# Behebung von IPv4-Fragmentierung, MTU-, MSS- und PMTUD-Problemen mit GRE und IPSec

## Inhalt



## Einleitung

In diesem Dokument wird die Funktionsweise von IPv4-Fragmentierung und Path Maximum Transmission Unit Discovery (PMTUD) beschrieben.

## Hintergrundinformationen

Ebenfalls diskutiert werden Szenarien, die das Verhalten der PMTUD in Kombination mit verschiedenen Kombinationen von IPv4-Tunneln betreffen.

## IPv4-Fragmentierung und -Reassemblierung

Obwohl die maximale Länge eines IPv4-Datagramms 65535 beträgt, erzwingen die meisten Übertragungswege eine kleinere maximale Paketlänge, die als MTU bezeichnet wird. Der MTU-Wert hängt von der Übertragungsstrecke ab.

Das Design von IPv4 berücksichtigt MTU-Unterschiede, da es Routern ermöglicht, IPv4-Datagramme nach Bedarf zu fragmentieren.

Die empfangende Station ist für die Reassemblierung der Fragmente in das ursprüngliche IPv4-Datagramm in voller Größe verantwortlich.

Die IPv4-Fragmentierung zerlegt ein Datagramm in Teile, die später wieder zusammengesetzt werden.

Die IPv4-Quell-, Ziel-, Identifizierungs-, Gesamtlänge- und Fragment-Offset-Felder sowie die Markierungen "Weitere Fragmente" (MF) und "Nicht fragmentieren" (DF) im IPv4-Header werden für die IPv4-Fragmentierung und -Reassemblierung verwendet.

Weitere Informationen über die Mechanik der IPv4-Fragmentierung und -Reassemblierung finden Sie unter RFC 791.

Dieses Bild zeigt das Layout eines IPv4-Headers.

**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 |

Die Kennung beträgt 16 Bit und ist ein Wert, der vom Sender eines IPv4-Datagramms zugewiesen wird. Dies unterstützt die Reassemblierung der Fragmente eines Datagramms.

Der Fragment-Offset beträgt 13 Bit und gibt an, wo ein Fragment im ursprünglichen IPv4-Datagramm hingehört. Dieser Wert ist ein Vielfaches von 8 Bytes.

Im Flags-Feld des IPv4-Headers befinden sich 3 Bit für Steuerungs-Flags. Das "Nicht fragmentieren"-Bit (DF-Bit) bestimmt, ob ein Paket fragmentiert werden darf.

Bit 0 ist reserviert und immer auf 0 gesetzt.

Bit 1 ist das DF-Bit (0 = "kann fragmentieren", 1 = "nicht fragmentieren").

Bit 2 ist das MF-Bit (0 = â€žletztes Fragmentâ€œ, 1 = â€žweitere Fragmenteâ€œ).

| **Wert** | **Bit 0 Reserviert** | **Bit 1 DF** | **Bit 2 MF** |
|---|---|---|---|
| 0 | 0 | Mai | Letzte |
| 1 | 0 | Das sollten Sie unterlassen: | Mehr |

Wenn die Längen der IPv4-Fragmente hinzugefügt werden, überschreitet der Wert die ursprüngliche IPv4-Datagrammlänge um 60.

Der Grund dafür, dass die Gesamtlänge um 60 erhöht wird, ist, dass drei zusätzliche IPv4-Header erstellt wurden, einer für jedes Fragment nach dem ersten.

Das erste Fragment hat einen Offset von 0, die Länge dieses Fragments beträgt 1500; dies beinhaltet 20 Byte für den leicht modifizierten ursprünglichen IPv4-Header.

Das zweite Fragment hat einen Offset von 185 (185 x 8 = 1480); der Datenteil dieses Fragments beginnt 1480 Byte nach dem ursprünglichen IPv4-Datagramm.

Die Länge dieses Fragments beträgt 1500; dies schließt den zusätzlichen IPv4-Header ein, der für dieses Fragment erstellt wurde.

Das dritte Fragment hat einen Offset von 370 (370 x 8 = 2960); der Datenteil dieses Fragments beginnt 2960 Byte nach dem ursprünglichen IPv4-Datagramm.

Die Länge dieses Fragments beträgt 1500; dies schließt den zusätzlichen IPv4-Header ein, der für dieses Fragment erstellt wurde.

Das vierte Fragment hat einen Offset von 555 (555 x 8 = 4440), was bedeutet, dass der Datenteil dieses Fragments nach 4440 Byte des ursprünglichen IPv4-Datagramms beginnt.

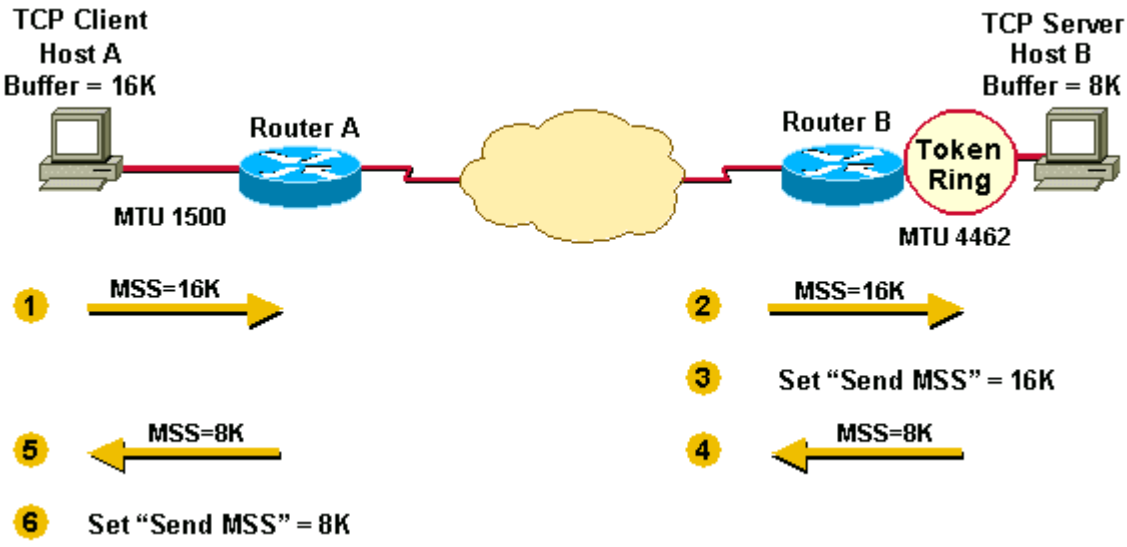
Die Länge dieses Fragments beträgt 700 Byte; dies schließt den zusätzlichen IPv4-Header ein, der für dieses Fragment erstellt wurde.

Erst wenn das letzte Fragment empfangen wird, kann die Größe des ursprünglichen IPv4-Datagramms bestimmt werden.

Der Fragment-Offset im letzten Fragment (555) ergibt einen Daten-Offset von 4440 Byte nach Beginn des ursprünglichen IPv4-Datagramms.

Die Summe der Datenbytes des letzten Fragments (680 = 700 - 20) ergibt 5120 Byte, das ist der Datenanteil des ursprünglichen IPv4-Datagramms.

Die Hinzufügung von 20 Byte für einen IPv4-Header entspricht der Größe des ursprünglichen IPv4-Datagramms (4440 + 680 + 20 = 5140), wie in den Bildern gezeigt.




## Probleme mit der IPv4-Fragmentierung

Die IPv4-Fragmentierung führt zu einem geringen Anstieg des CPU- und Arbeitsspeicher-Overheads zur

Fragmentierung eines IPv4-Datagramms. Dies gilt für den Sender und für einen Router im Pfad zwischen einem Sender und einem Empfänger.

Bei der Erstellung von Fragmenten werden Fragment-Header erstellt und das ursprüngliche Datagramm in die Fragmente kopiert.

Dies wird effizient durchgeführt, da die zur Erstellung der Fragmente benötigten Informationen sofort verfügbar sind.

Auf Empfängerseite verursacht die Reassemblierung der Fragmente eine höhere Auslastung, da der Empfänger Speicher für die ankommenden Fragmente zuweisen und diese nach dem Empfang aller Fragmente wieder zu einem Datagramm zusammenfügen muss.

Die Reassemblierung auf einem Host wird nicht als Problem angesehen, da der Host über die Zeit- und Speicherressourcen verfügt, die für diese Aufgabe erforderlich sind.

Die Reassemblierung ist jedoch auf einem Router, dessen Hauptaufgabe darin besteht, Pakete so schnell wie möglich weiterzuleiten, ineffizient.

Ein Router ist nicht dafür ausgelegt, Pakete für längere Zeit zu speichern.

Ein Router, der die Reassemblierung durchführt, wählt den größten verfügbaren Puffer (18 KB), da er die Größe des ursprünglichen IPv4-Pakets erst bestimmen kann, wenn das letzte Fragment empfangen wird.

Ein weiteres Fragmentierungsproblem betrifft den Umgang mit verloren gegangenen Fragmenten.

Wenn ein Fragment eines IPv4-Datagramms verworfen wird, muss das gesamte ursprüngliche IPv4-Datagramm vorhanden sein, und es ist ebenfalls fragmentiert.

Dies zeigt sich beim Network File System (NFS). NFS hat eine Lese- und Schreibblockgröße von 8192.

Daher beträgt ein NFS-IPv4/UDP-Datagramm ca. 8500 Byte (einschließlich NFS-, UDP- und IPv4-Headern).

Eine mit einem Ethernet (MTU 1500) verbundene Sendestation muss das 8500-Byte-Datagramm in sechs (6) Teile zerlegen: fünf (5) 1500-Byte-Fragmente und ein (1) 1100-Byte-Fragment.

Wenn eines der sechs Fragmente aufgrund einer überlasteten Verbindung verworfen wird, muss das gesamte ursprüngliche Datagramm erneut übertragen werden. Das Ergebnis sind sechs weitere Fragmente, die erzeugt werden müssen.

Wenn diese Verbindung eines von sechs Paketen verwirft, ist die Wahrscheinlichkeit gering, dass NFS-Daten über diese Verbindung übertragen werden, da mindestens ein IPv4-Fragment von jedem ursprünglichen NFS-IPv4-Datagramm mit 8500 Byte verworfen würde.

Firewalls, die Pakete basierend auf Layer-4- (L4) bis Layer-7- (L7)-Informationen filtern oder bearbeiten, haben Probleme bei der korrekten Verarbeitung von IPv4-Fragmenten.

