Problemen met IPv4-fragmentatie, MTU, MSS en PMTUD oplossen met GRE en IPsec

Inhoud

Inleiding

In dit document wordt beschreven hoe IPv4 Fragmentation and Path Maximum Transmission Unit Discovery (PMTUD) werkt.

Achtergrondinformatie

Ook worden scenario's besproken die het gedrag van PMTUD wanneer gecombineerd met verschillende combinaties van IPv4 tunnels impliceren.

IPv4-fragmentatie en opnieuw samenstellen van datagrammen

Hoewel de maximale lengte van een IPv4-datagram 65535 is, dwingen de meeste transmissielinks een lagere limiet voor maximale pakketlengte af: de MTU. De waarde MTU hangt af van de transmissielink.

Het ontwerp van IPv4 past MTU verschillen aan omdat het routers toestaat om IPv4 datagrammen zonodig te fragmenteren.

Het ontvangststation is verantwoordelijk voor de hermontage van de fragmenten in het originele IPv4 datagram van volledige grootte.

IPv4 fragmentatie breekt een datagram in stukken die later opnieuw worden geassembleerd.

De IPv4 bron, bestemming, identificatie, totale lengte, en fragment offset velden, samen met "meer fragmenten" (MF) en "niet fragment" (DF) vlaggen in de IPv4 header, worden gebruikt voor IPv4 fragmentatie en herassemblage.

Zie RFC 791 voor meer informatie over de technische achtergrond van IPv4-fragmentatie en het opnieuw samenstellen van het datagram.

Deze afbeelding toont de indeling van een IPv4-header.

Original IP Datagram

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0	345	5140	0	0	0

IP Fragments (Ethernet)

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0-0	345	1500	0	1	0
0-1	345	1500	0	1	185
0-2	345	1500	0	1	370
0-3	345	700	0	0	555

De identificatie is 16 bits en is een waarde die wordt toegewezen door de afzender van een IPv4-datagram. Dit helpt bij de hermontage van de fragmenten van een datagram.

De fragment-offset is 13 bits en geeft aan waar een fragment hoort in het oorspronkelijke IPv4-datagram. Deze waarde is een veelvoud van 8 bytes.

Er zijn 3 bits voor besturingsvlaggen in het vlaggenveld van de IPv4-header. Het "do not fragment" (DF)-bit bepaalt of een pakket al dan niet gefragmenteerd mag worden.

Bit 0 is gereserveerd en wordt altijd op 0 ingesteld.

Bit 1 is de DF bit (0 = "kan fragment", 1 = "niet fragment fragmenteren").

Bit 2 is de MF-bit (0 = â€˜laatste fragmentâ€™, 1 = â€˜meer fragmentenâ€™).

Waarde	**Bit 0 gereserveerd**	**Bit 1 DF**	**Bit 2 MF**
0	0	Mag wel	Laatste
1	0	Mag niet	Meer

Als de lengten van de IPv4 fragmenten worden toegevoegd, overschrijdt de waarde de originele IPv4 datagramlengte door 60.

Dat komt doordat de totale lengte met 60 bytes is toegenomen omdat drie aanvullende IPv4-headers zijn gemaakt: één voor elk fragment na het eerste fragment.

Het eerste fragment heeft een offset van 0, de lengte van dit fragment is 1500; dit omvat 20 bytes voor de enigszins aangepaste oorspronkelijke IPv4-header.

Het tweede fragment heeft een offset van 185 (185 x 8 = 1480); het gegevensgedeelte van dit fragment begint met 1480 bytes in het oorspronkelijke IPv4-datagram.

De lengte van dit fragment is 1500; dit omvat de extra IPv4-header die voor dit fragment is gemaakt.

Het derde fragment heeft een offset van 370 (370 x 8 = 2960); het gegevensgedeelte van dit fragment begint met 2960 bytes in het oorspronkelijke IPv4-datagram.

De lengte van dit fragment is 1500; dit omvat de extra IPv4-header die voor dit fragment is gemaakt.

Het vierde fragment heeft een offset van 555 (555 x 8 = 4440). Het datagedeelte van dit fragment begint dus na 4440 bytes in het oorspronkelijke IPv4-datagram.

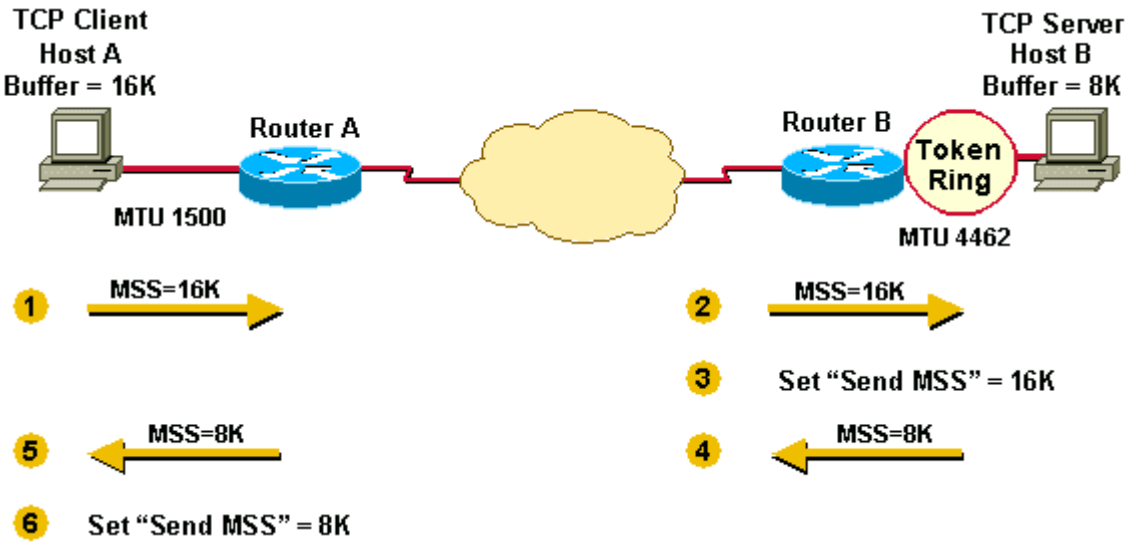
De lengte van dit fragment is 700 bytes. Hiertoe behoort de extra IPv4-header die voor dit fragment is gemaakt.

De grootte van het oorspronkelijke IPv4-datagram kan pas worden bepaald wanneer het laatste fragment is ontvangen.

De fragment-offset in het laatste fragment (555) levert een data-offset op van 4440 bytes in het oorspronkelijke IPv4-datagram.

De som van de data bytes van het laatste fragment (680 = 700 - 20) levert 5120 bytes op, wat het gegevensgedeelte is van het oorspronkelijke IPv4 datagram.

De toevoeging van 20 bytes voor een IPv4-header is gelijk aan de grootte van het oorspronkelijke IPv4 datagram (440 + 680 + 20 = 5140) zoals weergegeven in de afbeeldingen.



Problemen met IPv4-fragmentatie

IPv4 fragmentatie resulteert in een kleine toename in CPU en geheugen overhead om een IPv4 datagram te fragmenteren. Dit is waar voor de afzender en voor een router in de weg tussen een afzender en een ontvanger.

De creatie van fragmenten impliceert de verwezenlijking van fragmentkopballen en kopieert het originele datagram in de fragmenten.

Dit wordt efficiënt gedaan omdat de informatie die nodig is om de fragmenten te maken onmiddellijk beschikbaar is.

Fragmentatie leidt tot meer overhead voor de ontvanger bij het opnieuw samenstellen van het datagram aan de hand van de fragmenten, omdat de ontvanger geheugen moet toewijzen voor de gestuurde fragmenten en deze weer tot één datagram moet samenstellen nadat alle fragmenten zijn ontvangen.

De hermontage op een host wordt niet als een probleem gezien omdat de host de tijd en het geheugen heeft om aan deze taak te besteden.

De hermontage is echter inefficiënt op een router wiens primaire taak is om pakketten zo snel mogelijk door te sturen.

Een router is niet ontworpen om pakketten gedurende enige tijd vast te houden.

Een router die de herassemblage doet kiest de grootste beschikbare buffer (18K), omdat het geen manier heeft om de grootte van het originele IPv4 pakket te bepalen tot het laatste fragment wordt ontvangen.

Een ander fragmentatieprobleem heeft te maken met hoe er met afgewezen fragmenten wordt omgegaan.

Als één fragment van een IPv4 datagram wordt gelaten vallen, dan moet het volledige originele IPv4 datagram aanwezig zijn en het is ook gefragmenteerd.

Dit is te zien met Network File System (NFS). NFS heeft een lees- en schrijfblokgrootte van 8192.

Daarom is een NFS IPv4/UDP-datagram ongeveer 8500 bytes (inclusief NFS-, UDP- en IPv4-headers).

Een verzendende post die met een Ethernet (MTU 1500) wordt verbonden moet het datagram van 8500 bytes in zes (6) stukken fragmenteren; Vijf (5) 1500 bytesfragmenten en één (1) 1100 bytesfragment.

Als om het even welke zes fragmenten wegens een verstopte verbinding worden gelaten vallen, moet het volledige originele datagram worden opnieuw gebracht over. Dit resulteert in nog zes fragmenten die moeten worden gecreëerd.

Als deze link één op de zes pakketten verzwakt, is de kans klein dat NFS-gegevens via deze link worden overgebracht, omdat ten minste één IPv4-fragment van elk NFS 8500-byte origineel IPv4-datagram wordt verwijderd.

Firewalls die pakketten filteren of manipuleren op basis van Layer 4 (L4) door Layer 7 (L7)-informatie hebben problemen met de juiste verwerking van IPv4-fragmenten.

Als de IPv4-fragmenten niet goed werken, blokkeert een firewall de niet-initiële fragmenten omdat deze niet de informatie hebben die overeenkomt met het pakketfilter.

Dit betekent dat het oorspronkelijke IPv4 datagram niet kon worden geassembleerd door de ontvangende host.

Als de firewall zo is geconfigureerd dat niet-initiële fragmenten met onvoldoende informatie goed op het filter kunnen worden afgestemd, is een niet-initiële fragmentaanval via de firewall mogelijk.

Netwerkapparaten zoals Content Switch Engines direct pakketten gebaseerd op L4 tot L7 informatie, en als een pakket meerdere fragmenten overspant, dan heeft het apparaat problemen met het afdwingen van zijn beleid.

IPv4-fragmentatie vermijden: hoe TCP MSS werkt

De Transmission Control Protocol (TCP) Maximum Segment Size (MSS) definieert de maximale hoeveelheid gegevens die een host in één TCP/IPv4-datagram accepteert.

Dit TCP/IPv4 datagram is mogelijk gefragmenteerd in de IPv4-laag. De MSS-waarde wordt alleen als optie in de TCP-header verzonden in TCP SYN-segmenten.

Elke kant van een TCP-verbinding meldt zijn MSS-waarde aan de andere kant. De MSS waarde wordt *niet* tussen hosts onderhandeld.

De verzendende host moet de hoeveelheid data in één TCP-segment beperken tot een waarde die lager is dan of gelijk is aan de MSS-waarde die door de ontvangende host is gemeld.

Oorspronkelijk gaf de MSS-waarde aan hoe groot de buffer was (groter dan of gelijk aan 65496 bytes) die aan een ontvangend station werd toegekend om de TCP-data in één IPv4-datagram te kunnen opslaan.

MSS was het maximale gegevenssegment dat de TCP-ontvanger wilde accepteren. Dit TCP-segment kan zo groot zijn als 64K en gefragmenteerd op de IPv4-laag om naar de ontvangende host te worden verzonden.

De ontvangende host stelde het IPv4-datagram opnieuw samen voordat het volledige TCP-segment werd overgedragen aan de TCP-laag.

Hoe MSS-waarden worden ingesteld en gebruikt om de grootte van TCP-segment en IPv4-datagrammen te beperken.

