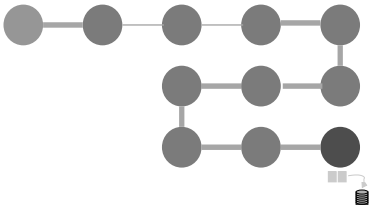
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



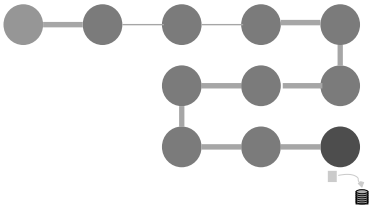
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



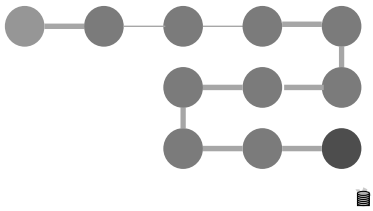
Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



Why not fragment?

- One lost fragment
  - means whole packet lost
- But fragments not lost in network
  - still delivered to destination
  - even though useless
  - waste bandwidth



Why not fragment (2)

- Usual cause of packet loss
  - Congestion
    - Very busy link
    - Insufficient capacity for demand
- Congestion occurs
  - data must be discarded
  - happens to be a fragment
    - other fragments still transmitted
    - over congested link
  - more data must be discarded
    - probably some other packet
- Discard good data
  - Transmit useless data !!

Fragmentation should be avoided

wherever possible

Contents

Avoiding Fragmentation

- If host sends small packets
  - fragmentation will not be needed
- If packet is lost
  - host can retransmit just that packet
- Other packets that reach destination
  - can be retained and used
    - Transport Protocol issues (TCP)
- How small is small enough?
  - 68 bytes certainly
    - Packets 68 bytes or less cannot be fragmented
    - Network must be able to send 68 byte packets
      - on every link used for IP
  - Using 68 byte packets
    - 20 byte IP header (minimum)
    - 20 byte TCP header (minimum)
      - 40 bytes of headers
    - 28 bytes remain for data
      - Almost 60% overhead - best case

Avoiding Fragmentation (2)

- 68 byte packets avoid fragmentation
  - 100% certainly
  - But too much overhead
- Can we send bigger packets
  - And still avoid fragmentation?
- Perhaps
  - It depends upon the network
- Usually assume 576 byte packets are OK
  - Certainly not guaranteed
  - Some slow links fragment around 200 bytes
    - so packets transmit quickly
- Almost always OK
  - Because of common mistake
  - Often believed that 576
    - is minimum required MTU
  - 576 is minimum packet reassembly size

Avoiding Fragmentation (2)

- In current network
    - Links are mostly
  - Ethernet MTU == 1500
  - ATM
    - transmits 48 byte cells (+ATM headers)
    - too small for IP packet (< 68 bytes)
    - must do link level fragmentation
      - fake a higher MTU
      - Can make it anything needed
        - Often set at 1500
        - for compatability with ethernet
  - Point to Point
    - transmits bytes (synchronously or asynchronously)
    - added link layer protocol
      - PPP usually
      - sends as many bytes as required
        - provided checksum still works well
      - Often set at 1500
        - compatability with ethernet
      - sometimes smaller (slow links)

Contents

- Why Fragmentation?
- Why Reassembly?
- When to Fragment
- When to Reassemble
- How to Fragment
- How to Reassemble
- Why not fragment
- How to avoid fragmentation (PMTUD)
  - Send packets small enough
  - What is Small enough?
  - Path MTU Discovery

Avoiding Fragmentation (3)

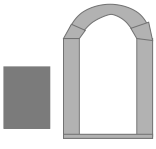
- Can often send 1500 byte packets
  - No fragmentation required
  - < 3% header overhead (40 header bytes)
- Rarely possible to send bigger
  - While still avoiding fragmentation
  - Usually at least 1 ethernet in the path
- Is OFTEN good enough ?
  - No - not really
    - Too many links still < 1500 bytes MTU
    - One example
      - Put packet inside another packet
      - Send outside packet across net
        carrying other packet inside it
      - Remove outer packet
        continue sending the inner one
      - Known as a tunnel