Wenn die IPv4-Fragmente nicht in der richtigen Reihenfolge angeordnet sind, blockiert eine Firewall die nicht anfänglichen Fragmente, da sie nicht die Informationen enthalten, die mit dem Paketfilter übereinstimmen.

Das bedeutet, dass das ursprüngliche IPv4-Datagramm vom empfangenden Host nicht wieder zusammengesetzt werden konnte.

Wenn die Firewall so konfiguriert ist, dass Fragmente, die keine ersten Fragmente enthalten und nicht über

ausreichende Informationen verfügen, den richtigen Filter verwenden können, ist ein Angriff durch die Firewall möglich, der keine ersten Fragmente enthält.

Netzwerkgeräte wie die Content Switch Engine leiten Pakete basierend auf L4- bis L7-Informationen weiter. Umfasst ein Paket mehrere Fragmente, kann das Gerät seine Richtlinien nur schwer durchsetzen.

## IPv4-Fragmentierung vermeiden: So funktioniert TCP MSS

Das Transmission Control Protocol (TCP) Maximum Segment Size (MSS) definiert die maximale Datenmenge, die ein Host in einem einzelnen TCP/IPv4-Datagramm akzeptiert.

Dieses TCP/IPv4-Datagramm ist möglicherweise auf der IPv4-Schicht fragmentiert. Der MSS-Wert wird als TCP-Kopfzeilenoption übertragen, allerdings nur in TCP SYN-Segmenten.

Die beiden Seiten der TCP-Verbindung geben sich gegenseitig ihren MSS-Wert bekannt. Der MSS-Wert wird *nicht* zwischen den Hosts ausgehandelt.

Der Ausgangshost muss die Größe der Daten pro TCP-Segment auf einen Wert begrenzen, der dem vom Eingangshost gemeldeten MSS-Wert entspricht oder diesen unterschreitet.

Ursprünglich bezeichnete MSS die Größe des zugewiesenen Puffers (größer oder gleich 65496 Byte) auf einer Empfangsstation, damit die in einem einzigen IPv4-Datagramm enthaltenen TCP-Daten gespeichert werden können.

MSS war das maximale Datensegment, das der TCP-Empfänger akzeptieren konnte. Dieses TCP-Segment kann bis zu 64 KB groß sein und auf der IPv4-Schicht fragmentiert werden, um an den empfangenden Host übertragen zu werden.

Der empfangende Host setzte dann das IPv4-Datagramm neu zusammen, bevor er das komplette TCP-Segment an den TCP-Layer weiterleitete.

**Festlegen und Verwenden von MSS-Werten zum Einschränken der Größe von TCP-Segmenten und IPv4-Datagrammen**

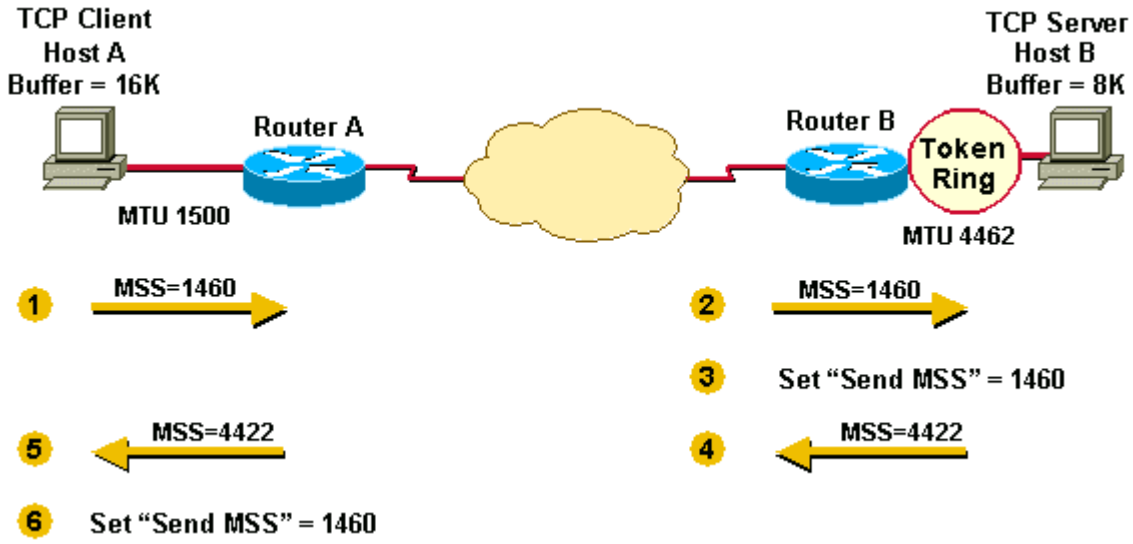
Beispiel 1 veranschaulicht, wie MSS zuerst implementiert wurde.

Host A hat einen Puffer von 16 kB und Host B einen Puffer von 8 kB. Die Hosts senden und empfangen ihre MSS-Werte und passen ihre Sende-MSS an, um Daten untereinander zu senden.

Host A und Host B müssen die IPv4-Datagramme fragmentieren, die größer als die MTU der Schnittstelle sind, jedoch kleiner als die Sende-MSS, da der TCP-Stack 16 KB oder 8 KB Daten den Stack entlang an IPv4 weiterleitet.

Im Fall von Host B werden Pakete fragmentiert, um in das Token Ring-LAN und wieder in das Ethernet-LAN zu gelangen.

**Beispiel 1**




1. Host A sendet seinen MSS-Wert von 16 kB an Host B.
2. Host B empfängt den 16 kB umfassenden MSS-Wert von Host A.
3. Host B setzt seinen Sende-MSS-Wert auf 16 kB.
4. Host B sendet seinen MSS-Wert von 8 kB an Host A.
5. Host A empfängt den 8 kB umfassenden MSS-Wert von Host B.
6. Host A setzt seinen Sende-MSS-Wert auf 8 kB.

Um eine IPv4-Fragmentierung an den Endpunkten der TCP-Verbindung zu vermeiden, wurde der MSS-Wert auf die minimale Puffergröße und die MTU der Ausgangsschnittstelle (- 40) geändert.

MSS-Nummern sind 40 Byte kleiner als MTU-Nummern, da MSS (die TCP-Datengröße) den 20-Byte-IPv4-Header und den 20-Byte-TCP-Header nicht enthält.

MSS basiert auf standardmäßigen Headergrößen. Der Sender-Stack muss die entsprechenden Werte für den IPv4-Header subtrahieren, und der TCP-Header hängt davon ab, welche TCP- oder IPv4-Optionen verwendet werden.

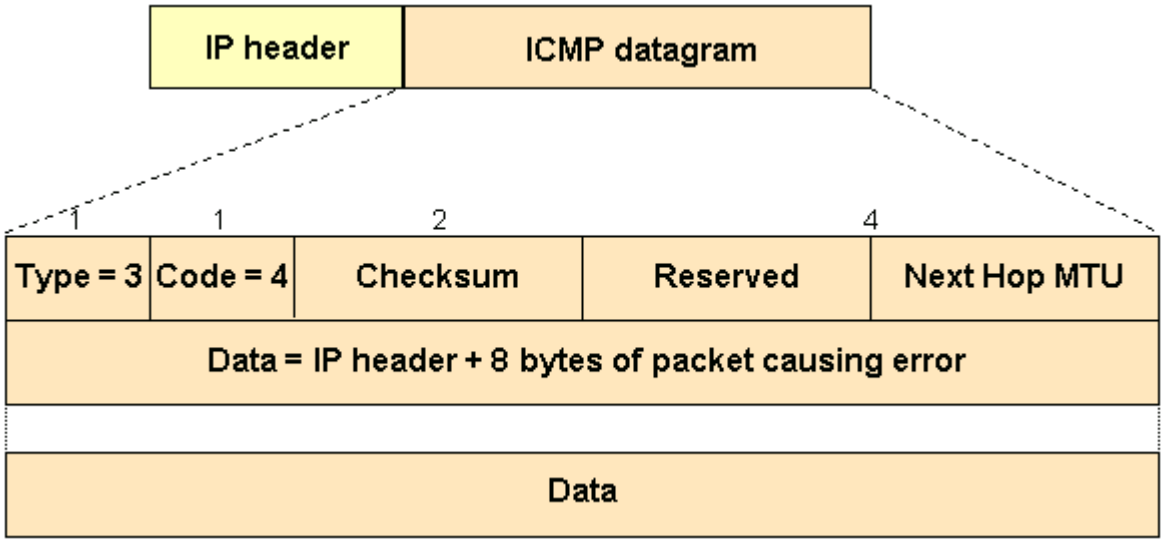
MSS funktioniert derzeit so, dass jeder Host zunächst seine MTU der Ausgangsschnittstelle mit seinem eigenen Puffer vergleicht und den niedrigsten Wert als zu sendende MSS auswählt.

Die Hosts vergleichen dann die empfangene MSS-Größe mit ihrer eigenen Schnittstellen-MTU und wählen wiederum den niedrigeren der beiden Werte aus.

**Beispiel 2** veranschaulicht diesen zusätzlichen Schritt des Absenders, um eine Fragmentierung der lokalen und Remote-Kabel zu vermeiden.

Die MTU der Ausgangsschnittstelle wird von jedem Host berücksichtigt, bevor die Hosts einander ihre MSS-Werte senden. Dies hilft, Fragmentierung zu vermeiden.

**Beispiel 2**




1. Host A vergleicht seinen MSS-Puffer (16 kB) und seine MTU (1500 - 40 = 1460) und verwendet den niedrigeren Wert (1460) als MSS zum Senden an Host B.
2. Host B empfängt die Sende-MSS (1460) von Host A und vergleicht sie mit dem Wert seiner Ausgangsschnittstelle MTU - 40 (4422).
3. Host B legt den kleineren Wert (1460) als MSS fest, um IPv4-Datagramme an Host A zu senden.
4. Host B vergleicht seinen MSS-Puffer (8 kB) und seine MTU (4462 - 40 = 4422) und verwendet 4422 als MSS, um an Host A zu senden.
5. Host A empfängt die Sende-MSS (4422) von Host B und vergleicht sie mit dem Wert seiner Ausgangsschnittstelle MTU -40 (1460).
6. Host A legt den unteren Wert (1460) als MSS für das Senden von IPv4-Datagrammen an Host B fest.