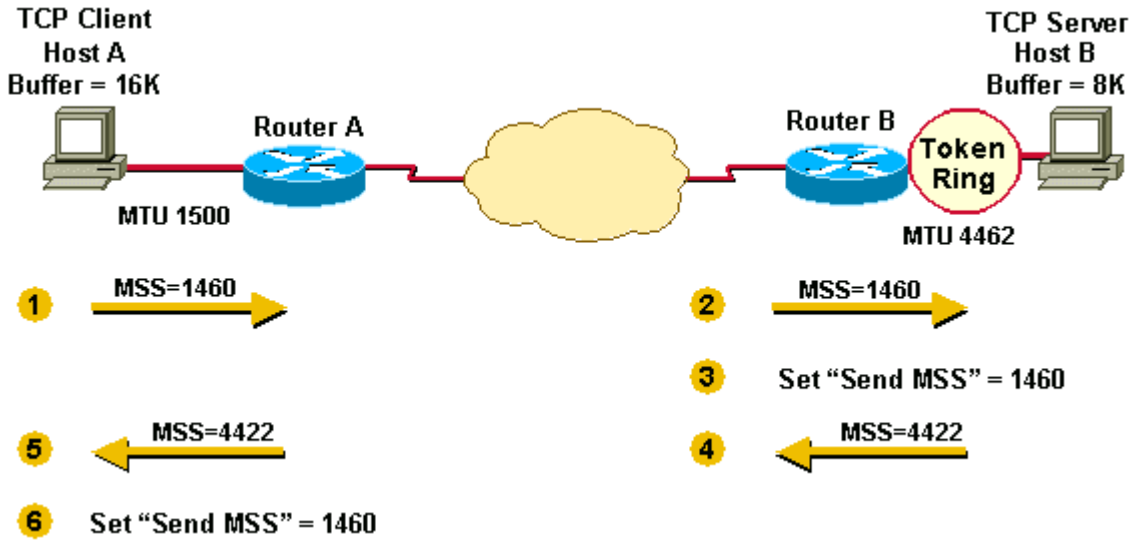
Voorbeeld 1 illustreert de manier waarop MSS voor het eerst werd geïmplementeerd.

Host A heeft een buffer van 16K bytes en host B van 8K bytes. De hosts verzenden en ontvangen hun MSS-waarden en passen hun MSS-waarde voor verzenden van data op elkaar af.

Host A en Host B moeten de IPv4-datagrammen fragmenteren die groter zijn dan de interface MTU, maar nog minder dan de Send MSS omdat de TCP-stack 16K of 8K bytes van gegevens door de stack naar IPv4 doorgeeft.

In het geval van Host B zijn pakketten gefragmenteerd om op Token Ring LAN te geraken en opnieuw op Ethernet LAN te geraken.

Voorbeeld 1



1. Host A stuurt zijn MSS-waarde van 16K bytes naar host B.
2. Host B ontvangt de MSS-waarde van 16K bytes afkomstig van host A.
3. Host B stelt zijn MSS-waarde voor verzenden in op 16K bytes.
4. Host B stuurt zijn MSS-waarde van 8K bytes naar host A.
5. Host A ontvangt de MSS-waarde van 8K bytes afkomstig van host B.
6. Host A stelt zijn MSS-waarde voor verzenden in op 8K bytes.

Om IPv4 fragmentatie op de eindpunten van de TCP-verbinding te voorkomen, is de selectie van de MSS-waarde gewijzigd in de minimale buffergrootte en de MTU van de uitgaande interface (- 40).

MSS-getallen zijn 40 bytes kleiner dan MTU-getallen omdat MSS (de TCP-gegevensgrootte) geen 20-bytes IPv4-header en de 20-bytes TCP-header omvat.

MSS is gebaseerd op standaard headergrootte; de afzenderstack moet de juiste waarden voor de IPv4 header aftrekken en de TCP header hangt af van welke TCP- of IPv4-opties worden gebruikt.

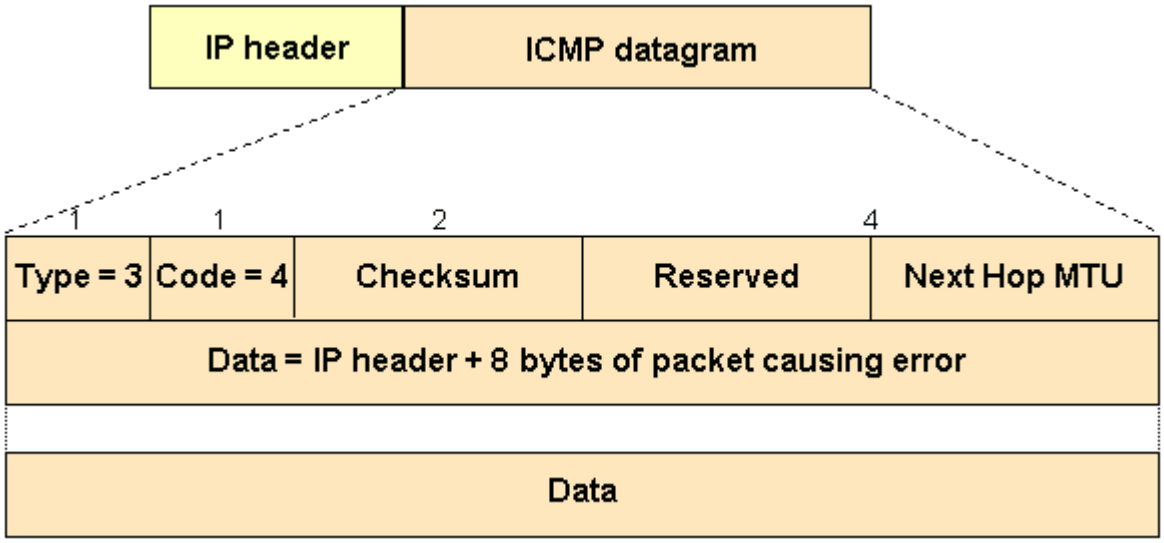
MSS werkt momenteel op een manier waarbij elke host eerst zijn uitgaande interface MTU met zijn eigen buffer vergelijkt en de laagste waarde kiest als de MSS om te verzenden.

De hosts vergelijken vervolgens de ontvangen MSS-grootte met hun eigen interface MTU en kiezen opnieuw de laagste van de twee waarden.

Voorbeeld 2 illustreert deze extra stap die door de afzender wordt genomen om fragmentatie op de lokale en externe bedradingen te voorkomen.

De MTU van de uitgaande interface wordt door elke host meegenomen voordat de hosts elkaar hun MSS-waarden sturen. Dit helpt fragmentatie te voorkomen.

Voorbeeld 2



1. Host A vergelijkt zijn MSS-buffer (16K bytes) en MTU (1500 â€“ 40 = 1460) en gebruikt de lagere waarde (1460) als de MSS-waarde voor verzending naar host B.
2. Host B ontvangt de Send MSS (1460) van Host A en vergelijkt deze met de waarde van zijn uitgaande interface MTU - 40 (4422).
3. Host B stelt de lagere waarde (1460) in als de MSS-waarde voor het verzenden van IPv4-datagrammen naar host A.
4. Host B vergelijkt zijn MSS-buffer (8K bytes) en MTU (4462 â€“ 40 = 4422) en gebruikt 4422 als de MSS-waarde voor verzending naar host A.
5. Host A ontvangt de Send MSS (4422) van Host B en vergelijkt deze met de waarde van zijn uitgaande interface MTU -40 (1460).
6. Host A stelt de lagere waarde (1460) in als de MSS-waarde voor het verzenden van IPv4-datagrammen naar host B.

1460 is de waarde die door beide hosts wordt gekozen als de MSS-waarde voor verzenden naar elkaar. Vaak is de waarde voor verzenden MSS hetzelfde aan elk uiteinde van een TCP-verbinding.

In Voorbeeld 2, komt de fragmentatie niet op de eindpunten van een verbinding van TCP voor omdat beide uitgaande interface MTUs door de gastheren in rekening worden gebracht.

Pakketten worden nog steeds gefragmenteerd in het netwerk tussen router A en router B als ze een link tegenkomen met een lagere MTU dan die van de uitgaande interface van beide hosts.

Wat is PMTUD

TCP MSS richt fragmentatie op de twee eindpunten van een TCP verbinding, maar het behandelt geen gevallen waar er een kleinere MTU verbinding in het midden tussen deze twee eindpunten is.

PMTUD werd ontwikkeld om fragmentatie op het pad tussen de endpoints te vermijden. Het wordt gebruikt om dynamisch de laagste MTU langs de weg van een pakketbron aan zijn bestemming te bepalen.

> **Opmerking**: PMTUD wordt alleen ondersteund door TCP en UDP. Andere protocollen ondersteunen PMTUD niet. Als PMTUD op een host is ingeschakeld, hebben alle TCP- en UDP-pakketten van de host de DF-bitset.

Wanneer een host een volledig MSS-datapakket verzendt met de DF-bit, verlaagt de PMTUD de MSS-waarde voor verzenden voor de verbinding als informatie wordt ontvangen dat het pakket moet worden

gefragmenteerd.

Een host registreert de MTU-waarde voor een bestemming omdat er een host (/32)-ingang wordt gemaakt in de routeringstabel met deze MTU-waarde.

Als een router probeert een IPv4 datagram (met de DF bit-set) door te sturen naar een link met een lagere MTU dan de grootte van het pakket, laat de router het pakket vallen en geeft een ICMP-bericht (Internet Control Message Protocol) "Bestemming onbereikbaar" naar de IPv4 datagrambron terug met de code die "fragmentatie nodig en DF-set" (type 3, code 4) aangeeft.

Wanneer het bronstation het ICMP-bericht ontvangt, verlaagt het de verzend-MSS, en wanneer TCP het segment opnieuw overbrengt, gebruikt het de kleinere segmentgrootte.

Hier is een voorbeeld van een ICMP "fragmentatie nodig en DF set" bericht gezien op een router na de **debug ip icmp** De opdracht is ingeschakeld:

```
ICMP: dst (10.10.10.10) frag. needed and DF set
unreachable sent to 10.1.1.1
```

Dit diagram toont de indeling van de ICMP-header van het bericht â€˜Bestemming onbereikbaarâ€™ met de code voor â€˜Fragmentatie nodig maar DF-bit ingesteldâ€™.

```
Plateau    MTU    Comments                      Reference
------     ---    --------                      ---------
           65535  Official maximum MTU          RFC 791
           65535  Hyperchannel                  RFC 1044
65535
32000             Just in case
           17914  16Mb IBM Token Ring           ref. [6]
17914
           8166   IEEE 802.4                    RFC 1042
8166
           4464   IEEE 802.5 (4Mb max)          RFC 1042
           4352   FDDI (Revised)                RFC 1188
4352 (1%)
           2048   Wideband Network              RFC 907
           2002   IEEE 802.5 (4Mb recommended)  RFC 1042
2002 (2%)
           1536   Exp. Ethernet Nets            RFC 895
           1500   Ethernet Networks             RFC 894
           1500   Point-to-Point (default)      RFC 1134
           1492   IEEE 802.3                    RFC 1042
1492 (3%)
           1006   SLIP                          RFC 1055
           1006   ARPANET                       BBN 1822
1006
           576    X.25 Networks                 RFC 877
           544    DEC IP Portal                 ref. [10]
           512    NETBIOS                       RFC 1088
           508    IEEE 802/Source-Rt Bridge     RFC 1042
           508    ARCNET                        RFC 1051
508 (13%)
           296    Point-to-Point (low delay)    RFC 1144
296
68                Official minimum MTU          RFC 791
```

Vanaf RFC 1191 moet een router die een ICMP-bericht stuurt met â€˜Fragmentatie nodig maar DF-bit ingesteldâ€™ de MTU van het netwerk van de volgende hop bevatten in de lage 16 bits van het aanvullende ICMP-headerveld dat in ICMP-specificatie RFC 792 het label â€˜ongebruiktâ€™ heeft.

Vroegere implementaties van RFC 1191 boden geen MTU-informatie voor de volgende hop. Zelfs wanneer de informatie wel werd geboden, werd deze door sommige hosts genegeerd.

Voor dit geval bevat RFC 1191 ook een tabel met de voorgestelde waarden waarmee de MTU tijdens PMTUD wordt verlaagd.

Het wordt gebruikt door hosts om sneller tot een redelijke waarde te komen voor de verstuurde MSS en zoals in dit voorbeeld wordt getoond.

PMTUD wordt voortdurend op alle pakketten uitgevoerd omdat het pad tussen afzender en ontvanger dynamisch kan veranderen.

Telkens wanneer een afzender ICMP-berichten ontvangt die niet kunnen worden gefragmenteerd, wordt de routerinformatie bijgewerkt (waar de PMTUD wordt opgeslagen).