Tunnelling

Tunnelling

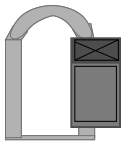
Tunnelling

Tunnelling

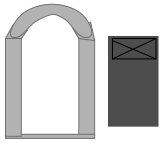


Tunnelling

Tunnelling



Tunnelling

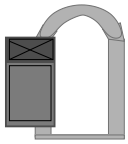


Tunnelling

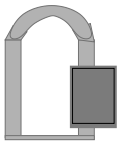
Tunnelling

Tunnelling

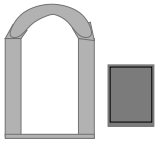
Tunnelling



Tunnelling



Tunnelling



Tunnelling

- Red (outer) packet <= 1500 bytes
  - to avoid fragmentation
- Thus green (inner) packet must be
  - <= 1500 bytes - red packet header overhead
  - Or red packet will be fragmented
- If our packet might be green packet
    - And we have no way to know for sure
  - We must send < 1500 bytes
    - or fragmentation will probably happen
- How big can we send?
  - And how do we find out?

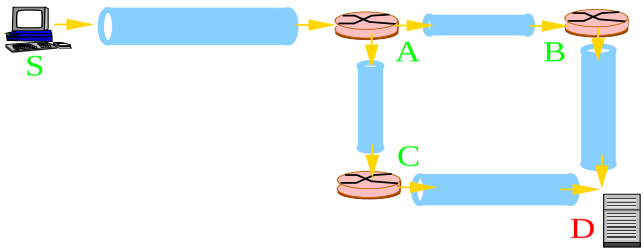
Contents

Path MTU Discovery

- We need a mechanism
  - To find the smallest MTU
  - On the current path
  - From sender (me) to the destination
- The smallest MTU on the path
  - The Minimum Maximum Transmission Unit
  - that is the choke point for fragmentation
  - send packets <= this Minimum MTU in size
    - fragmentation never required on this path
  - can send packets this big
    - no need to send smaller
    - lower header overheads
- How do we find this important number ?
  - This is Path MTU Discovery PMTUD
    - Finds the MTU of the current path
      - from sender to recipient

Path MTU Discovery (2)

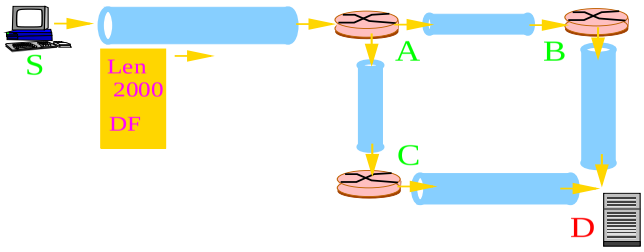
- Recall this example



  - We want to find the Path MTU (PMTU)
    - from sender S to destination D
- Start by assuming
  - PMTU is MTU of local link
    - certainly cannot be bigger than that
- Send biggest allowable packet
  - That will avoid fragmentation
- With the DF flag bit set in IP header

Path MTU Discovery (3)

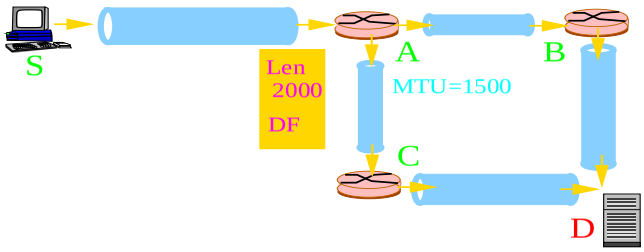
- Sender sends biggest packet
  - Local link will allow
  - Sets DF in header



  - Packet arrives at router
    - Where next link MTU
    - Smaller than packet size