1460 ist der von beiden Hosts gewählte Wert als Sende-MSS füreinander. Häufig sind die Sende-MSS-Werte an jedem Ende einer TCP-Verbindung gleich.

In Beispiel 2 tritt keine Fragmentierung an den Endpunkten einer TCP-Verbindung auf, da beide MTUs der ausgehenden Schnittstellen von den Hosts berücksichtigt werden.

Pakete werden im Netzwerk zwischen Router A und Router B immer noch fragmentiert, wenn sie auf eine Verbindung mit einer niedrigeren MTU stoßen als die Ausgangsschnittstelle eines Hosts.

# Was ist die PMTUD

TCP-MSS adressiert die Fragmentierung an den beiden Endpunkten einer TCP-Verbindung, behandelt jedoch keine Fälle, in denen in der Mitte zwischen diesen beiden Endpunkten eine Verbindung mit kleinerer MTU vorhanden ist.

Die PMTUD wurde entwickelt, um eine Fragmentierung im Pfad zwischen den Endpunkten zu vermeiden. Er wird verwendet, um die niedrigste MTU auf dem Pfad von einer Paketquelle zu ihrem Ziel dynamisch zu bestimmen.

---

> **Hinweis**: Die PMTUD wird nur von TCP und UDP unterstützt. aber nicht von anderen Protokollen unterstützt. Wenn die PMTUD auf einem Host aktiviert ist, ist für alle TCP- und UDP-Pakete vom Host das DF-Bit festgelegt.

---

Wenn ein Host ein vollständiges MSS-Datenpaket mit festgelegtem DF-Bit sendet, reduziert die PMTUD den Sende-MSS-Wert für die Verbindung, wenn sie die Information erhält, dass das Paket fragmentiert

werden muss.

Ein Host zeichnet den MTU-Wert für ein Ziel auf, da er einen Host-Eintrag (/32) in seiner Routing-Tabelle mit diesem MTU-Wert erstellt.

Wenn ein Router versucht, ein IPv4-Datagramm (mit festgelegtem DF-Bit) an eine Verbindung weiterzuleiten, deren MTU kleiner ist als die Größe des Pakets, verwirft der Router das Paket und gibt eine ICMP-Meldung (Internet Control Message Protocol) "Destination Unreachable" mit dem Code "fragmentation needed and DF set" an die IPv4-Datagrammquelle zurück (Typ 3, Code 4).

Wenn die Quellstation die ICMP-Nachricht empfängt, senkt sie die Sende-MSS, und wenn TCP das Segment erneut sendet, verwendet sie die kleinere Segmentgröße.

Hier ist ein Beispiel für eine ICMP-Meldung "fragmentation needed and DF set", die auf einem Router nach dem **debug ip icmp** -Befehl ist aktiviert:

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

Dieses Diagramm zeigt das Format des ICMP-Headers einer Meldung â€žfragmentation needed and DF setâ€œ â€žDestination Unreachable".

| 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 |

Gemäß RFC 1191 muss ein Router, der eine ICMP-Meldung mit dem Hinweis â€žfragmentation needed and DF setâ€œ angibt, die MTU dieses Next-Hop-Netzwerks in die unteren 16 Bit des ICMP-Zusatz-Headerfeldes aufnehmen, das in der ICMP-Spezifikation RFC 792 als â€žunusedâ€œ gekennzeichnet ist.

Frühe Implementierungen von RFC 1191 haben die Informationen zur MTU am nächsten Hop nicht geliefert. Selbst wenn diese Informationen geliefert wurden, ignorieren einige Hosts sie.

Für diesen Fall enthält RFC 1191 auch eine Tabelle, in der die vorgeschlagenen Werte aufgeführt sind, um die die MTU während der PMTUD gesenkt wird.

Es wird von Hosts verwendet, um schneller zu einem angemessenen Wert für die Sende-MSS zu gelangen, wie in diesem Beispiel gezeigt.

Die PMTUD wird kontinuierlich für alle Pakete durchgeführt, da sich der Pfad zwischen Sender und Empfänger dynamisch ändern kann.

Jedes Mal, wenn ein Absender eine ICMP-Meldung "Cannot Fragment" (Kann nicht fragmentieren) empfängt, aktualisiert er die Routing-Informationen (in denen er die PMTUD speichert).

Während der PMTUD können zwei Dinge passieren:

1. Das Paket kann bis zum Empfänger gelangen, ohne fragmentiert zu werden.

---

> **Hinweis:** Damit ein Router die CPU vor DoS-Angriffen schützen kann, drosselt er die Anzahl der nicht erreichbaren ICMP-Nachrichten, die er senden würde, auf zwei pro Sekunde. Wenn Sie in diesem Zusammenhang ein Netzwerkszenario haben, in dem Sie erwarten, dass der Router mit mehr als zwei ICMP-Nachrichten (Typ = 3, Code = 4) pro Sekunde antworten muss (kann verschiedene Hosts sein), deaktivieren Sie die Drosselung von ICMP-Nachrichten mit dem **no ip icmp rate-limit unreachable [df] interface** aus.

---

2. Der Sender erhält ICMP-Meldungen "Cannot Fragment" (Fragmentierung nicht möglich) von Hops entlang des Pfads zum Empfänger.

Die PMTUD wird unabhängig für beide Richtungen eines TCP-Flows durchgeführt.

Es gibt Fälle, in denen die PMTUD in einer Richtung eines Flusses eine der Endstationen veranlasst, die Sende-MSS zu senken, und die andere Endstation die ursprüngliche Sende-MSS beibehält, da sie nie ein IPv4-Datagramm gesendet hat, das groß genug ist, um die PMTUD auszulösen.

Ein Beispiel ist die in Beispiel 3 dargestellte HTTP-Verbindung. Der TCP-Client sendet kleine und der Server große Pakete.

In diesem Fall lösen nur die großen Pakete vom Server (größer als 576 Byte) die PMTUD aus.

Die Pakete vom Client sind klein (weniger als 576 Byte) und lösen keine PMTUD aus, da sie keine Fragmentierung erfordern, um über die 576-MTU-Verbindung zu gelangen.

**Beispiel 3**

Beispiel 4 zeigt ein Beispiel für asymmetrisches Routing, bei dem einer der Pfade eine kleinere MTU als der andere hat.

Asymmetrisches Routing tritt auf, wenn zum Senden und Empfangen von Daten zwischen zwei Endpunkten verschiedene Pfade verwendet werden.

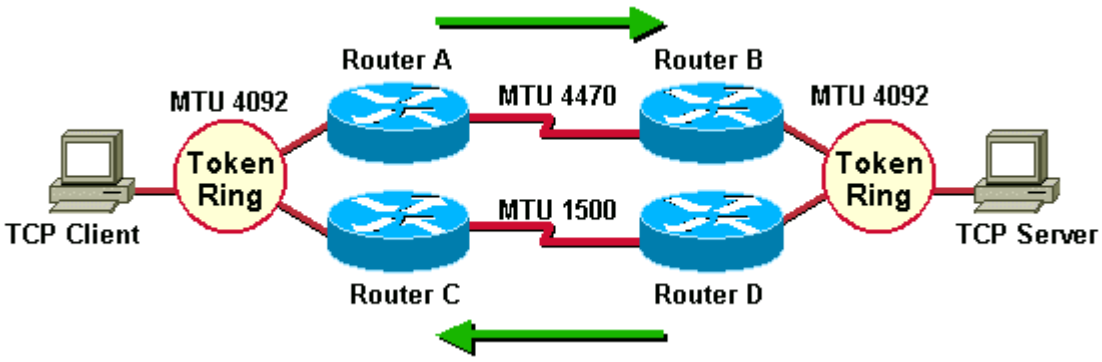
In diesem Beispiel löst die PMTUD das Absenken der Sende-MSS nur in eine Richtung eines TCP-Flusses aus.

Der Datenverkehr vom TCP-Client zum Server fließt über Router A und Router B, während der Datenverkehr vom Server zum Client über Router D und Router C fließt.

Wenn der TCP-Server Pakete an den Client sendet, veranlasst die PMTUD den Server, die Sende-MSS zu senken, da Router D die 4092-Byte-Pakete fragmentieren muss, bevor er sie an Router C senden kann.

Umgekehrt erhält der Client nie eine ICMP-Meldung "Destination Unreachable" mit dem Code, der "fragmentation needed and DF set" angibt, da Router A Pakete nicht fragmentieren muss, wenn er sie über Router B an den Server sendet.

**Beispiel 4**




---

**Hinweis**: Der Befehl **ip tcp path-mtu-discovery** wird verwendet, um die TCP-MTU-Pfaderkennung für TCP-Verbindungen zu aktivieren, die von Routern (z. B. BGP und Telnet) initiiert werden.

---

## Probleme mit der PMTUD

Diese Faktoren können die PMTUD unterbrechen.

- Ein Router verwirft ein Paket und sendet keine ICMP-Nachricht. (Ungewöhnlich)
- Ein Router generiert und sendet eine ICMP-Nachricht, die jedoch von einem Router oder einer Firewall zwischen diesem Router und dem Absender blockiert wird. (Häufig)
- Ein Router generiert und sendet eine ICMP-Nachricht, die jedoch vom Sender ignoriert wird. (Ungewöhnlich)

Der erste und der letzte der drei Aufzählungszeichen hier sind normalerweise das Ergebnis eines Fehlers, aber der mittlere Aufzählungszeichen beschreibt ein häufiges Problem.

Diejenigen, die ICMP-Paketfilter implementieren, neigen dazu, alle ICMP-Nachrichtentypen zu blockieren, anstatt nur bestimmte ICMP-Nachrichtentypen zu blockieren.

Paketfilter können alle ICMP-Nachrichtentypen blockieren, mit Ausnahme derjenigen, die "nicht erreichbar" oder "Zeitüberschreitung" sind.

Der Erfolg oder Misserfolg der PMTUD hängt davon ab, ob ICMP-Nachrichten, die nicht erreichbar sind, an den Absender eines TCP/IPv4-Pakets weitergeleitet werden.

â€žICMP time-exceededâ€œ-Meldungen sind wichtig für andere IPv4-Probleme.

Ein Beispiel für einen solchen Paketfilter, der auf einem Router implementiert ist, ist hier dargestellt.

```
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
```

Es gibt andere Techniken, die verwendet werden können, um das Problem eines vollständig blockierten ICMP zu beheben.