Tijdens PMTUD kunnen twee dingen gebeuren:

1. Het pakket kan helemaal tot aan de ontvanger worden gebracht zonder te worden gefragmenteerd.

> **Opmerking**: een router kan de CPU beschermen tegen DoS-aanvallen door het aantal onbereikbare ICMP-berichten te beperken tot twee per seconde. Daarom, in deze context, als u een netwerkscenario hebt waarin u verwacht dat de router met meer dan twee ICMP berichten (type = 3, code = 4) per seconde (kunnen verschillende gastheren zijn) zou moeten antwoorden, schakel de verstikking van ICMP berichten met uit **no ip icmp rate-limit unreachable [df] interface** uit.

---

2. De afzender krijgt ICMP "Cannot Fragment" berichten van hop langs het pad naar de ontvanger.

PMTUD vindt onafhankelijk plaats voor beide richtingen van een TCP-stroom.

Er zijn gevallen waar PMTUD in de ene richting van een stroom een van de eindstations activeert om te verlagen verzenden MSS en het andere eindstation houdt het origineel verzenden MSS omdat het nooit een IPv4 datagram groot genoeg heeft verzonden om PMTUD te activeren.

Een voorbeeld is de HTTP-verbinding die in voorbeeld 3 is afgebeeld. De TCP-client verzendt kleine pakketten en de server verzendt grote pakketten.

In dit geval worden alleen de grote pakketten vanaf de server (groter dan 576 bytes) geactiveerd als PMTUD.

De pakketten van de cliënt zijn klein (minder dan 576 bytes) en brengen geen PMTUD teweeg omdat zij geen fragmentatie vereisen om over de 576 MTU verbinding te krijgen.

Voorbeeld 3

Voorbeeld 4 toont een asymmetrisch routeringsvoorbeeld waar een van de paden een kleinere minimum MTU heeft dan de andere.

Asymmetrische routing vindt plaats wanneer verschillende paden worden gebruikt om data te verzenden en ontvangen tussen twee endpoints.

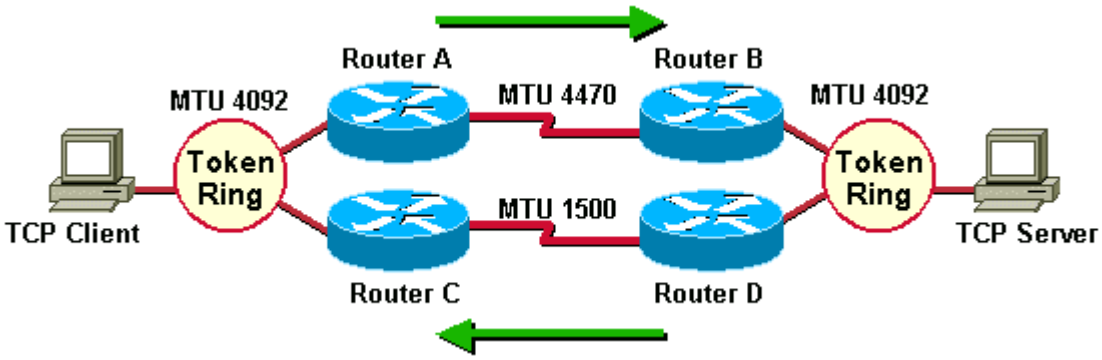
In dit voorbeeld, leidt PMTUD tot het verlagen van verzend MSS slechts in één richting van een stroom van TCP.

Het verkeer van de TCP-client naar de server gaat via router A en router B. Het retourverkeer van de server naar de client gaat via router D en router C.

Wanneer de TCP server pakketten naar de client verzendt, activeert PMTUD de server om het verzenden van MSS te verlagen omdat router D de 4092 byte pakketten moet fragmenteren voordat het ze naar router C kan verzenden.

Omgekeerd, ontvangt de client nooit een ICMP "Bestemming Onbereikbaar" bericht met de code die "fragmentatie nodig en PDF-set" aangeeft, omdat router A geen pakketten hoeft te fragmenteren wanneer deze via router B naar de server worden verzonden.

Voorbeeld 4



Opmerking: de opdracht **IP TCP-pad-mtu-detectie** wordt gebruikt om TCP-MTU-paddetectie in te schakelen voor TCP-verbindingen die worden gestart door routers (BGP en Telnet bijvoorbeeld).

Problemen met PMTUD

Dit zijn dingen die PMTUD kunnen doorbreken.

- Een router laat een pakket vallen en verzendt geen ICMP-bericht. (Ongebruikelijk)
- Een router genereert en verstuurt een ICMP-bericht, maar het ICMP-bericht wordt geblokkeerd door een router of firewall tussen deze router en de afzender. (Gebruikelijk)
- Een router genereert en verstuurt een ICMP-bericht, maar de afzender negeert het bericht. (Ongebruikelijk)

De eerste en laatste van de drie kogels hier zijn meestal het gevolg van een fout, maar de middelste kogel beschrijft een algemeen probleem.

Die die ICMP pakketfilters uitvoeren neigen ertoe om alle ICMP berichttypes te blokkeren eerder dan slechts bepaalde ICMP berichttypes te blokkeren.

Het is mogelijk voor pakketfilter om alle ICMP berichttypes behalve die te blokkeren die "onbereikbaar"of "tijd-overschreden zijn."

Het succes of falen van PMTUD hangt af van ICMP onbereikbare berichten die door naar de afzender van een TCP/IPv4 pakket.

ICMP-berichten â€˜tijd verstrekenâ€™ zijn van belang voor andere IPv4-problemen.

Hier wordt een voorbeeld getoond van een dergelijk pakketfilter, geïmplementeerd op een router.

```
access-list 101 permit icmp any any unreachable
access-list 101 permit icmp any any time-exceeded
access-list 101 deny icmp any any
access-list 101 permit ip any any
```

Er zijn andere technieken die kunnen worden gebruikt om het probleem van een volledig geblokkeerde ICMP te verlichten.

- Schakel de DF-bit op de router uit en sta fragmentatie toe. (Dit is echter geen goed idee.) zie Problemen met IPv4-fragmentatie voor meer informatie).
- De TCP MSS-optiewaarde MSS manipuleren met de interfaceopdracht **ip tcp adjust-mss <500-1460>.**

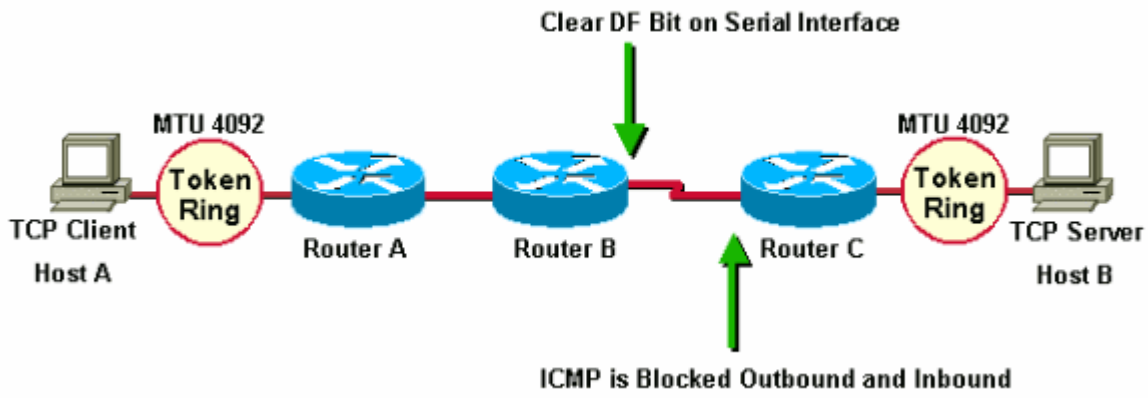
In het volgende voorbeeld, router A en router B zijn in het zelfde administratieve domein. Router C is ontoegankelijk en blokkeert ICMP, waardoor PMTUD niet werkt.

Een tijdelijke oplossing is om de DF-bit in beide richtingen op router B te wissen om fragmentatie toe te staan. Dit kan worden bewerkstelligd via beleidsroutering.

De syntaxis voor het wissen van de DF-bit is beschikbaar in Cisco IOS®-softwarerelease 12.1(6) en hoger.

```
interface serial0
...
ip policy route-map clear-df-bit
route-map clear-df-bit permit 10
    match ip address 111
    set ip df 0

access-list 111 permit tcp any any
```



Een andere optie is om de TCP MSS-waarde te wijzigen bij SYN-pakketten die de router passeren (beschikbaar in Cisco IOS®-versie 12.2(4)T en hoger).

Hierdoor wordt de MSS-optiewaarde in het TCP/SYN-pakket verlaagd, zodat deze kleiner is dan de waarde (1460) in het **ip tcp adjust-mss**uit.

Het resultaat is dat de TCP-afzender segmenten niet groter dan deze waarde verstuurt.

De IPv4-pakketgrootte is 40 bytes groter (1500) dan de MSS-waarde (1460 bytes) om rekening te houden met de TCP-header (20 bytes) en de IPv4-header (20 bytes).

U kunt de MSS van TCP/SYN-pakketten aanpassen met de **ip tcp adjust-mss** uit. Deze syntaxis vermindert de MSS-waarde op TCP-segmenten tot 1460.

Deze opdracht heeft betrekking op zowel inkomend als uitgaand verkeer op interface serial0.

```
int s0
ip tcp adjust-mss 1460
```

Problemen met IPv4-fragmentatie komen vaker voor als gevolg van bredere implementatie van IPv4-tunnels.

Tunnels veroorzaken meer fragmentatie omdat de tunnelinsluiting "overhead" aan de grootte van een pakket toevoegt.

Bijvoorbeeld, voegt de toevoeging van Generieke Router Encapsulation (GRE) 24 bytes aan een pakket toe, en na deze verhoging, moet het pakket worden gefragmenteerd omdat het groter is dan uitgaande MTU.

Gangbare netwerktopologieën die PMTUD vereisen

PMTUD is vereist in netwerksituaties waarbij tussenliggende links lagere MTU-waarden hebben dan de MTU van de eindlinks. Enkele gangbare redenen voor het bestaan van deze links met een lagere MTU zijn:

- Via Token Ring (of FDDI) verbonden eindhosts met een Ethernet-verbinding tussen de hosts. De MTUâ€™s voor Token Ring (of FDDI) aan de uiteinden zijn groter dan de Ethernet-MTU in het midden.
- PPPoE (vaak gebruikt met ADSL) heeft 8 bytes nodig voor de header. Hierdoor wordt de effectieve Ethernet-MTU verlaagd tot 1492 (1500 â€“ 8).

Tunnelprotocollen zoals GRE, IPv4sec en L2TP hebben ook ruimte nodig voor hun respectievelijke headers en trailers. Dit vermindert ook de effectieve MTU van de uitgaande interface.

Tunnel

Een tunnel is een logische interface op een Cisco-router die een manier biedt om passenger-pakketten in te kapselen in een transportprotocol.

Deze architectuur is ontwikkeld om services te bieden voor de implementatie van een point-to-point inkapselingsschema. Tunnelinterfaces hebben deze drie primaire componenten:

- Passenger-protocol (AppleTalk, Banyan VINES, CLNS, DECnet, IPv4 of IPX)
- Carrier-protocol â€“ één van de volgende inkapselingsprotocollen:
    - GRE - Cisco multiprotocol transportprotocol. Zie RFC 2784 en RFC 1701 voor meer informatie.
    - IPv4 in IPv4-tunnels â€“ zie RFC 2003 voor meer informatie.
- Transportprotocol â€“ het protocol dat wordt gebruikt voor het overdragen van het ingekapselde protocol.

De pakketten in deze sectie illustreren de concepten van IPv4-tunneling waarbij GRE het inkapselingsprotocol is en IPv4 het transportprotocol.

IPv4 is ook het passenger-protocol. In dit geval is IPv4 zowel het transport- als het passenger-protocol.