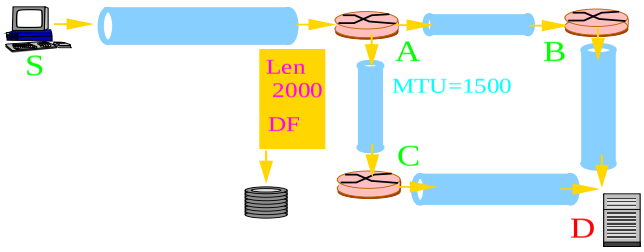
Path MTU Discovery (3)

- Sender sends biggest packet
  - Local link will allow
  - Sets DF in header



Path MTU Discovery (3)

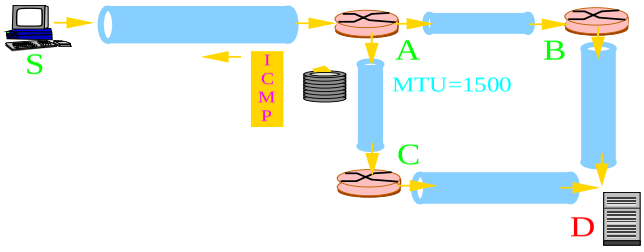
- Packet bigger than MTU of link to C
  - We assume path is S-A-C-D
- Router must discard packet
  - Too big for link, and DF is set



- Router also sends ICMP message to S
  - Tells it that packet was discarded
  - And why packet was discarded

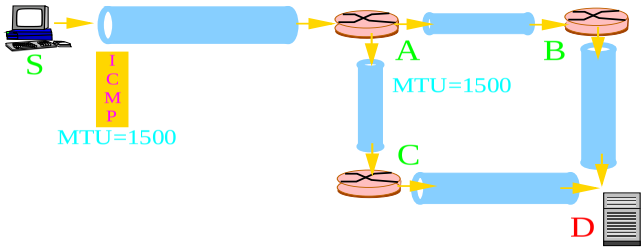
Path MTU Discovery (3)

- Packet bigger than MTU of link to C
  - We assume path is S-A-C-D
- Router must discard packet
  - Too big for link, and DF is set



Path MTU Discovery (4)

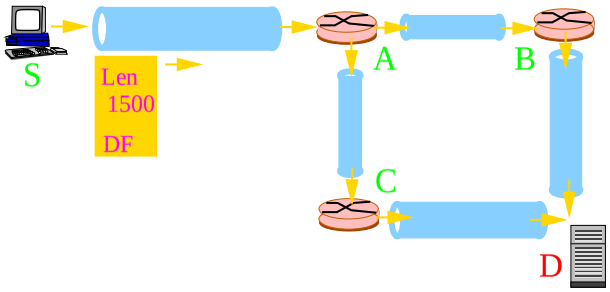
- S receives ICMP message
  - Discovers that packet it sent was dropped
  - Dropped because it was too big for link
    - And that 1500 is the available MTU



  - S can now send a smaller packet
    - limited to 1500 bytes max
  - And remember that 1500 is Path MTU to D
  - This is Path MTU Discovery

More PMTUD

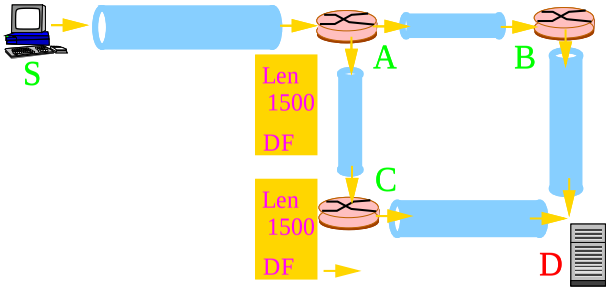
- S now knows 1500 as PMTU to D
  - So sends packets that big