- Löschen Sie das DF-Bit auf dem Router, und lassen Sie die Fragmentierung zu. (Dies ist jedoch keine gute Idee. weitere Informationen finden Sie unter Probleme mit der IP-Fragmentierung).
- Bearbeiten Sie den Wert der TCP-MSS-Option mithilfe des Schnittstellenbefehls. **ip tcp adjust-mss <500-1460>.**

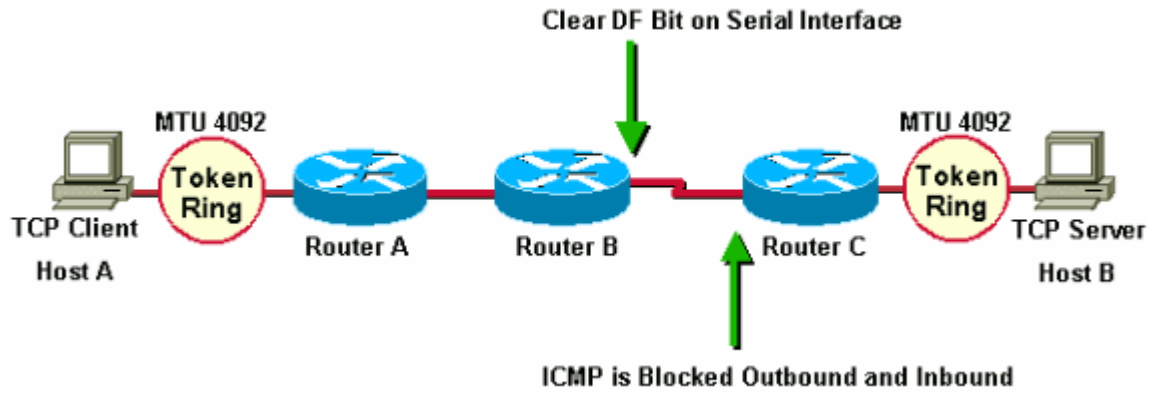
Im nächsten Beispiel befinden sich Router A und Router B in derselben administrativen Domäne. Router C ist nicht erreichbar und blockiert ICMP, sodass die PMTUD unterbrochen wird.

Eine Problemumgehungen für diese Situation ist das Löschen des DF-Bits in beide Richtungen auf Router B, um eine Fragmentierung zu ermöglichen. Dies kann mithilfe von richtlinienbasiertem Routing erfolgen.

Die Syntax zum Löschen des DF-Bits ist in Cisco IOS® Softwareversion 12.1(6) und höher verfügbar.

```
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
```




Eine weitere Möglichkeit besteht darin, den Wert der TCP-MSS-Option auf SYN-Paketen zu ändern, die den Router durchlaufen (verfügbar in Cisco IOS® 12.2(4)T und höher).

Dadurch wird der Wert der MSS-Option im TCP-SYN-Paket reduziert, sodass er kleiner ist als der Wert (1460) im **ip tcp adjust-mss**aus.

Das Ergebnis ist, dass der TCP-Sender Segmente sendet, die nicht größer als dieser Wert sind.

Die IPv4-Paketgröße ist 40 Byte größer (1500) als der MSS-Wert (1460 Byte), um den TCP-Header (20 Byte) und den IPv4-Header (20 Byte) zu berücksichtigen.

Sie können die MSS von TCP-SYN-Paketen mit dem **ip tcp adjust-mss** aus. Diese Syntax reduziert den MSS-Wert auf TCP-Segmenten auf 1460.

Dieser Befehl hat Auswirkungen auf den Eingangs- und Ausgangs-Datenverkehr auf Schnittstelle serial0.

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

IPv4-Fragmentierungsprobleme haben sich seit der zunehmenden Verbreitung von IPv4-Tunneln immer weiter ausgebreitet.

Tunnel verursachen mehr Fragmentierung, da die Tunnelkapselung die Größe eines Pakets um einen "Overhead" erhöht.

Beispielsweise werden durch die Hinzufügung von Generic Router Encapsulation (GRE) 24 Byte zu einem Paket hinzugefügt. Nach dieser Erhöhung muss das Paket fragmentiert werden, da es größer ist als die MTU für ausgehende Pakete.

## Häufig vorkommende Netzwerktopologien, die PMTUD benötigen

Die PMTUD wird in Netzwerksituationen benötigt, in denen Zwischenverbindungen kleinere MTUs haben als die MTU der Endverbindungen. Einige häufige Gründe für die Existenz dieser kleineren MTU in Verbindungen sind:

- Via Token Ring (oder FDDI) verbundene Endhosts mit einer Ethernet-Verbindung dazwischen. Die Token Ring (oder FDDI)-MTUs an den Enden sind größer als die Ethernet-MTU in der Mitte.
- PPPoE (oft mit ADSL verwendet) benötigt 8 Byte für seinen Header. Dies reduziert die effektive

MTU des Ethernet auf 1492 (1500 - 8).

Tunnelprotokolle wie GRE, IPv4sec und L2TP benötigen ebenfalls Platz für ihre jeweiligen Header und Trailer. Dadurch wird auch die effektive MTU der Ausgangsschnittstelle verringert.

# Tunnel

Ein Tunnel ist eine logische Schnittstelle auf einem Cisco Router, die eine Möglichkeit bietet, Passagierpakete innerhalb eines Transportprotokolls zu kapseln.

Es handelt sich um eine Architektur, die darauf ausgelegt ist, Services bereitzustellen, um ein Punkt-zu-Punkt-Kapselungsschema zu implementieren. Tunnelschnittstellen haben die folgenden drei Hauptkomponenten:

- Passagierprotokoll (AppleTalk, Banyan VINES, CLNS, DECnet, IPv4 oder IPX)
- Trägerprotokoll â€“ eines der folgenden Kapselprotokolle:
  - GRE = Cisco Multiprotocol Carrier Protocol Weitere Informationen finden Sie unter RFC 2784 und RFC 1701.
  - IPv4 in IPv4-Tunneln â€“ weitere Informationen finden Sie unter RFC 2003.
- Transportprotokoll â€“ das Protokoll, das für die Übertragung des Kapselprotokolls verwendet wird.

Die in diesem Abschnitt dargestellten Pakete veranschaulichen die IPv4-Tunneling-Konzepte, wobei GRE das Kapselprotokoll und IPv4 das Transportprotokoll ist.

Das Passagierprotokoll ist ebenfalls IPv4. In diesem Fall ist IPv4 sowohl das Transport- als auch das Passagierprotokoll.

Normales Paket

| IPv4 | TCP | Telnet |
|---|---|---|

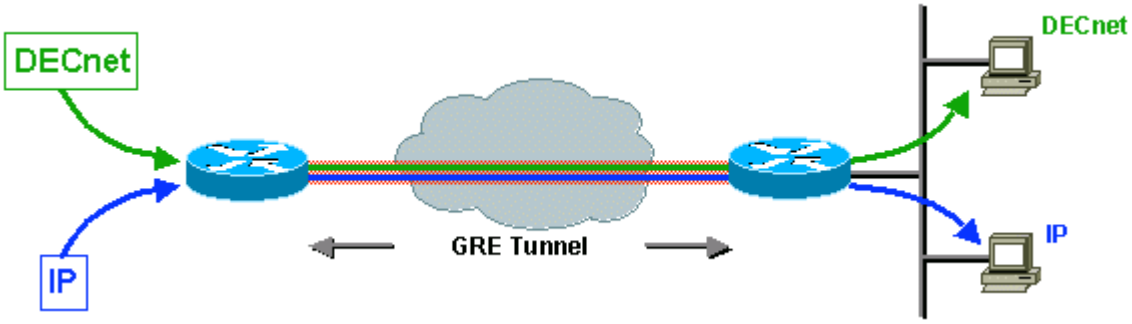
Tunnel-Paket

| IPv4 | GRE | IPv4 | TCP | Telnet |
|---|---|---|---|---|

- IPv4 ist das Transportprotokoll.
- GRE ist das Kapselprotokoll.
- IPv4 ist das Passagierprotokoll.

Das nächste Beispiel zeigt die Kapselung von IPv4 und DECnet als Passagierprotokolle mit GRE als Trägerprotokoll.

Dies veranschaulicht die Möglichkeit, dass Trägerprotokolle mehrere Passagierprotokolle kapseln, wie im Bild dargestellt.




Ein Netzwerkadministrator erwägt das Tunneling in einer Situation, in der es zwei nicht zusammenhängende Nicht-IPv4-Netzwerke gibt, die durch einen IPv4-Backbone getrennt sind.

Wenn die nicht zusammenhängenden Netzwerke DECnet ausführen, kann der Administrator entscheiden, ob er diese miteinander verbindet (oder nicht), indem er DECnet im Backbone konfiguriert.

Der Administrator möchte nicht zulassen, dass das DECnet-Routing Backbone-Bandbreite beansprucht, da dies die Leistung des IPv4-Netzwerks beeinträchtigen könnte.

Eine sinnvolle Alternative ist ein Tunneling von DECnet über den IPv4-Backbone. Die Tunnellösung kapselt die DECnet-Pakete in IPv4 und sendet sie über den Backbone an den Tunnel-Endpunkt, wo die Kapselung entfernt wird und die DECnet-Pakete über DECnet an ihr Ziel geleitet werden.

Es gibt Vorteile, Datenverkehr in ein anderes Protokoll zu kapseln:

- Die Endpunkte verwenden private Adressen (RFC 1918) und der Backbone unterstützt kein Routing dieser Adressen.
- Ermöglichung virtueller privater Netzwerke (VPNs) über WANs oder das gesamte Internet hinweg.
- Verbindung nicht zusammenhängender Multiprotocol-Netzwerke über einen einzigen Protokoll-Backbone.
- Verschlüsselung des Datenverkehrs über den Backbone oder das Internet.

Im Folgenden wird IPv4 als Passagierprotokoll und IPv4 als Transportprotokoll verwendet.

## Überlegungen zu Tunnelschnittstellen

Folgende Überlegungen sollten Sie beim Tunneling anstellen.