Normaal pakket

IPv4	TCP	Telnet

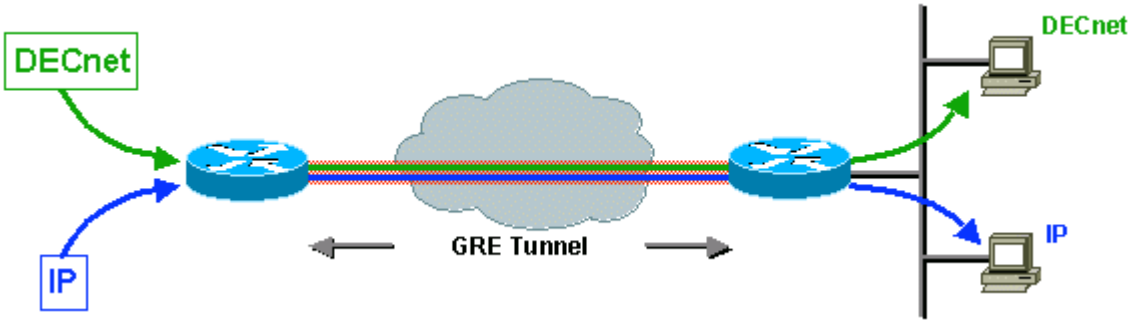
Tunnelpakket

IPv4	GRE	IPv4	TCP	Telnet

- IPv4 is het transportprotocol.
- GRE is het inkapselingsprotocol.
- IPv4 is het passenger-protocol.

In het volgende voorbeeld wordt de inkapseling getoond met IPv4 en DECnet als passenger-protocollen en GRE als carrier.

Dit illustreert de mogelijkheid dat de dragerprotocollen meerdere passagiersprotocollen zoals getoond in het beeld inkapselen.



Een netwerkbeheerder overweegt het tunnelen in een situatie waar er twee discontiguous niet-IPv4 netwerken zijn die door een IPv4 backbone worden gescheiden.

Als de discontiguous netwerken DECnet uitvoeren, kan de beheerder ervoor kiezen om ze samen (of niet om) te verbinden door DECnet in de backbone te configureren.

De beheerder wil niet toestaan dat DECnet Routing backbone bandbreedte verbruikt omdat dit de prestaties van het IPv4 netwerk zou kunnen storen.

Een haalbaar alternatief is tunneling van DECnet via de IPv4-backbone. De tunneloplossing kapselt de DECnet-pakketten in IPv4 in en stuurt ze over de backbone naar het tunneleindpunt waar de inkapseling wordt verwijderd en de DECnet-pakketten via DECnet naar hun bestemming worden verstuurd.

Er zijn voordelen om verkeer binnen een ander protocol in te kapselen:

- De endpoints gebruiken privé-adressen (RFC 1918) en de backbone ondersteunt geen routing van deze adressen.
- Virtual Private Networks (VPNâ€™s) zijn toegestaan op WANâ€™s of het internet.
- Opgedeelde multiprotocol netwerken worden samengevoegd via een backbone met één protocol.
- Verkeer wordt versleuteld via de backbone of het internet.

Hierna wordt IPv4 gebruikt als het passagiersprotocol en IPv4 als het transportprotocol.

Overwegingen voor tunnelinterfaces

De volgende overwegingen zijn van toepassing bij tunneling.

- Fast switching van GRE-tunnels werd geïntroduceerd in Cisco IOS® versie 11.1 en CEF-switching in versie 12.0.
- CEF-switching voor multipoint GRE-tunnels werd geïntroduceerd in versie 12.2(8)T.
- Inkapseling en ontkapseling bij tunnel-endpoints waren langzame processen in eerdere versies van Cisco IOS® wanneer alleen proces-switching werd ondersteund.
- Er zijn beveiligings- en topologieproblemen bij de tunneling van pakketten. Tunnels kunnen toegangscontrolelijsten (ACLâ€™s) en firewalls omzeilen.
- Als u door een firewall tunnelt, passeert u het passagiersprotocol dat wordt getunneld. Het is daarom raadzaam firewallfunctionaliteit op te nemen aan de tunnel-endpoints om beleid af te dwingen op de passenger-protocollen.

- Tunneling leidt tot problemen met vervoerprotocollen die tijdopnemers (bijvoorbeeld, DECnet) wegens verhoogde latentie hebben beperkt.
- Tunneling over omgevingen met verschillende snelheidskoppelingen, zoals snelle FDDI-ringen en door langzame 9600 Gbps telefoonlijnen, introduceert problemen bij het opnieuw ordenen van pakketten. Sommige passenger-protocollen werken slecht op netwerken met verschillende media.
- Point-to-point tunnels verbruiken bandbreedte op een fysieke link. Over meerdere point-to-point tunnels heeft elke tunnelinterface een bandbreedte en de fysieke interface waarover de tunnel loopt heeft een bandbreedte. Stel de tunnelbandbreedte bijvoorbeeld in op 100 Kb als er 100 tunnels over een 10 Mb-verbinding lopen. De standaardbandbreedte voor een tunnel is 9 Kb.
- Routing-protocollen geven de voorkeur aan een tunnel boven een echte link, omdat de tunnel op bedrieglijke wijze een één-hop-link met de laagste-kosten-route lijkt te zijn, hoewel er meer hop mee gemoeid is en het daardoor duurder is dan een ander pad. Dit wordt verlicht met juiste configuratie van het routeringsprotocol. Overweeg het uitvoeren van een ander routingprotocol via de tunnelinterface dan het routingprotocol dat op de fysieke interface wordt uitgevoerd.
- Problemen met recursieve routing worden vermeden door de juiste statische routes naar de tunnelbestemming te configureren. Er is sprake van een recursieve route wanneer het beste pad naar de tunnelbestemming via de tunnel zelf loopt. In deze situatie is de tunnelinterface niet stabiel. Deze fout wordt gezien wanneer er een recursief routeringsprobleem is.

```
%TUN-RECURDOWN Interface Tunnel 0
temporarily disabled due to recursive routing
```

Router als PMTUD-deelnemer bij endpoint van tunnel

De router heeft twee verschillende PMTUD-rollen wanneer de route het endpoint van een tunnel is.

- In de eerste rol is de router het doorsturende apparaat van een hostpakket. Bij PMTUD-verwerking moet de router de DF-bit en pakketgrootte van het oorspronkelijke datapakket controleren en waar nodig gepaste actie ondernemen.
- De tweede rol wordt vervuld nadat de router het oorspronkelijke IPv4-pakket heeft ingekapseld in het tunnelpakket. In deze fase fungeert de router meer als host voor PMTUD en het IPv4-tunnelpakket.

Wanneer de router in de eerste rol (een router die IPv4 pakketten door:sturen) handelt, komt deze rol in spelen alvorens de router het gastheer IPv4 pakket binnen het tunnelpakket inkapselt.

Als de router deelneemt als doorgeefster van een hostpakket, voltooit het deze acties:

- Controleren of de DF-bit is ingesteld.
- Controleren welke pakketgrootte de tunnel kan verwerken.
- Fragment (als pakket te groot is en DF-bit niet is ingesteld), fragmenten inkapselen en verzenden; of
- Het pakket afwijzen (als pakket te groot is en de DF-bit is ingesteld) en een ICMP-bericht naar de verzender sturen.
- Inkapselen (als pakket niet te groot is) en verzenden.

Er kan dus worden gekozen tussen inkapseling en vervolgens fragmentatie (verzending van twee inkapselfragmenten) of fragmentatie en vervolgens inkapseling (verzending van twee ingekapselde fragmenten).

Twee voorbeelden die de interactie van PMTUD en pakketten tonen die voorbeeldnetwerken doorkruisen worden in deze sectie gedetailleerd.

In het eerste voorbeeld wordt getoond wat er gebeurt met een pakket wanneer de router (bij de tunnelbron) de rol van doorsturende router vervult.

Om PMTUD te verwerken, moet de router het DF-bit en de pakketgrootte van het oorspronkelijke gegevenspakket controleren en de juiste actie ondernemen.

In dit voorbeeld is er sprake van GRE-inkapseling voor de tunnel. GRE doet fragmentatie voor inkapseling.

In latere voorbeelden vindt fragmentatie plaats na inkapseling.

In voorbeeld 1 is de DF-bit niet ingesteld (DF = 0) en is de IPv4-MTU van de GRE-tunnel 1476 (1500 â€“ 24).

Voorbeeld 1

1. De verzendende router (bij de tunnelbron) ontvangt een datagram van 1500 bytes met het DF-bit helder (DF = 0) van de verzendende host.

Dit datagram bestaat uit een IP-header van 20 bytes en een TCP-payload van 1480 bytes.

IPv4	1480 bytes TCP + data

2. Omdat het pakket te groot is voor de IPv4 MTU nadat de GRE-overhead (24 bytes) is toegevoegd, breekt de doorstuurrouter het datagram in twee fragmenten van 1476 (20 bytes IPv4 header + 1456 bytes IPv4 payload) en 44 bytes (20 bytes van IPv4 header + 24 bytes van IPv4 payload)

Nadat de GRE-insluiting is toegevoegd, is het pakket niet groter dan de uitgaande fysieke interface MTU.

IP_0	1456 bytes TCP + data
IP_1	24 bytes data

3. De router voor doorsturen voegt GRE-inkapseling, die een GRE-header van 4 bytes plus een IPv4-header van 20 bytes omvat, toe aan elk fragment van het oorspronkelijke IPv4-datagram.

Deze twee IPv4-datagrammen hebben nu lengtes van 1500 en 68 bytes en worden beschouwd als afzonderlijke IPv4-datagrammen, niet als fragmenten.

IPv4	GRE	IP_0	1456 bytes TCP + data
IPv4	GRE	IP_1	24 bytes data

4. De router van de tunnelbestemming verwijdert de inkapseling GRE uit elk fragment van het originele datagram, dat twee IPv4 fragmenten van lengtes 1476 en 24 bytes verlaat.

Deze IPv4-datagramfragmenten worden afzonderlijk door deze router doorgestuurd naar de ontvangende host.

IP_0	1456 bytes TCP + data

IP_1	24 bytes data

5. De ontvangende host brengt deze twee fragmenten weer samen in het oorspronkelijke datagram.

IPv4	1480 bytes TCP + data

Voorbeeld 2 schildert de rol van de door:sturen router in de context van een netwerktopologie af.

De router werkt in dezelfde rol van het doorsturen van router, maar dit keer wordt het DF bit ingesteld (DF = 1).

Voorbeeld 2

1. De verzendende router bij de tunnelbron ontvangt een datagram van 1500 bytes met DF = 1 van de verzendende host.

IPv4	1480 bytes TCP + data

2. Aangezien het DF-bit is ingesteld en de grootte van het datagram (1500 bytes) groter is dan de GRE-tunnel IPv4 MTU (1476), laat de router het datagram vallen en verstuurt een "ICMP fragmentatie nodig maar DF bit set"-bericht naar de bron van het datagram.

Het ICMP bericht waarschuwt de afzender dat de MTU 1476 is.

IPv4	ICMP MTU 1476

3. De verzendende host ontvangt het ICMP bericht en wanneer het de oorspronkelijke gegevens opnieuw verstuurt gebruikt het een 1476-byte IPv4 datagram.

IPv4	1456 bytes TCP + data

4. Deze IPv4 datagramlengte (1476 bytes) is nu gelijk in waarde aan de GRE-tunnel IPv4 MTU, zodat de router de GRE-insluiting aan het IPv4 datagram toevoegt.

IPv4	GRE	IPv4	1456 bytes TCP + data

5. De ontvangende router (op de tunnelbestemming) verwijdert de GRE-insluiting van het IPv4-datagram en stuurt deze naar de ontvangende host.

IPv4	1456 bytes TCP + data

Dit is wat er gebeurt wanneer de router in de tweede rol als verzendende gastheer met betrekking tot PMTUD en met betrekking tot het tunnel IPv4 pakket handelt.

Deze rol komt in spel nadat de router het originele IPv4 pakket binnen het tunnelpakket heeft ingekapseld.

> **Opmerking**: standaard voert een router geen PMTUD uit op de GRE-tunnelpakketten die worden gegenereerd. Het **tunnel path-mtu-discovery** Deze opdracht kan worden gebruikt om PMTUD in te schakelen voor GRE-IPv4-tunnelpakketten.

In voorbeeld 3 wordt beschreven wat er gebeurt wanneer de host IPv4-datagrammen verzendt die klein genoeg zijn om in de IPv4-MTU van de GRE-tunnelinterface te passen.

In dit geval kan de DF-bit wel of niet zijn ingesteld (1 of 0).

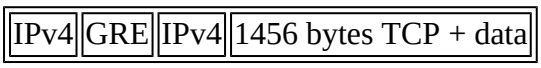
De GRE-tunnelinterface heeft niet de **tunnel path-mtu-discovery** opdracht geconfigureerd zodat de router geen PMTUD instelt op het GRE-IPv4-pakket.