  - These go via A and C, and arrive at D

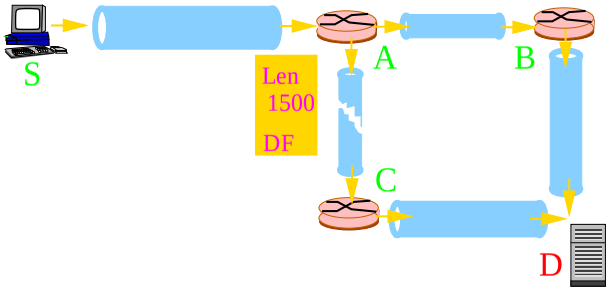
More PMTUD

- S now knows 1500 as PMTU to D
  - So sends packets that big



More PMTUD (2)

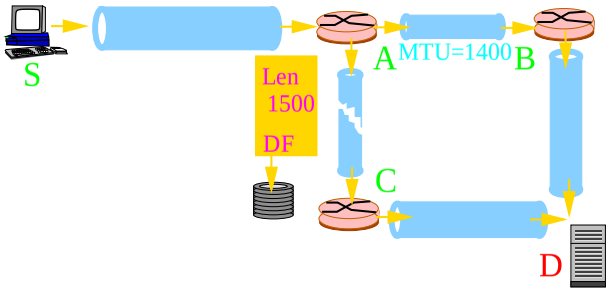
- S keeps sending with DF flag set
  - This is because...



  - The link from A to C might fail
  - The alternate link (from A to B)
    - has an MTU of just 1400
  - Because the DF flag remains set
    - A discards the packet
      - 1500 bytes is too big for MTU==1400 link

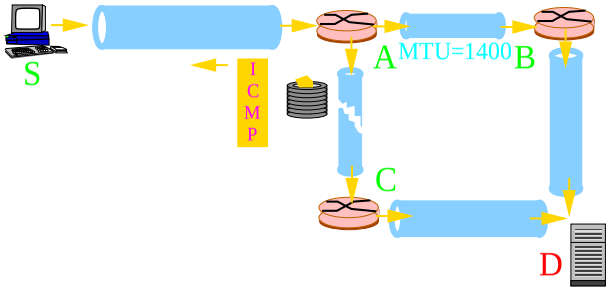
More PMTUD (2)

- S keeps sending with DF flag set
  - This is because...



More PMTUD (3)

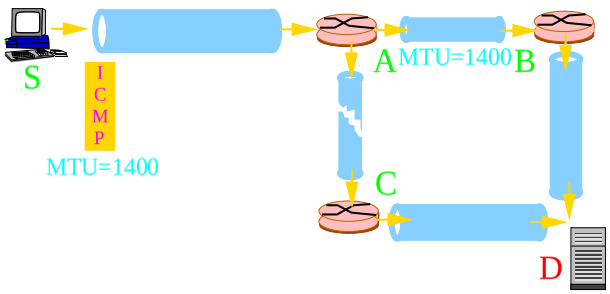
- A sends ICMP message
  - packet was discarded



  - This tells S that MTU of Path is now 1400

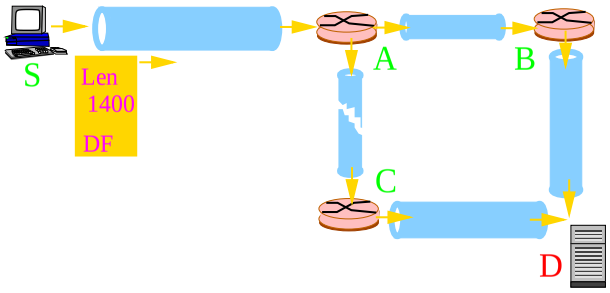
More PMTUD (3)

- A sends ICMP message
  - packet was discarded



More PMTUD (4)

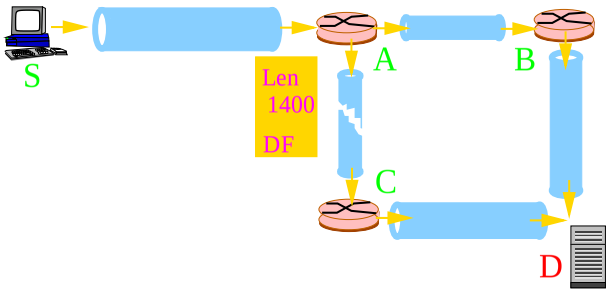
- S updates its record of the Path MTU to D
  - And ensures future packets
    - are no bigger than 1400 bytes