- Fast Switching von GRE-Tunneln wurde in Cisco IOS® Version 11.1 und CEF Switching in Version 12.0 eingeführt.
- CEF-Switching für Multi-Point-GRE-Tunnel wurde in Version 12.2(8)T eingeführt.
- Kapselung und Entkapselung an Tunnel-Endpunkten waren in früheren Versionen von Cisco IOS®, als nur Prozess-Switching unterstützt wurde, langsame Vorgänge.
- Beim Tunneling von Paketen treten Sicherheits- und Topologieprobleme auf. Tunnel können Zugriffskontrolllisten (ACLs) und Firewalls umgehen.
- Wenn Sie durch eine Firewall tunneln, umgehen Sie das Passagierprotokoll, das getunnelt wird. Daher wird empfohlen, an den Tunnelendpunkten Firewall-Funktionalität zu integrieren, um die Einhaltung der Richtlinien für die Passagierprotokolle durchzusetzen.

- Tunneling führt aufgrund erhöhter Latenz zu Problemen mit Transportprotokollen mit begrenzten Timern (beispielsweise DECnet).

- Tunneling über Umgebungen mit unterschiedlichen Geschwindigkeiten, wie schnelle FDDI-Ringe und über langsame Telefonleitungen mit 9600 Bit/s, führt zu Problemen bei der Paketneuordnung. Einige Passagierprotokolle funktionieren in gemischten Netzwerken schlecht.

- Punkt-zu-Punkt-Tunnel verbrauchen Bandbreite auf einer physischen Verbindung. Über mehrere Punkt-zu-Punkt-Tunnel verfügt jede Tunnelschnittstelle über eine Bandbreite und die physische Schnittstelle, über die der Tunnel verläuft, über eine Bandbreite. Legen Sie beispielsweise die Tunnelbandbreite auf 100 KB fest, wenn 100 Tunnel über eine 10-MB-Verbindung betrieben werden. Die Standardbandbreite für einen Tunnel beträgt 9 kbit.

- Routing-Protokolle ziehen einen Tunnel einer echten Verbindung vor, da der Tunnel trügerisch eine One-Hop-Verbindung mit dem kostengünstigsten Pfad zu sein scheint, obwohl er mehr Hops und daher teurer ist als ein anderer Pfad. Dies wird durch die ordnungsgemäße Konfiguration des Routing-Protokolls verhindert. Ziehen Sie für die Tunnelschnittstelle ein anderes Routing-Protokoll in Betracht als das Routing-Protokoll, das auf der physischen Schnittstelle ausgeführt wird.

- Probleme mit rekursivem Routing werden vermieden, indem geeignete statische Routen zum Tunnelziel konfiguriert werden. Eine rekursive Route bedeutet, dass der beste Weg zum Tunnelziel durch den Tunnel selbst führt. Diese Situation führt zu einem Auf-und-ab-Bouncing der Tunnelschnittstelle. Dieser Fehler tritt auf, wenn ein rekursives Routing-Problem vorliegt.

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

## Router als PMTUD-Teilnehmer am Endpunkt des Tunnels

Der Router hat zwei verschiedene PMTUD-Rollen, wenn er Endpunkt eines Tunnels ist.

- In der ersten Rolle ist der Router der Forwarder eines Host-Pakets. Für die PMTUD-Verarbeitung muss der Router das DF-Bit und die Paketgröße des ursprünglichen Datenpakets überprüfen und gegebenenfalls entsprechende Maßnahmen ergreifen.

- Die zweite Rolle kommt ins Spiel, nachdem der Router das ursprüngliche IPv4-Paket in das Tunnelpaket gekapselt hat. In dieser Phase verhält sich der Router eher wie ein Host in Bezug auf die PMTUD und das Tunnel-IPv4-Paket.

Wenn der Router in der ersten Rolle agiert (ein Router, der Host-IPv4-Pakete weiterleitet), kommt diese Rolle ins Spiel, bevor der Router das Host-IPv4-Paket in das Tunnelpaket einkapselt.

Wenn der Router als Forwarder eines Host-Pakets teilnimmt, führt er die folgenden Aktionen aus:

- Überprüft, ob das DF-Bit festgelegt ist.

- Überprüft, welche Paketgröße der Tunnel aufnehmen kann.

- Fragmentieren (wenn das Paket zu groß und das DF-Bit nicht festgelegt ist), Einkapseln von Fragmenten und Senden oder

- Verwirft das Paket (wenn das Paket zu groß und das DF-Bit festgelegt ist) und sendet eine ICMP-

Meldung an den Sender.

- Kapselt (wenn das Paket nicht zu groß ist) und sendet.

Im Allgemeinen besteht die Wahl zwischen Kapselung und dann Fragmentierung (Senden zweier Kapselungsfragmente) oder Fragmentierung und dann Kapselung (Senden zweier gekapselter Fragmente).

**In** diesem Abschnitt werden **zwei Beispiele** beschrieben, die die Interaktion von PMTUD und Paketen zeigen, die Beispielnetzwerke durchlaufen.

Das erste Beispiel zeigt, was mit einem Paket passiert, wenn der Router (an der Tunnelquelle) als aktiver Router fungiert.

Um die PMTUD zu verarbeiten, muss der Router das DF-Bit und die Paketgröße des ursprünglichen Datenpakets überprüfen und entsprechende Maßnahmen ergreifen.

In diesem Beispiel wird die GRE-Kapselung für den Tunnel verwendet. GRE führt vor der Kapselung eine Fragmentierung durch.

Spätere Beispiele zeigen Szenarien, in denen die Fragmentierung nach der Verkapselung erfolgt.

In Beispiel 1 ist das DF-Bit nicht festgelegt (DF = 0) und die IPv4-MTU des GRE-Tunnels ist 1476 (1500 - 24).

**Beispiel 1**

1. Der Forwarding-Router (an der Tunnelquelle) empfängt vom sendenden Host ein 1500-Byte-Datagramm mit freiem DF-Bit (DF = 0).

Dieses Datagramm besteht aus einem 20-Byte-IP-Header und einer 1480-Byte-TCP-Nutzlast.

| IPv4 | 1480 Byte TCP + Daten |
|---|---|

2. Da das Paket für die IPv4-MTU zu groß ist, nachdem der GRE-Overhead (24 Byte) hinzugefügt wurde, teilt der Forwarding-Router das Datagramm in zwei Fragmente auf: 1476 (20 Byte IPv4-Header + 1456 Byte IPv4-Nutzlast) und 44 Byte (20 Byte IPv4-Header + 24 Byte IPv4-Payload)

Nach Hinzufügen der GRE-Kapselung ist das Paket nicht größer als die MTU der ausgehenden physischen Schnittstelle.

| $IP_0$ | 1456 Byte TCP + Daten |
|---|---|
| $IP_1$ | 24 Byte Daten |

3. Der weiterleitende Router fügt jedem Fragment des ursprünglichen IPv4-Datagramms eine GRE-Kapselung hinzu, die einen 4-Byte-GRE-Header und einen 20-Byte-IPv4-Header umfasst.

Diese beiden IPv4-Datagramme haben nun eine Länge von 1500 bzw. 68 Byte und werden als individuelle IPv4-Datagramme und nicht als Fragmente betrachtet.

| IPv4 | GRE | $IP_0$ | 1456 Byte TCP + Daten |
|---|---|---|---|
| IPv4 | GRE | $IP_1$ | 24 Byte Daten |

4. Der Tunnel-Zielrouter entfernt die GRE-Kapselung von jedem Fragment des ursprünglichen Datagramms,

wodurch zwei IPv4-Fragmente der Längen 1476 und 24 Byte übrig bleiben.

Diese IPv4-Datagrammfragmente werden von diesem Router separat an den empfangenden Host weitergeleitet.

| $IP_0$ | 1456 Byte TCP + Daten |
|---|---|

| $IP_1$ | 24 Byte Daten |
|---|---|

5. Der empfangende Host reassembliert diese beiden Fragmente in das ursprüngliche Datagramm.

| IPv4 | 1480 Byte TCP + Daten |
|---|---|

Beispiel 2 zeigt die Rolle des weiterleitenden Routers im Kontext einer Netzwerktopologie.

Der Router agiert in derselben Rolle wie der weiterleitende Router, aber diesmal ist das DF-Bit festgelegt (DF = 1).

**Beispiel 2**

1. Der Forwarding-Router an der Tunnelquelle empfängt vom sendenden Host ein 1500-Byte-Datagramm mit DF = 1.

| IPv4 | 1480 Byte TCP + Daten |
|---|---|

2. Da das DF-Bit festgelegt ist und die Datagrammgröße (1500 Byte) größer ist als die IPv4-MTU des GRE-Tunnels (1476), verwirft der Router das Datagramm und sendet eine Meldung mit dem Vermerk "ICMP fragmentation needed but DF bit set" an die Quelle des Datagramms.

Die ICMP-Meldung benachrichtigt den Sender, dass die MTU 1476 beträgt.

| IPv4 | ICMP MTU 1476 |
|---|---|

3. Der sendende Host empfängt die ICMP-Nachricht, und wenn er die ursprünglichen Daten erneut sendet, verwendet er ein IPv4-Datagramm mit 1476 Byte.

| IPv4 | 1456 Byte TCP + Daten |
|---|---|

4. Diese IPv4-Datagrammlänge (1476 Byte) entspricht jetzt dem Wert der IPv4-MTU des GRE-Tunnels, sodass der Router die GRE-Kapselung zum IPv4-Datagramm hinzufügt.

| IPv4 | GRE | IPv4 | 1456 Byte TCP + Daten |
|---|---|---|---|

5. Der empfangende Router (am Tunnelziel) entfernt die GRE-Kapselung des IPv4-Datagramms und sendet sie an den empfangenden Host.

| IPv4 | 1456 Byte TCP + Daten |
|---|---|

Dies geschieht, wenn der Router in Bezug auf die PMTUD und das Tunnel-IPv4-Paket in der zweiten Rolle als sendender Host fungiert.

Diese Rolle kommt ins Spiel, nachdem der Router das ursprüngliche IPv4-Paket in das Tunnelpaket gekapselt hat.

**Hinweis**: Standardmäßig führt ein Router keine PMTUD für die von ihm generierten GRE-Tunnelpakete aus. Die Fehlermeldung **tunnel path-mtu-discovery** kann verwendet werden, um die PMTUD für GRE-IPv4-Tunnelpakete zu aktivieren.

Beispiel 3 zeigt, was passiert, wenn der Host IPv4-Datagramme sendet, deren Größe unterhalb der IPv4-MTU auf der GRE-Tunnelschnittstelle liegt.

Das DF-Bit kann in diesem Fall entweder festgelegt oder nicht festgelegt (1 oder 0) sein.