Voorbeeld 3

1. De verzendende router bij de tunnelbron ontvangt een datagram van 1476 bytes van de verzendende host.

2. Deze router kapselt het 1476-byte IPv4 datagram in GRE in om een 1500-byte GRE IPv4 datagram te krijgen.

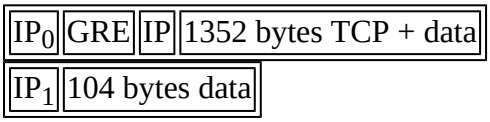
De DF-bit in de GRE IPv4-header wordt gewist (DF = 0). Deze router stuurt dit pakket vervolgens door naar de tunnelbestemming.



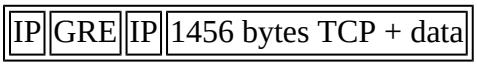
3. Stel dat er een router is tussen de tunnelbron en de bestemming met een koppeling MTU van 1400.

Deze router fragmenteert het tunnelpakket aangezien het DF-bit helder is (DF = 0).

Vergeet niet dat dit voorbeeld de buitenste IPv4 fragmenten, zodat de GRE, binnenste IPv4, en TCP-headers alleen in het eerste fragment verschijnen.



4. De router van de tunnelbestemming moet het GRE-tunnelpakket opnieuw assembleren.



5. Nadat het GRE-tunnelpakket opnieuw is geassembleerd, verwijdert de router de GRE IPv4-header en verstuurt het oorspronkelijke IPv4-datagram onderweg.

Voorbeeld 4 laat zien wat er gebeurt wanneer de router in de rol van een verzendende host optreedt met betrekking tot PMTUD en met betrekking tot het IPv4-pakket van de tunnel.

Ditmaal wordt het DF-bit ingesteld (DF = 1) in de oorspronkelijke IPv4-header en de **tunnel path-mtu-discovery** De opdracht is zo geconfigureerd dat de DF-bit van de interne IPv4-header naar de externe (GRE + IPv4) is gekopieerd.

Voorbeeld 4

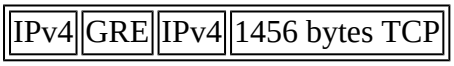
1. De verzendende router bij de tunnelbron ontvangt een datagram van 1476 bytes met DF = 1 van de verzendende host.

2. Deze router kapselt het 1476-byte IPv4 datagram in GRE in om een 1500-byte GRE IPv4 datagram te

krijgen.

Deze GRE IPv4 header heeft de DF bit set (DF = 1) omdat de originele IPv4 datagram de DF bit set had.

Deze router stuurt dit pakket vervolgens door naar de tunnelbestemming.



3. Opnieuw, veronderstel er een router tussen de tunnelbron en de bestemming met een verbinding MTU van 1400 is.

Deze router fragmenteert het tunnelpakket niet omdat het DF-bit is ingesteld (DF=1).

Deze router moet het pakket laten vallen en een ICMP-foutbericht naar de tunnelbronrouter sturen, omdat dat het IPv4-bronadres op het pakket is.

4. De verzendende router bij de tunnelbron ontvangt deze "ICMP"-foutmelding en verlaagt de GRE-tunnel IPv4 MTU tot 1376 (1400 - 24).

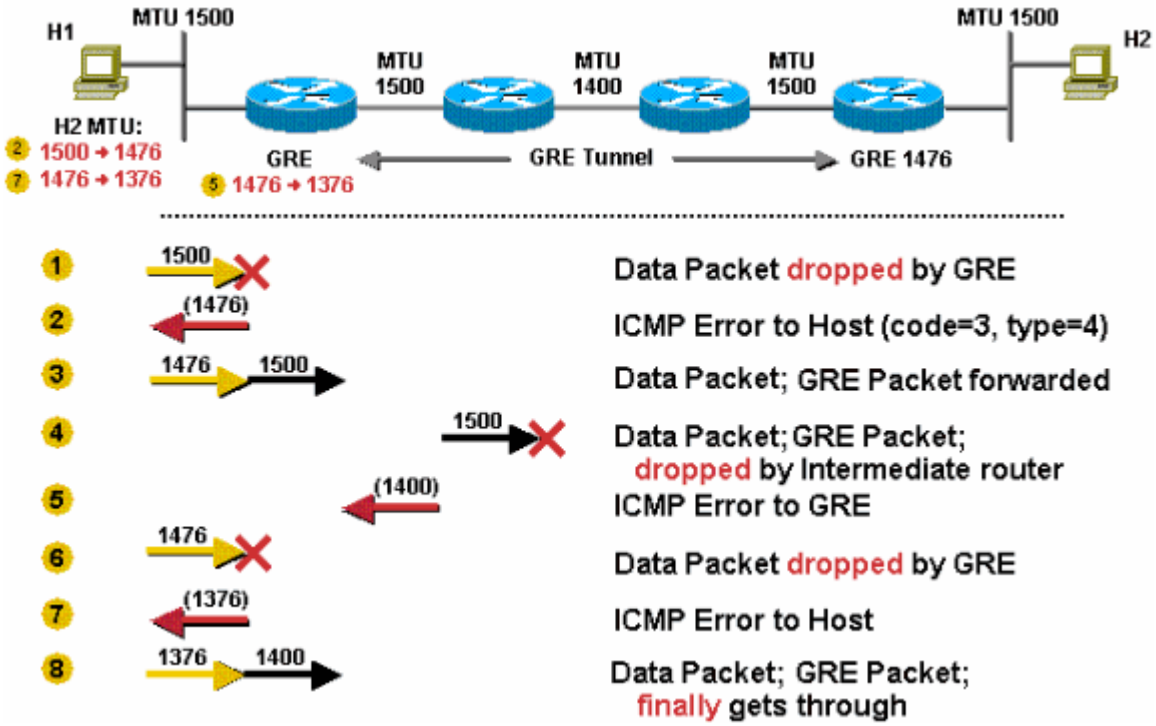
De volgende keer dat de verzendende host de gegevens opnieuw verzendt in een IPv4-pakket van 1476 bytes, kan dit pakket te groot zijn en deze router verstuurt vervolgens een "ICMP"-foutmelding naar de afzender met een MTU-waarde van 1376.

Wanneer de verzendende host de gegevens opnieuw verzendt, wordt deze verzonden in een IPv4-pakket van 1376 bytes. Dit pakket maakt de gegevens door de GRE-tunnel naar de ontvangende host.

Voorbeeld 5

Dit voorbeeld illustreert GRE-fragmentatie. Fragment voor insluiting voor GRE, dan doe PMTUD voor het gegevenspakket, en het DF-bit wordt niet gekopieerd wanneer het IPv4-pakket door GRE wordt ingekapseld.

Het DF-bit is niet ingesteld. De IPv4-MTU van de GRE-tunnelinterface is standaard 24 bytes minder dan de IPv4-MTU van de fysieke interface. De IPv4-MTU van de GRE-interface is dus 1476, zoals aangegeven in de afbeelding.



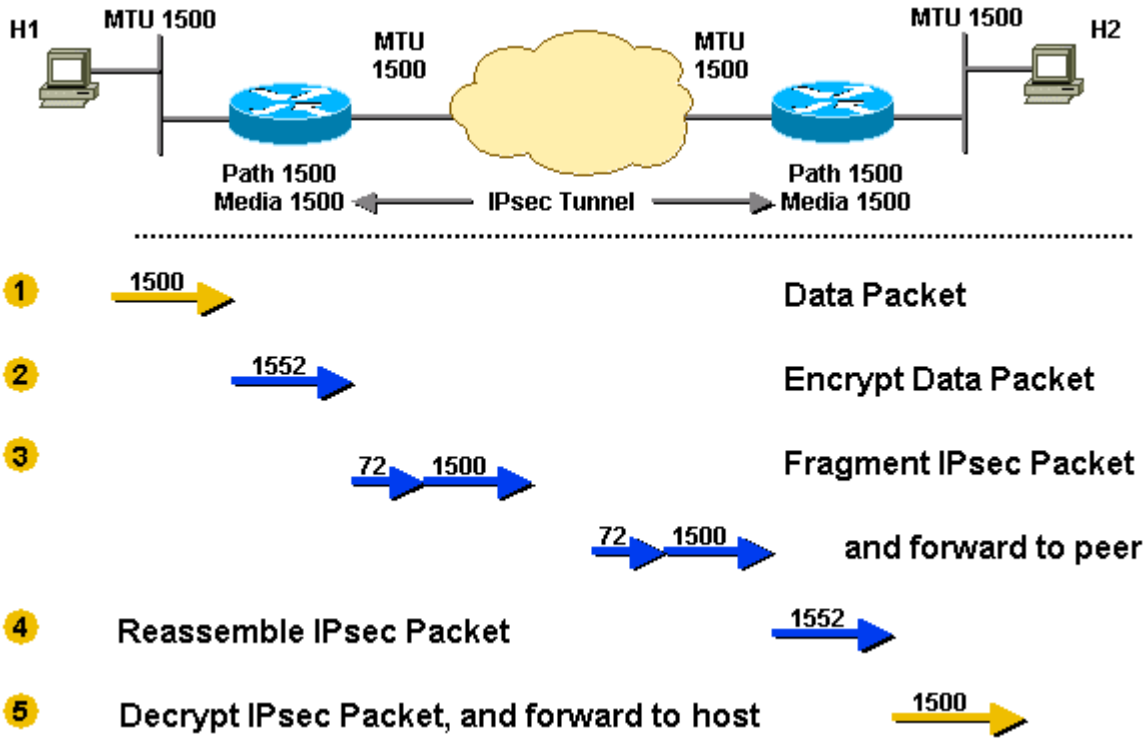
1. De verzender verzendt een pakket van 1500 bytes (IPv4-header van 20 bytes + TCP-payload van 1480 bytes).
2. Omdat de MTU van de GRE-tunnel 1476 is, is het 1500-byte pakket gebroken in twee IPv4 fragmenten van 1476 en 44 bytes, elk in afwachting van de extra 24 bytes van GRE-header.
3. De 24 bytes van de GRE-header worden toegevoegd aan elk IPv4-fragment. De fragmenten zijn nu 1500 (1476 + 24) en 68 (44 + 24) bytes.
4. De GRE + IPv4-pakketten met de twee IPv4-fragmenten worden doorgestuurd naar de router van de GRE-tunnelpeer.
5. De router van de GRE-tunnelpeer verwijdert de GRE-headers uit de twee pakketten.
6. Deze router stuurt de twee pakketten door naar de bestemmingshost.
7. De bestemmingshost stelt de IPv4-fragmenten opnieuw samen tot het oorspronkelijke IPv4-datagram.

Voorbeeld 6

Dit voorbeeld is vergelijkbaar met Voorbeeld 5, maar deze keer wordt het DF-bit ingesteld. De router is ingesteld om PMTUD te doen op GRE + IPv4 tunnelpakketten met de **tunnel path-mtu-discovery**opdracht en de DF-bit wordt gekopieerd van de oorspronkelijke IPv4-header naar de GRE IPv4-header.

Als de router een ICMP-foutbericht ontvangt voor het GRE + IPv4-pakket, verlaagt deze de IPv4-MTU voor de GRE-tunnelinterface.

De GRE Tunnel IPv4 MTU is standaard ingesteld op 24 bytes minder dan de fysieke interface MTU, dus de GRE IPv4 MTU hier is 1476. Er is een 1400 MTU-verbinding in het GRE-tunnelpad zoals in de afbeelding.



1. De router ontvangt een pakket van 1500 bytes (IPv4-header van 20 bytes + TCP-payload van 1480 bytes) en wijst het pakket af. Het pakket wordt door de router afgewezen omdat het groter is dan de IPv4-MTU (1476) van de GRE-tunnelinterface.
2. De router stuurt een ICMP-foutbericht naar de verzender met de melding dat de MTU van de volgende hop 1476 is. De host registreert deze informatie, gewoonlijk als een host route voor de bestemming in zijn routingtabel.
3. De verzendende host gebruikt een pakketgrootte van 1476 bytes bij het opnieuw verzenden van de data. De GRE-router voegt 24 bytes aan GRE-inkapseling toe en verzendt een pakket van 1500 bytes.
4. Het pakket van 1500 bytes kan niet worden verzonden via de link van 1400 bytes en wordt dus door de tussenliggende router afgewezen.
5. De tussenliggende router stuurt een ICMP-bericht (type 3, code 4) naar de GRE-router met een MTU van de volgende hop van 1400. De GRE-router verlaagt deze tot 1376 (1400 â€“ 24) en stelt een interne IPv4-MTU-waarde in voor de GRE-interface. Deze verandering kan alleen worden waargenomen als u de **debug tunnel** commando; het kan niet worden gezien in de uitvoer van de **show ip interface tunnel**<#> uit.
6. De volgende keer dat de host het 1476-byte pakket opnieuw verstuurt, laat de GRE-router het pakket vallen, omdat het groter is dan de huidige IPv4 MTU (1376) op de GRE-tunnelinterface.
7. De GRE router verstuurt een andere ICMP (type = 3, code = 4) naar de afzender met een volgende-hop MTU van 1376 en de host werkt zijn huidige informatie bij met nieuwe waarde.
8. De host verstuurt de gegevens opnieuw, maar nu in een kleiner 1376-byte pakket, GRE voegt 24 bytes van inkapseling toe en doorsturen het op. Dit keer maakt het pakket het tot de GRE tunnelpeer, waar het pakket wordt gedecapsuleerd en verzonden naar de bestemmingsgastheer.