  - This allows packets from S to reach D
    - without fragmentation

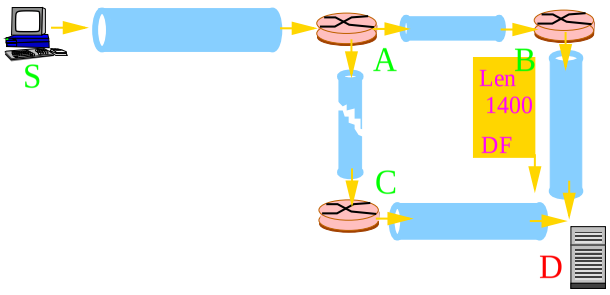
More PMTUD (4)

- S updates its record of the Path MTU to D
  - And ensures future packets
    - are no bigger than 1400 bytes



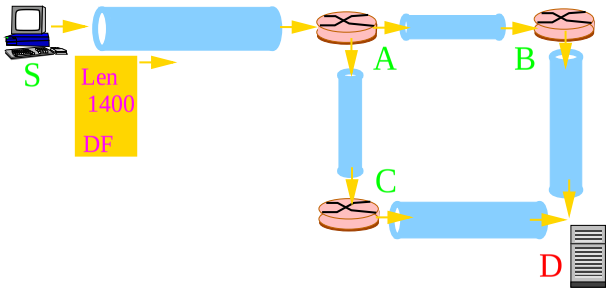
More PMTUD (4)

- S updates its record of the Path MTU to D
  - And ensures future packets
    - are no bigger than 1400 bytes



More PMTUD (5)

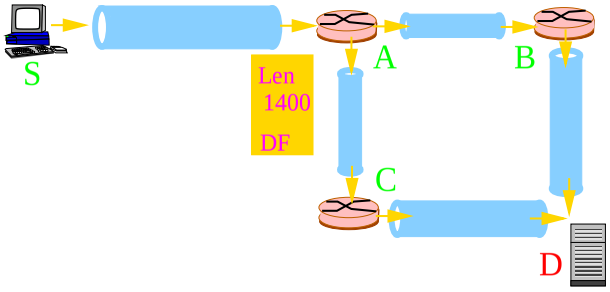
- Notice that when the A to C link is repaired
  - S still sends packets 1400 bytes



  - Even though path now allows 1500 bytes
  - There is no Packet too small ICMP message!

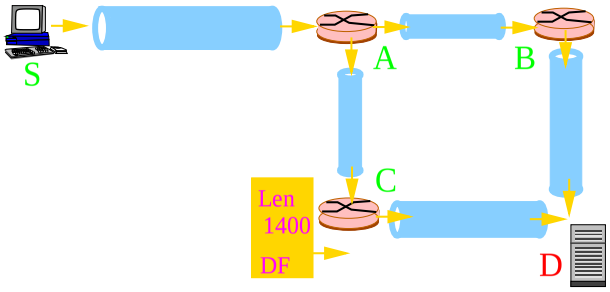
More PMTUD (5)

- Notice that when the A to C link is repaired
  - S still sends packets 1400 bytes



More PMTUD (5)

- Notice that when the A to C link is repaired
  - S still sends packets 1400 bytes



Final PMTUD

- To allow for possibility
  - that PMTU has increased
- Sender occasionally
  - sends slightly bigger packet
- If that fails
    - ICMP Too Big is returned
  - Then nothing has changed
    - Determined PMTU remains as found earlier
      - Or as rediscovered now
- If bigger packet reaches D
  - Then PMTU has increased
    - S remembers bigger packets work
    - Tries again with even bigger packet
      - Until local link MTU reached
      - Or until ICMP is returned
  - Now the new PMTU is known