Die GRE-Tunnelschnittstelle verfügt nicht über die **tunnel path-mtu-discovery** -Befehl konfiguriert, sodass der Router keine PMTUD auf dem GRE-IPv4-Paket stirbt.

**Beispiel 3**

1. Der Forwarding-Router an der Tunnelquelle empfängt ein 1476 Byte großes Datagramm vom sendenden Host.

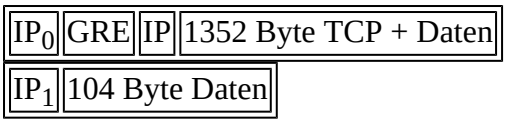

2. Dieser Router kapselt das 1476-Byte-IPv4-Datagramm in die GRE, um ein 1500-Byte-GRE-IPv4-Datagramm zu erhalten.

Das DF-Bit im GRE-IPv4-Header wird gelöscht (DF = 0). Dieser Router leitet dieses Paket dann an das Tunnelziel weiter.


3. Angenommen, es gibt einen Router zwischen Tunnelquelle und -ziel mit einer Verbindungs-MTU von 1400.

Dieser Router fragmentiert das Tunnelpaket, da das DF-Bit klar ist (DF = 0).

Denken Sie daran, dass dieses Beispiel das äußerste IPv4 fragmentiert, sodass die GRE-, inneren IPv4- und TCP-Header nur im ersten Fragment angezeigt werden.




4. Der Tunnel-Zielrouter muss das GRE-Tunnelpaket neu assemblieren.


5. Nachdem das GRE-Tunnelpaket wieder zusammengesetzt wurde, entfernt der Router den GRE-IPv4-Header und sendet das ursprüngliche IPv4-Datagramm auf den Weg.


Beispiel 4 zeigt, was passiert, wenn der Router in Bezug auf die PMTUD und das Tunnel-IPv4-Paket in der Rolle eines sendenden Hosts agiert.

Diesmal wird das DF-Bit im ursprünglichen IPv4-Header festgelegt (DF = 1) und **tunnel path-mtu-discovery** wurde so konfiguriert, dass das DF-Bit vom inneren IPv4-Header in den äußeren (GRE + IPv4)-Header

kopiert wird.

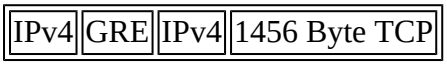
**Beispiel 4**

1. Der Forwarding-Router an der Tunnelquelle empfängt vom sendenden Host ein 1476-Byte-Datagramm mit DF = 1.


2. Dieser Router kapselt das 1476-Byte-IPv4-Datagramm in die GRE, um ein 1500-Byte-GRE-IPv4-Datagramm zu erhalten.

Für diesen GRE-IPv4-Header ist das DF-Bit festgelegt (DF = 1), da das DF-Bit im ursprünglichen IPv4-Datagramm festgelegt war.

Dieser Router leitet dieses Paket dann an das Tunnelziel weiter.




3. Nehmen wir erneut an, es gibt einen Router zwischen Tunnelquelle und -ziel mit einer Verbindungs-MTU von 1400.

Dieser Router fragmentiert das Tunnelpaket nicht, da das DF-Bit festgelegt ist (DF=1).

Dieser Router muss das Paket verwerfen und eine ICMP-Fehlermeldung an den Tunnel-Quellrouter senden, da dies die IPv4-Quelladresse im Paket ist.


4. Der Forwarding-Router an der Tunnelquelle erhält diese "ICMP"-Fehlermeldung und senkt die IPv4-MTU des GRE-Tunnels auf 1376 (1400 - 24).

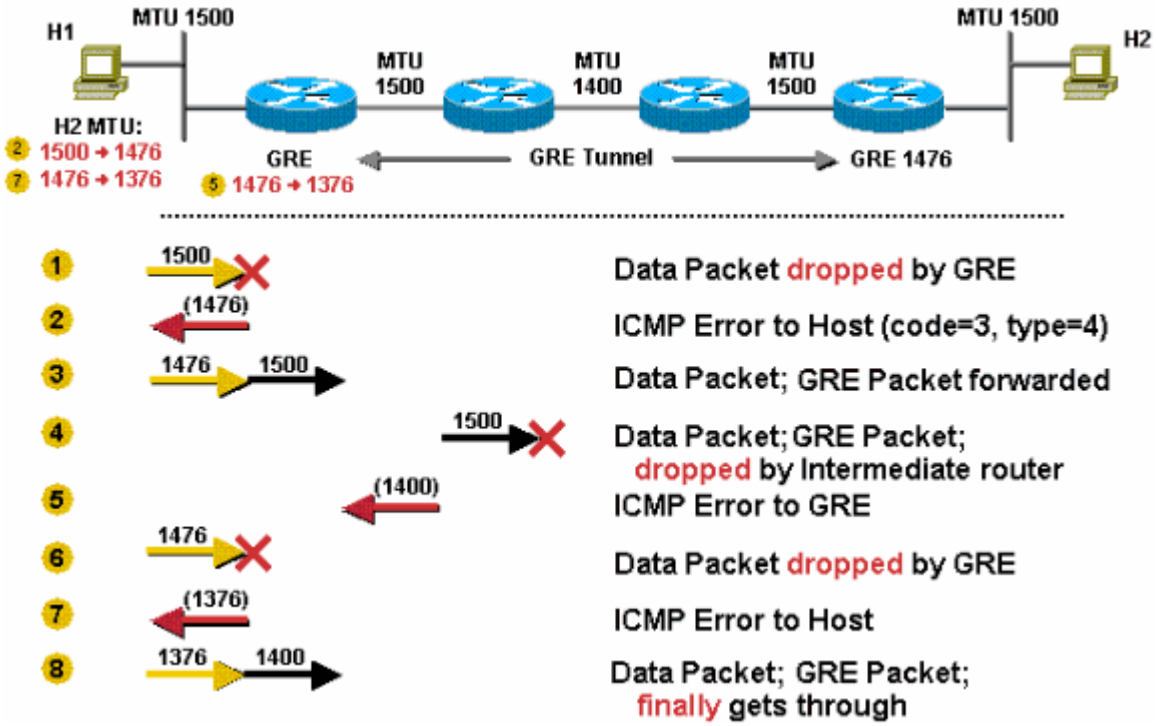
Wenn der sendende Host die Daten das nächste Mal in einem 1476-Byte-IPv4-Paket erneut sendet, kann dieses Paket zu groß sein, und dieser Router sendet dann eine "ICMP"-Fehlermeldung mit einem MTU-Wert von 1376 an den Sender.

Wenn der sendende Host die Daten erneut sendet, sendet er sie in einem 1376-Byte-IPv4-Paket, und dieses Paket leitet sie durch den GRE-Tunnel zum empfangenden Host weiter.

**Beispiel 5**

Dieses Beispiel veranschaulicht die GRE-Fragmentierung. Fragment vor der Kapselung für GRE, dann PMTUD für das Datenpaket, und das DF-Bit wird nicht kopiert, wenn das IPv4-Paket von GRE gekapselt wird.

Das DF-Bit ist nicht festgelegt. Die IPv4-MTU der GRE-Tunnelschnittstelle ist standardmäßig 24 Byte kleiner als die IPv4-MTU der physischen Schnittstelle. Somit beträgt die IPv4-MTU der GRE-Schnittstelle 1476, wie im Bild gezeigt.



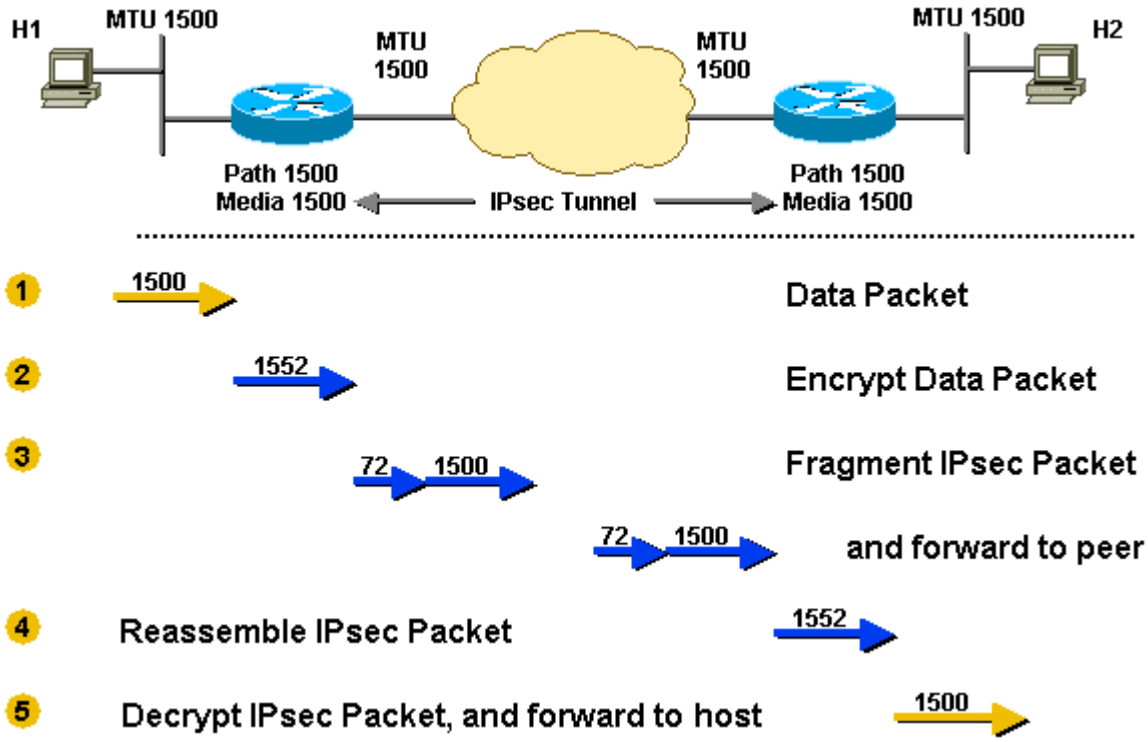

1. Der Sender sendet ein 1500-Byte-Paket (20 Byte IPv4-Header + 1480 Byte TCP-Nutzlast).
2. Da die MTU des GRE-Tunnels 1476 beträgt, wird das 1500-Byte-Paket in zwei IPv4-Fragmente von 1476 und 44 Byte aufgeteilt, die jeweils die zusätzlichen 24 Byte des GRE-Headers erwarten.
3. Die 24 Byte des GRE-Headers werden zu jedem IPv4-Fragment hinzugefügt. Nun beträgt die Fragmentgröße 1500 (1476 + 24) und 68 (44 + 24) Byte.
4. Die GRE + IPv4-Pakete, die die beiden IPv4-Fragmente enthalten, werden an den GRE-Tunnel-Peer-Router weitergeleitet.
5. Der GRE-Tunnel-Peer-Router entfernt die GRE-Header aus den beiden Paketen.
6. Dieser Router leitet die beiden Pakete an den Zielhost weiter.
7. Der Zielhost reassembliert die IPv4-Fragmente wieder zum ursprünglichen IPv4-Datagramm.