Opmerking: Als de **tunnel path-mtu-discovery** Het bevel werd niet gevormd op de het door:sturen router in dit scenario, en het DF-beetje werd geplaatst in de pakketten die door de tunnel GRE door:sturen, slaagt Gastheer 1 nog in het verzenden van TCP/IPv4 pakketten naar Gastheer 2, maar zij worden gefragmenteerd in het midden bij de 1400 MTU verbinding. Ook de GRE tunnelpeer moet ze opnieuw monteren voordat ze kunnen decapsuleren en doorsturen.

Pure IPsec-tunnelmodus

Het IPv4sec-protocol (IPv4 Security) is een op standaarden gebaseerde methode die de privacy, integriteit en authenticiteit garandeert van informatie die via IPv4-netwerken wordt overgedragen.

IPv4sec biedt IPv4-encryptie op de netwerklaag. IPv4sec verlengt het IPv4-pakket door ten minste één IPv4-header toe te voegen (tunnelmodus).

De toegevoegde header(s) varieert in lengte afhankelijk van de IPv4sec-configuratiemodus, maar ze overschrijden niet ~58 bytes (Encapsulating Security payload (ESP) en ESP-verificatie (ESPauth) per pakket.

IPv4sec heeft twee modi: tunnelmodus en transportmodus.

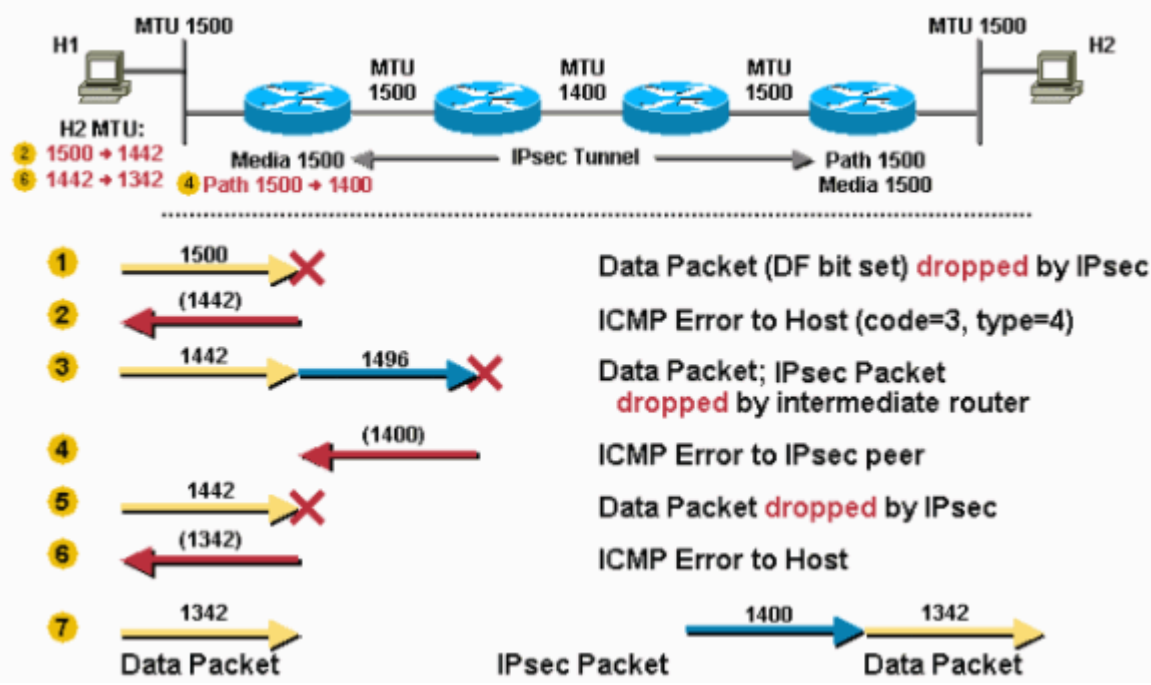
1. De tunnelmodus is de standaardmodus. In de tunnelmodus wordt het volledige oorspronkelijke IPv4-pakket beschermd (versleuteld, geverifieerd of beide) en ingekapseld door de IPv4sec-headers en â€˜trailers. Vervolgens wordt een nieuwe IPv4-header toegevoegd aan het pakket die de IPv4sec-endpoints (peers) aangeeft als bron en bestemming. De tunnelmodus kan worden gebruikt met al het unicast IPv4-verkeer en moet worden gebruikt als IPv4sec verkeer van hosts achter de IPv4sec-peers beschermt. De tunnelmodus wordt bijvoorbeeld gebruikt met Virtual Private Networks (VPNâ€™s) waar hosts op een beschermd netwerk pakketten verzenden naar hosts op een ander beschermd netwerk via enkele IPv4sec-peers. Bij VPNâ€™s beschermt de IPv4sec-â€˜tunnelâ€™ het IPv4-verkeer tussen hosts door het verkeer tussen de IPv4sec-peerrouters te versleutelen.
2. Met transportmodus (geconfigureerd met subcommando, **mode transport**, op de transformatiedefinitie) wordt alleen de payload van het oorspronkelijke IPv4-pakket beveiligd (versleuteld, geverifieerd of beide). De payload wordt ingekapseld door de IPv4sec-headers en â€˜trailers. De oorspronkelijke IPv4-headers blijven intact, maar het IPv4-protocolveld wordt gewijzigd naar ESP (50). De oorspronkelijke protocolwaarde wordt opgeslagen in de IPv4sec-trailer en wordt hersteld wanneer het pakket wordt ontsleuteld. De transportmodus wordt alleen gebruikt wanneer het te beschermen IPv4-verkeer tussen de IPv4sec-peers zelf plaatsvindt en de IPv4-adressen voor bron en bestemming in het pakket gelijk zijn aan de adressen van de IPv4sec-peers. Normaliter wordt de IPv4sec-transportmodus alleen gebruikt wanneer een ander tunnelingprotocol (zoals GRE) wordt gebruikt om eerst het IPv4-datapakket in te kapselen, waarna IPv4sec wordt gebruikt om de GRE-tunnelpakketten te beschermen.

IPv4sec voert altijd PMTUD uit voor datapakketten en voor de eigen pakketten. Er zijn IPv4sec-configuratieopdrachten om PMTUD-verwerking voor het IPv4sec IPv4-pakket aan te passen. IPv4 kan de DF-bit wissen, instellen of kopiëren van de IPv4-header in het datapakket naar de IPv4sec IPv4-header. Dit wordt de â€˜Functionaliteit voor negeren van DF-bitâ€™ genoemd.

> **Opmerking**: fragmentatie na inkapseling voorkomen wanneer hardware-encryptie met IPv4sec is uitgevoerd. Hardware encryptie geeft je doorvoersnelheid van ongeveer 50 Mbps die afhankelijk is van de hardware, maar als het IPv4sec pakket gefragmenteerd is verlies je 50 tot 90 procent van de doorvoersnelheid. Dit verlies wordt veroorzaakt doordat proces-switching wordt toegepast op de gefragmenteerde IPv4sec-pakketten voor het opnieuw samenstellen ervan, waarna ze voor decryptie worden overgedragen aan de engine voor hardware-encryptie. Dit verlies aan doorvoersnelheid kan ertoe leiden dat de snelheid van hardware-encryptie wordt verlaagd tot het prestatieniveau van software-encryptie (2â€“10 MB).

Voorbeeld 7

In dit scenario wordt IPv4sec-fragmentatie in actie getoond. De MTU langs het gehele pad is 1500. In dit scenario is de DF-bit niet ingesteld.



1. De router ontvangt een pakket van 1500 bytes (IPv4-header van 20 bytes + TCP-payload van 1480 bytes) met host 2 als bestemming.
2. Het pakket van 1500 bytes is versleuteld door IPv4sec en er zijn 52 bytes aan overhead toegevoegd (IPv4sec-header, â€'trailer en aanvullende IPv4-header). IPv4sec moet dus een pakket van 1552 bytes verzenden. Aangezien de uitgaande MTU 1500 is, moet dit pakket worden gefragmenteerd.
3. Het IPv4sec-pakket wordt opgedeeld in twee fragmenten. Tijdens fragmentatie wordt er een extra IPv4-header van 20 bytes toegevoegd voor het tweede fragment, wat resulteert in een fragment van 1500 bytes en een IPv4-fragment van 72 bytes.
4. De router van de IPv4sec-tunnelpeer ontvangt de fragmenten, verwijdert de aanvullende IPv4-header en stelt het oorspronkelijke IPv4sec-pakket weer samen uit de IPv4-fragmenten. Vervolgens ontsleutelt IPv4sec dit pakket.
5. De router stuurt het oorspronkelijke datapakket van 1500 bytes naar host 2.

Voorbeeld 8

Dit voorbeeld is vergelijkbaar met Voorbeeld 6 behalve dat in dit geval het DF-bit is ingesteld in het originele gegevenspakket en er een koppeling is in het pad tussen de IPv4sec-tunnelpeers die een lagere MTU heeft dan de andere koppelingen.

Dit voorbeeld laat zien hoe de peer router IPv4sec beide PMTUD-rollen uitvoert, zoals beschreven in The Router als een PMTUD-deelnemer aan het eindpunt van een tunnelsectie.

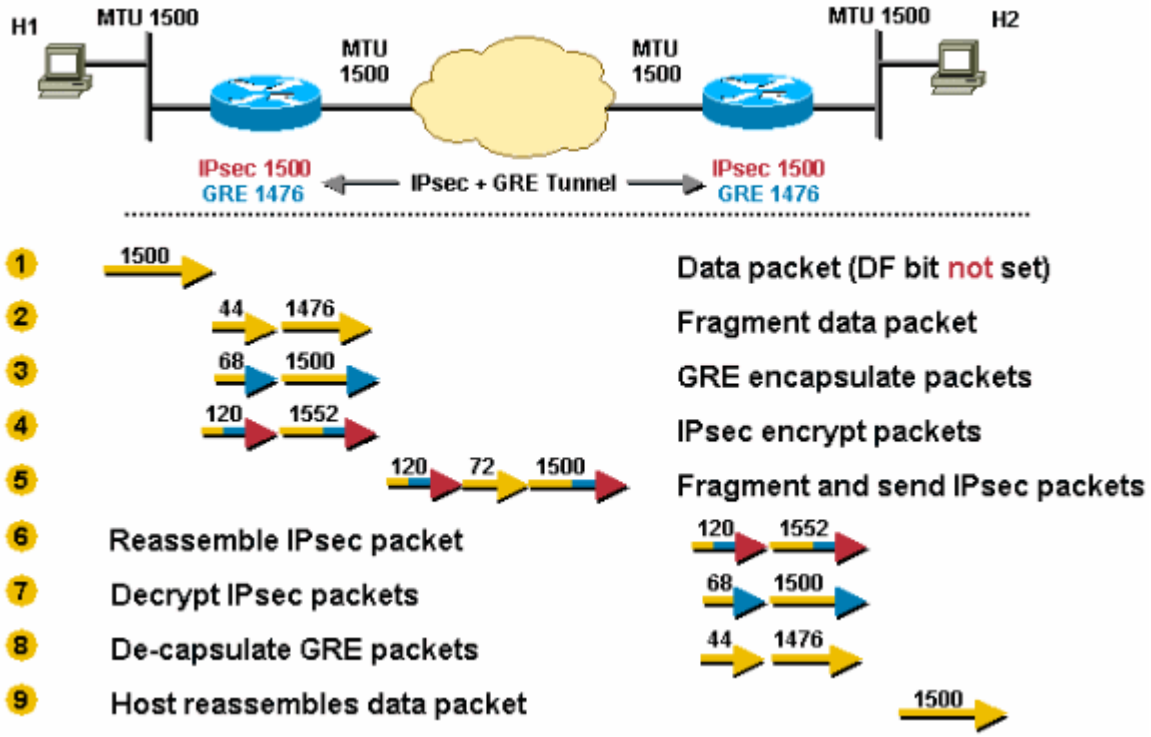
IPv4sec PMTU verandert in een lagere waarde als gevolg van de behoefte aan fragmentatie.

Het DF-bit wordt gekopieerd van de interne IPv4-header naar de externe IPv4-header wanneer IPv4sec een pakket versleutelt.

De MTU- en PMTU-waarde van de media worden opgeslagen in de IPv4sec Security Association (SA).

De MTU van de media is gebaseerd op de MTU van de uitgaande routerinterface; de PMTU is gebaseerd op de minimale MTU in het pad tussen de IPv4sec-peers.

IPv4sec kapselt/versleutelt het pakket voordat wordt geprobeerd het te fragmenteren zoals in de afbeelding.



1. De router ontvangt een 1500-byte pakket en laat het vallen omdat de IPv4sec overhead, wanneer toegevoegd, het pakket groter maakt dan de PMTU (1500).
2. De router stuurt een ICMP-bericht naar host 1 met de melding dat de MTU van de volgende hop 1442 is (1500 â€“ 58 = 1442). Deze 58 bytes zijn de maximale overhead van IPv4sec wanneer u IPv4sec ESP en ESPauth gebruikt. De echte IPv4sec-overhead is mogelijk 7 bytes minder dan deze waarde. Host 1 legt deze informatie vast in de routingtabel, doorgaans als een hostroute voor de bestemming (host 2).
3. Host 1 verlaagt zijn PMTU voor Host 2 tot 1442, zodat Host 1 kleinere (1442 bytes) pakketten verstuurt wanneer het de gegevens opnieuw doorgeeft aan Host 2. De router ontvangt het pakket van 1442 bytes en IPv4sec voegt 52 bytes aan encryptie-overhead toe, zodat het uiteindelijke IPv4sec-pakket 1496 bytes groot is. Aangezien de DF-bit is ingesteld in de header van dit pakket, wordt het afgewezen door de middelste router met de link met een MTU van 1400.
4. De middelste router die het pakket afwijst, stuurt een ICMP-bericht naar de verzender van het IPv4sec-pakket (de eerste router) met de melding dat de MTU van de volgende hop 1400 is. Deze waarde wordt vastgelegd in de IPv4sec SA-PMTU.
5. De volgende keer dat Host 1 het 1442-byte pakket opnieuw verzendt (er is geen bevestiging voor ontvangen), laat IPv4sec het pakket vallen. De router laat het pakket vallen omdat IPv4sec, wanneer toegevoegd aan het pakket, het groter maakt dan PMTU (1400).
6. De router stuurt een ICMP-bericht naar host 1 met de melding dat de MTU van de volgende hop nu 1342 is. (1400 â€“ 58 = 1342). Host 1 slaat deze informatie opnieuw op.
7. Wanneer host 1 de gegevens opnieuw doorgeeft, gebruikt het het kleinere omvangpakket (1342). Dit pakket vereist geen fragmentatie en maakt het door de IPv4sec-tunnel naar host 2.

Combinatie van GRE en IPv4sec

Complexere interacties voor fragmentatie en PMTUD treden op wanneer IPv4sec wordt gebruikt om GRE-tunnels te versleutelen.

IPv4sec en GRE worden op deze manier gecombineerd omdat IPv4sec geen ondersteuning biedt voor multicast IPv4-pakketten, waardoor geen protocol voor dynamische routing kan worden uitgevoerd via het IPv4sec-VPN.

GRE-tunnels ondersteunen multicast wel, zodat een GRE-tunnel kan worden gebruikt om het multicast pakket eerst via het protocol voor dynamische routing in te kapselen in een unicast GRE IPv4-pakket dat vervolgens door IPv4sec kan worden versleuteld.

Wanneer u dit doet, wordt IPv4sec vaak geïmplementeerd in transportmodus boven op GRE omdat de IPv4sec-peers en de GRE-tunnelendpoints (de routers) hetzelfde zijn en in transportmodus 20 bytes van IPv4sec-overhead worden opgeslagen.

Het wordt interessant wanneer een IPv4-pakket is opgedeeld in twee fragmenten en ingekapseld door GRE.

In dit geval ziet IPv4sec twee onafhankelijke GRE + IPv4-pakketten. Vaak in een standaardconfiguratie is een van deze pakketten groot genoeg dat het moet worden gefragmenteerd nadat het is versleuteld.

De IPv4sec peer moet dit pakket opnieuw assembleren voordat het kan worden gedecrypteerd. Deze â€˜dubbele fragmentatieâ€™ (een keer vóór GRE en opnieuw na IPv4sec) op de verzendende router vergroot de latentie en verlaagt de doorvoersnelheid.

De hermontage is proces-switched, zodat is er een CPU-hit op de ontvangende router wanneer dit gebeurt.

Deze situatie kan worden vermeden door de â€˜ip mtuâ€™ van de GRE-tunnelinterface laag genoeg in te stellen om rekening te houden met de overhead van zowel GRE als IPv4sec (standaard is de â€˜ip mtuâ€™ van de GRE-tunnelinterface ingesteld op de MTU van de werkelijke uitgaande interface â€“ GRE-overheadbytes).

Deze tabel geeft de voorgestelde MTU-waarden voor elke tunnel/moduscombinatie die ervan uitgaat dat de uitgaande fysieke interface een MTU van 1500 heeft.

Tunnelcombinatie	**Specifiek benodigde MTU**	**Aanbevolen MTU**
GRE + IPv4sec (transportmodus)	1440 bytes	1400 bytes
GRE + IPv4sec (tunnelmodus)	1420 bytes	1400 bytes

> **Opmerking**: de MTU-waarde van 1400 wordt aanbevolen omdat deze de meest gebruikelijke combinaties van GRE + IPv4sec-modus dekt. Bovendien is er geen duidelijk nadeel verbonden aan de extra overhead van 20 of 40 bytes. Het is eenvoudiger om slechts één waarde te onthouden en in te stellen die van toepassing is op bijna alle scenarioâ€™s.

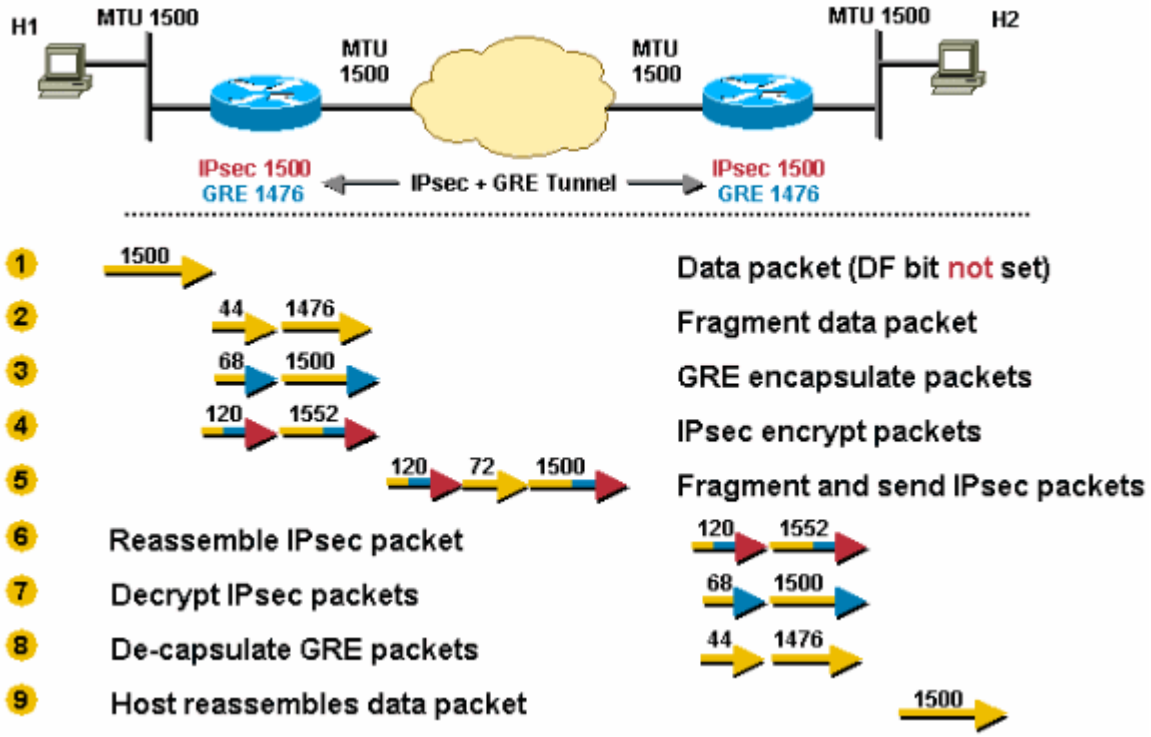
Voorbeeld 9

IPv4sec wordt geïmplementeerd bovenop GRE. De MTU van de uitgaande fysieke interface is 1500, de IPv4sec-PMTU is 1500 en de GRE IPv4-MTU is 1476 (1500 â€“ 24 = 1476).

TCP/IPv4-pakketten zijn daarom twee keer gefragmenteerd, één keer voor GRE en één keer na IPv4sec.

Het pakket is gefragmenteerd voor GRE-insluiting en een van deze GRE-pakketten wordt opnieuw gefragmenteerd na IPv4sec-encryptie.

Door â€˜ip mtu 1440â€™ (IPv4sec-transportmodus) of â€˜ip mtu 1420â€™ (IPv4sec-tunnelmodus) te configureren op de GRE-tunnel wordt in dit scenario de kans op dubbele fragmentatie weggenomen.