**Beispiel 6**

Dieses Beispiel ähnelt Beispiel 5, aber diesmal ist das DF-Bit festgelegt. Der Router ist so konfiguriert, dass er PMTUD auf GRE + IPv4-Tunnelpaketen mit dem **tunnel path-mtu-discovery**-Befehl, und das DF-Bit wird vom ursprünglichen IPv4-Header in den GRE-IPv4-Header kopiert.

Wenn der Router einen ICMP-Fehler für das GRE + IPv4-Paket erhält, reduziert er die IPv4-MTU auf der GRE-Tunnelschnittstelle.

Die IPv4-MTU des GRE-Tunnels ist standardmäßig auf 24 Byte kleiner als die MTU der physischen Schnittstelle festgelegt, sodass die IPv4-MTU des GRE hier 1476 beträgt. Im GRE-Tunnelpfad befindet sich eine Verbindung mit 1400 MTU, wie im Bild gezeigt.




1. Der Router empfängt ein 1500-Byte-Paket (20 Byte IPv4-Header + 1480 Byte TCP-Nutzlast) und verwirft dieses Paket. Dies liegt daran, dass es größer ist als die IPv4-MTU auf der GRE-Tunnelschnittstelle (1476).
2. Der Router sendet einen ICMP-Fehler an den Sender und teilt ihm mit, dass die Next-Hop-MTU 1476 beträgt. Der Host zeichnet diese Informationen auf, in der Regel als Host-Route für das Ziel in seiner Routing-Tabelle.
3. Der sendende Host verwendet beim erneuten Senden der Daten eine Paketgröße von 1476 Byte. Der GRE-Router fügt 24 Byte GRE-Kapselung hinzu und sendet ein 1500-Byte-Paket.
4. Das 1500-Byte-Paket kann die 1400-Byte-Verbindung nicht durchlaufen, sodass es vom Zwischenrouter verworfen wird.
5. Der Zwischenrouter sendet ein ICMP (Typ = 3, Code = 4) mit einer Next-Hop-MTU von 1400 an den GRE-Router. Der GRE-Router reduziert dies auf 1376 (1400 - 24) und legt einen internen IPv4-MTU-Wert auf der GRE-Schnittstelle fest. Diese Änderung ist nur sichtbar, wenn Sie das **debug tunnel** -Befehls; in der Ausgabe des Befehls **show ip interface tunnel**<#> aus.
6. Wenn der Host das 1476-Byte-Paket das nächste Mal sendet, verwirft der GRE-Router das Paket, da es größer ist als die aktuelle IPv4-MTU (1376) an der GRE-Tunnelschnittstelle.
7. Der GRE-Router sendet ein weiteres ICMP (Typ = 3, Code = 4) mit einer Next-Hop-MTU von 1376 an den Sender, und der Host aktualisiert seine aktuellen Informationen mit einem neuen Wert.
8. Der Host sendet die Daten erneut, aber nun fügt die GRE in einem kleineren 1376-Byte-Paket 24 Byte Kapselung hinzu und leitet sie weiter. Dieses Mal gelangt das Paket zum GRE-Tunnel-Peer, wo es entkapselt und an den Zielhost gesendet wird.

---

> **Hinweis**: Wenn der **tunnel path-mtu-discovery** wurde in diesem Szenario nicht auf dem Forwarding-Router konfiguriert, und das DF-Bit wurde in den über den GRE-Tunnel weitergeleiteten Paketen festgelegt. Host 1 kann weiterhin erfolgreich TCP/IPv4-Pakete an Host 2 senden, aber sie werden in der Mitte an der Verbindung mit einer MTU von 1400 fragmentiert. Auch der GRE-Tunnel-Peer muss sie wieder zusammenbauen, bevor er sie entkapseln und weiterleiten kann.

---

# Reiner IPsec-Tunnel-Modus

Das IPv4 Security (IPv4sec)-Protokoll ist eine standardbasierte Methode, die Datenschutz, Integrität und

Authentizität für Informationen bietet, die über IPv4-Netzwerke übertragen werden.

IPv4sec bietet IPv4-Verschlüsselung auf Netzwerkebene. IPv4sec verlängert das IPv4-Paket durch Hinzufügen von mindestens einem IPv4-Header (Tunnelmodus).

Die hinzugefügten Header variieren in Abhängigkeit vom IPv4sec-Konfigurationsmodus in der Länge, überschreiten jedoch nicht ~58 Byte (Encapsulating Security Payload (ESP) und ESP authentication (ESPauth)) pro Paket.

IPv4sec kennt zwei Betriebsmodi: Tunnelmodus und Transportmodus.

1. Der Standardmodus ist der Tunnelmodus. Im Tunnelmodus ist das gesamte ursprüngliche IPv4-Paket geschützt (verschlüsselt, authentifiziert oder beides) und durch die IPv4sec-Header und -Trailer gekapselt. Dann wird dem Paket ein neuer IPv4-Header vorangestellt, der die IPv4sec-Endpunkte (Peers) als Quelle und Ziel angibt. Der Tunnelmodus kann mit jedem Unicast-IPv4-Datenverkehr verwendet werden und muss verwendet werden, wenn IPv4sec den Datenverkehr von Hosts hinter den IPv4sec-Peers schützt. Der Tunnelmodus wird beispielsweise bei virtuellen privaten Netzwerken (VPNs) verwendet, bei denen Hosts in einem geschützten Netzwerk Pakete über ein Paar von IPv4sec-Peers an Hosts in einem anderen geschützten Netzwerk senden. Bei VPNs schützt der IPv4sec-â€žTunnelâ€œ den IPv4-Datenverkehr zwischen Hosts, indem er ihn zwischen den IPv4sec-Peer-Routern verschlüsselt.
2. Mit Transportmodus (konfiguriert mit dem Unterbefehl) **mode transport**, auf der Transformationsdefinition), wird nur die Nutzlast des ursprünglichen IPv4-Pakets geschützt (verschlüsselt, authentifiziert oder beides). Die Nutzlast wird durch IPv4sec-Header und -Trailer gekapselt. Die ursprünglichen IPv4-Header bleiben intakt, mit der Ausnahme, dass das IPv4-Protokollfeld auf ESP (50) geändert wird und der ursprüngliche Protokollwert im IPv4sec-Trailer gespeichert und bei der Entschlüsselung des Pakets wiederhergestellt wird. Der Transportmodus wird nur verwendet, wenn der zu schützende IPv4-Verkehr zwischen den IPv4sec-Peers selbst stattfindet. Die Quell- und Ziel-IPv4-Adressen auf dem Paket sind identisch mit den IPv4sec-Peer-Adressen. Normalerweise wird der IPv4sec-Transportmodus nur verwendet, wenn ein anderes Tunneling-Protokoll (wie GRE) verwendet wird, um zuerst das IPv4-Datenpaket zu kapseln, und anschließend IPv4sec zum Schutz der GRE-Tunnelpakete verwendet wird.

IPv4sec führt für Datenpakete und seine eigenen Pakete immer eine PMTUD durch. Es gibt IPv4sec-Konfigurationsbefehle, um die PMTUD-Verarbeitung für das IPv4sec-IPv4-Paket zu modifizieren. IPv4sec kann das DF-Bit im IPv4-Header des Datenpakets löschen, festlegen oder in den IPv4sec-IPv4-Header kopieren. Dies wird als â€žDF Bit Override Functionalityâ€œ bezeichnet.

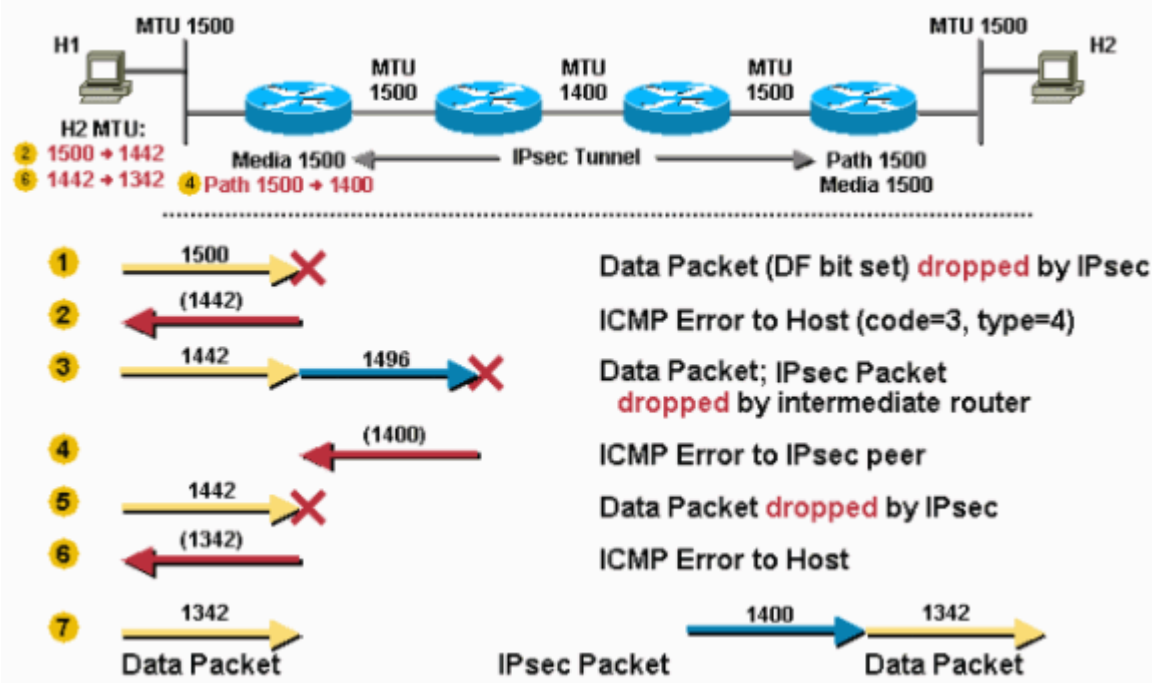
---

> **Hinweis**: Vermeiden Sie die Fragmentierung nach der Kapselung, wenn die Hardwareverschlüsselung mit IPv4sec erfolgt ist. Die Hardwareverschlüsselung ermöglicht einen Durchsatz von ca. 50 Mbit/s, was von der Hardware abhängt. Wenn das IPv4sec-Paket jedoch fragmentiert ist, verlieren Sie 50 bis 90 Prozent des Durchsatzes. Dieser Verlust entsteht, weil die fragmentierten IPv4sec-Pakete zur Reassemblierung per Prozess-Switching weitergeleitet und dann zur Entschlüsselung an das Hardware-Verschlüsselungsmodul übergeben werden. Durch diesen Verlust kann der Durchsatz der Hardwareverschlüsselung auf das Leistungsniveau der Softwareverschlüsselung (2â€“10 Mbit pro Sekunde) sinken.