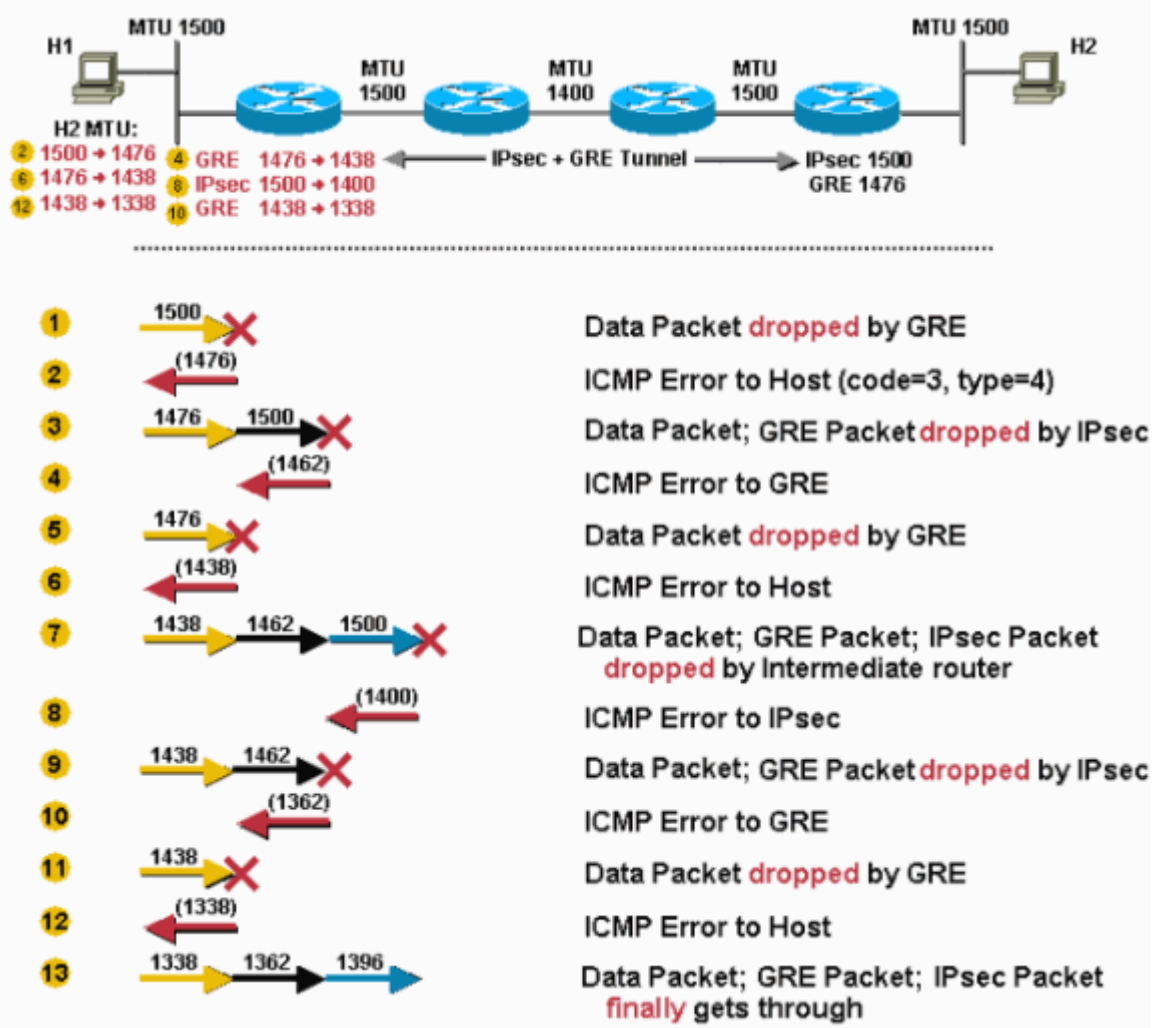


1. De router ontvangt een datagram van 1500 bytes.
2. Voorafgaand aan inkapseling fragmenteert GRE het pakket van 1500 bytes in twee delen: 1476 bytes (1500 â€“ 24 = 1476) en 44 bytes (24 voor data + 20 voor IPv4-header).
3. GRE kapselt de IPv4-fragmenten in, waardoor nog eens 24 bytes aan elk pakket worden toegevoegd. Dit levert twee GRE + IPv4sec-pakketten van respectievelijk 1500 bytes (1476 + 24) en 68 bytes (44 + 24) op.
4. IPv4sec versleutelt de twee pakketten, die 52 bytes (IPv4sec-tunnelmodus) aan elke inkapselingsmodus toevoegen, om een 152-byte en een 120-byte pakket te geven.
5. Het IPv4sec-pakket van 1552 bytes wordt door de router gefragmenteerd omdat het groter is dan de MTU van de uitgaande interface (1500). Het pakket van 1552 bytes wordt opgedeeld in een pakket van 1500 bytes en één van 72 bytes (payload van 52 bytes + 20 bytes extra van IPv4-header voor het tweede fragment). De drie pakketten van 1500, 72 en 120 bytes worden doorgestuurd naar de IPv4sec + GRE-peer.
6. De ontvangende router stelt de twee IPv4sec-fragmenten (1500 bytes en 72 bytes) opnieuw samen tot het oorspronkelijke IPv4sec + GRE-pakket van 1552 bytes. Er hoeft niets te worden gedaan met het IPv4sec + GRE-pakket van 120 bytes.
7. IPv4sec ontsleutelt beide IPv4sec + GRE-pakketten van 1552 en 120 bytes tot GRE-pakketten van 1500 en 68 bytes.
8. GRE ontkapselt de GRE-pakketten van 1500 en 68 bytes tot IPv4-pakketfragmenten van 1476 en 44 bytes. Deze IPv4-pakketfragmenten worden doorgestuurd naar de bestemmingshost.
9. Host 2 stelt deze IPv4-fragmenten opnieuw samen tot het oorspronkelijke IPv4-datagram van 1500 bytes.

Scenario 10 is vergelijkbaar met scenario 8 met dit verschil dat zich nu een link met een lagere MTU in het tunnelpad bevindt. In dit scenario is sprake van het slechtste geval voor het eerste pakket dat van host 1 naar host 2 wordt verzonden. Na de laatste stap in dit scenario stelt Host 1 de juiste PMTU voor Host 2 in en alles is goed voor de TCP-verbindingen tussen Host 1 en Host 2. TCP-stromen tussen host 1 en andere hosts (bereikbaar via de IPv4sec + GRE-tunnel) hoeven alleen door de laatste drie stappen van Scenario 10 te gaan.

In dit scenario **tunnel path-mtu-discovery** Het bevel wordt gevormd op de tunnel GRE en het DF beetje wordt geplaatst op TCP/IPv4 pakketten die uit Gastheer 1 voortkomen.

Voorbeeld 10



- De router ontvangt een pakket van 1500 bytes. Dit pakket wordt door GRE afgewezen omdat het pakket niet kan worden gefragmenteerd of doorgestuurd aangezien de DF-bit is ingesteld en het pakket groter is dan de â€˜ip mtuâ€™ van de uitgaande interface na toevoeging van de GRE-overhead (24 bytes).
- De router stuurt een ICMP-bericht naar host 1 met de melding dat de MTU van de volgende hop 1476 is (1500 â€“ 24 = 1476).
- Host 1 wijzigt zijn PMTU voor host 2 naar 1476 en verzendt vervolgens een kleiner pakket. GRE kapselt het in en draagt het pakket van 1500 bytes over aan IPv4sec. IPv4sec wijst het pakket af omdat GRE de DF-bit (ingesteld) heeft gekopieerd van de binnenste IPv4-header. Met de IPv4sec-overhead (maximaal 38 bytes) is het pakket te groot om te worden doorgestuurd naar de fysieke interface.
- IPv4sec verzendt een ICMP-bericht naar GRE dat aangeeft dat de volgende hop-MTU 1462 bytes is (aangezien er maximaal 38 bytes zijn toegevoegd voor codering en IPv4-overhead). GRE legt de waarde 1438 (1462 â€“ 24) vast als de â€˜ip mtuâ€™ voor de tunnelinterface.

- **Opmerking**: deze waardeverandering wordt intern opgeslagen en kan niet worden gezien in de uitvoer van de **show ip interface tunnel**<#>uit. U ziet deze verandering alleen als u de **debug tunnel** uit.

- De volgende keer dat host 1 het pakket van 1476 bytes opnieuw verzendt, zal GRE het pakket afwijzen.
- De router stuurt een ICMP-bericht naar host 1 met de melding dat de MTU van de volgende hop 1438 is.
- Host 1 verlaagt de PMTU voor host 2 en verzendt opnieuw een pakket van 1438 bytes. Deze keer

accepteert GRE het pakket, kapselt het in en draagt het over aan IPv4sec voor encryptie.
- Het IPv4sec-pakket wordt doorgestuurd naar de tussenliggende router en afgewezen omdat die een MTU van de uitgaande interface van 1400 heeft.
- De tussenliggende router stuurt een ICMP-bericht naar IPv4sec met de melding dat de MTU van de volgende hop 1400 is. Deze waarde wordt door IPv4sec vastgelegd in de PMTU-waarde van de bijbehorende IPv4sec SA.
- Wanneer host 1 het pakket van 1438 bytes opnieuw verzendt, kapselt GRE het pakket in. Vervolgens draagt GRE het pakket over aan IPv4sec. IPv4sec wijst het pakket af omdat het de eigen PMTU heeft gewijzigd in 1400.
- IPv4sec stuurt een ICMP-foutbericht naar GRE met de melding dat de MTU van de volgende hop 1362 is en GRE legt de waarde 1338 intern vast.
- Wanneer host 1 het oorspronkelijke pakket opnieuw verzendt (omdat geen bevestiging is ontvangen), wijst GRE het pakket af.
- De router stuurt een ICMP-bericht naar host 1 met de melding dat de MTU van de volgende hop 1338 is (1362 â€“ 24). Host 1 verlaagt zijn PMTU voor host 2 tot 1338.
- Host 1 verzendt opnieuw een pakket van 1338 bytes en deze keer bereikt het host 2 wel.

Meer aanbevelingen

De **tunnel path-mtu-discovery** het bevel op een tunnelinterface kan GRE en IPv4sec interactie helpen wanneer zij op de zelfde router worden gevormd.

Zonder de **tunnel path-mtu-discovery** Als de opdracht geconfigureerd is, wordt de DF-bit altijd gewist in de GRE IPv4-header.

Hierdoor kan het GRE IPv4-pakket worden gefragmenteerd ook al is de DF-bit ingesteld in de ingekapselde data-IPv4-header (waardoor normaliter het pakket niet kan worden gefragmenteerd).

Indien de **tunnel path-mtu-discovery** Het bevel wordt gevormd op de GRE tunnelinterface:

1. GRE kopieert de DF-bit van de IPv4-header van de gegevens naar de GRE IPv4-header.
2. Als het DF-bit in de GRE IPv4-header is ingesteld en het pakket na de IPv4sec-codering "te groot" is voor de IPv4 MTU op de fysieke uitgaande interface, dan laat IPv4sec het pakket vallen en waarschuwt de GRE-tunnel om de IPv4 MTU-grootte te beperken.
3. IPv4sec doet PMTUD voor zijn eigen pakketten en als de IPv4sec PMTU verandert (als het wordt verminderd), dan IPv4sec niet onmiddellijk op de hoogte GRE, maar wanneer een ander groter pakket komt door, dan het proces in stap 2 voorkomt.
4. De GRE IPv4 MTU is nu kleiner, dus het laat alle IPv4-pakketten vallen met de DF-bitset die nu te groot zijn en verstuur een ICMP-bericht naar de verzendende host.

Het **tunnel path-mtu-discovery** de opdracht helpt de GRE-interface om zijn IPv4 MTU dynamisch in te stellen, in plaats van statisch met de **ip mtu** uit. Het wordt aanbevolen beide opdrachten te gebruiken.

Het **ip mtu** De opdracht wordt gebruikt om ruimte te bieden voor de GRE- en IPv4sec-overhead ten opzichte van de lokale fysieke uitgaande interface IPv4 MTU.

Het **tunnel path-mtu-discovery** Met deze opdracht kan de GRE-tunnel IPv4 MTU verder worden gereduceerd als er een lagere IPv4 MTU-link is in het pad tussen de IPv4sec-peers.

Hierna volgen enkele oplossingen voor als u problemen ondervindt met PMTUD in een netwerk met geconfigureerde GRE + IPv4sec-tunnels.

De lijst begint met de oplossing die de voorkeur heeft.

1. Los het probleem van niet-werkende PMTUD op. Dit wordt meestal veroorzaakt door een router of firewall die ICMP blokkeert.
2. Gebruik de **ip tcp adjust-mss** opdracht op de tunnelinterfaces, zodat de router de TCP MSS-waarde in het TCP/SYN-pakket vermindert. Dit helpt de twee eindgastheren (de afzender en de ontvanger van TCP) om pakketten te gebruiken klein genoeg zodat PMTUD niet nodig is.
3. Gebruik beleidsroutering op de inkomende interface van de router en configureer een routekaart om de DF-bit in de data-IPv4-header van de data te wissen voordat deze de GRE-tunnelinterface bereiken. Hierdoor kan het IPv4-pakket met gegevens worden gefragmenteerd voor GRE-insluiting.
4. Verhoog de â€˜ip mtuâ€™ voor de GRE-tunnelinterface zodat deze overeenkomt met de MTU van de uitgaande interface. Hierdoor kan het IPv4-pakket met gegevens worden ingekapseld zonder dat het eerst wordt gefragmenteerd. Het GRE-pakket wordt vervolgens versleuteld met IPv4sec en vervolgens gefragmenteerd om de fysieke uitgaande interface uit te gaan. In dit geval zou u niet configureren **tunnel path-mtu-discovery** opdracht op de GRE-tunnelinterface. Hierdoor kan de doorvoersnelheid aanzienlijk afnemen omdat opnieuw samenstellen van het IPv4-pakket op de IPv4sec-peer wordt uitgevoerd in de modus proces-switching.

Gerelateerde informatie

- **Ondersteuningspagina voor IP-routing**
- **Ondersteuningspagina voor IPsec (IP security protocol)**
- **RFC 1191 PMTUD â€“ Path MTU Discovery**
- **RFC 1063 IP MTU Discovery Options**
- **RFC 791 Internet Protocol**
- **RFC 793 Transmission Control Protocol**
- **RFC 879 The TCP Maximum Segment Size and Related Topics**
- **RFC 1701 Generic Routing Encapsulation (GRE)**
- **RFC 1241 A Scheme for an Internet Encapsulation Protocol**
- **RFC 2003 IP Encapsulation within IP**
- **Technische ondersteuning en documentatie â€“ Cisco Systems**

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.