---

**Beispiel 7**

Dieses Szenario stellt die IPv4sec-Fragmentierung in Aktion dar. In diesem Szenario beträgt die MTU auf dem gesamten Pfad 1500. In diesem Szenario ist das DF-Bit nicht festgelegt.




1. Der Router empfängt ein 1500-Byte-Paket (20 Byte IPv4-Header + 1480 Byte TCP-Nutzlast), das für Host 2 bestimmt ist.
2. Das 1500-Byte-Paket ist mit IPv4sec verschlüsselt und es werden 52 Byte Overhead hinzugefügt (IPv4sec-Header, -Trailer und zusätzlicher IPv4-Header). Jetzt muss IPv4sec ein 1552-Byte-Paket senden. Da die MTU für ausgehende Pakete 1500 beträgt, muss dieses Paket fragmentiert werden.
3. Aus dem IPv4sec-Paket werden zwei Fragmente erstellt. Während der Fragmentierung wird ein zusätzlicher 20-Byte-IPv4-Header für das zweite Fragment hinzugefügt, was zu einem 1500-Byte-Fragment und einem 72-Byte-IPv4-Fragment führt.
4. Der IPv4sec-Tunnel-Peer-Router empfängt die Fragmente, entfernt den zusätzlichen IPv4-Header und reassembliert die IPv4-Fragmente wieder zu dem ursprünglichen IPv4sec-Paket. Dann entschlüsselt IPv4sec dieses Paket.
5. Der Router leitet dann das ursprüngliche 1500-Byte-Datenpaket an Host 2 weiter.

**Beispiel 8**

Dieses Beispiel ähnelt Beispiel 6, mit dem Unterschied, dass in diesem Fall das DF-Bit im ursprünglichen Datenpaket festgelegt ist und es eine Verbindung im Pfad zwischen den IPv4sec-Tunnel-Peers gibt, die eine geringere MTU als die anderen Verbindungen hat.

Dieses Beispiel zeigt, wie der IPv4sec-Peer-Router beide PMTUD-Rollen ausführt, wie im Abschnitt Der Router als PMTUD-Teilnehmer am Endpunkt eines Tunnels beschrieben.

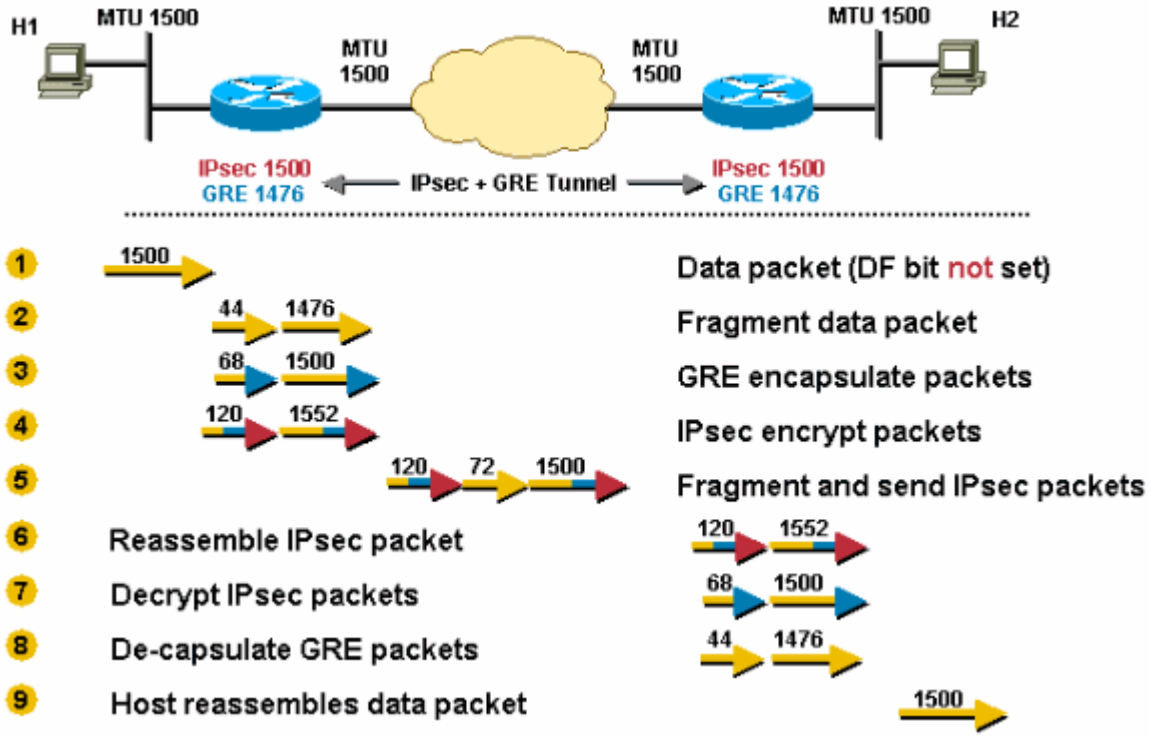
Die IPv4sec-PMTU wird aufgrund der erforderlichen Fragmentierung auf einen niedrigeren Wert geändert.

Das DF-Bit wird vom inneren IPv4-Header in den äußeren IPv4-Header kopiert, wenn IPv4sec ein Paket verschlüsselt.

Die MTU- und PMTU-Werte der Medien werden in der IPv4sec Security Association (SA) gespeichert.

Die MTU des Mediums basiert auf der MTU der Router-Ausgangsschnittstelle und die PMTU auf der kleinsten MTU im Pfad zwischen den IPv4sec-Peers.

IPv4sec kapselt/verschlüsselt das Paket, bevor es wie im Bild gezeigt fragmentiert werden soll.




1. Der Router empfängt ein 1500-Byte-Paket und verwirft es, da das Paket durch den IPv4sec-Overhead beim Hinzufügen größer wird als die PMTU (1500).
2. Der Router sendet eine ICMP-Meldung an Host 1 und teilt ihm mit, dass die Next-Hop-MTU 1442 (1500 - 58 = 1442) beträgt. Diese 58 Byte stellen den maximalen IPv4sec-Overhead dar, wenn Sie IPv4sec ESP und ESPauth verwenden. Der tatsächliche IPv4sec-Overhead ist möglicherweise 7 Byte kleiner als dieser Wert. Host 1 zeichnet diese Informationen, in der Regel als Hostroute für das Ziel (Host 2), in seiner Routingtabelle auf.
3. Host 1 senkt seine PMTU für Host 2 auf 1442, sodass Host 1 kleinere Pakete (1442 Byte) sendet, wenn er die Daten erneut an Host 2 sendet. Der Router empfängt das 1442-Byte-Paket und IPv4sec fügt 52 Byte Verschlüsselungs-Overhead hinzu, sodass das resultierende IPv4sec-Paket 1496 Byte groß ist. Da im Header dieses Pakets das DF-Bit festgelegt ist, wird es vom mittleren Router, dessen Verbindungs-MTU 1400 Byte beträgt, verworfen.
4. Der mittlere Router, der das Paket verworfen hat, sendet eine ICMP-Meldung an den Sender des IPv4sec-Pakets (den ersten Router) und teilt ihm mit, dass die Next-Hop-MTU 1400 Byte beträgt. Dieser Wert wird in der IPv4sec-SA-PMTU aufgezeichnet.
5. Wenn Host 1 das 1442-Byte-Paket das nächste Mal sendet (er hat keine Bestätigung dafür erhalten), verwirft IPv4sec das Paket. Der Router verwirft das Paket, da es aufgrund des IPv4sec-Overheads beim Hinzufügen zum Paket größer ist als die PMTU (1400).
6. Der Router sendet eine ICMP-Meldung an Host 1 und teilt ihm mit, dass die Next-Hop-MTU nun 1342 beträgt. (1400 - 58 = 1342). Host 1 zeichnet diese Informationen erneut auf.
7. Wenn Host 1 die Daten erneut sendet, verwendet er das kleinere Paket (1342). Dieses Paket erfordert keine Fragmentierung und gelangt über den IPv4sec-Tunnel zu Host 2.

# GRE und IPv4sec zusammen

Komplexere Interaktionen für Fragmentierung und PMTUD treten auf, wenn IPv4sec zur Verschlüsselung von GRE-Tunneln verwendet wird.

IPv4sec und GRE werden auf diese Weise kombiniert, da IPv4sec keine IPv4-Multicast-Pakete unterstützt, was bedeutet, dass Sie kein dynamisches Routing-Protokoll über das IPv4sec-VPN-Netzwerk ausführen können.

GRE-Tunnel unterstützen Multicast, sodass ein GRE-Tunnel verwendet werden kann, um zunächst das Multicast-Paket des dynamische Routing-Protokolls in einem GRE-IPv4-Unicast-Paket zu kapseln, das dann durch IPv4sec verschlüsselt werden kann.

Dabei wird IPv4sec häufig im Transportmodus zusätzlich zu GRE bereitgestellt, da die IPv4sec-Peers und die GRE-Tunnelendpunkte (die Router) identisch sind und der Transportmodus 20 Byte an IPv4sec-Overhead spart.

Ein interessanter Fall tritt auf, wenn ein IPv4-Paket in zwei Fragmente aufgeteilt und durch GRE gekapselt wurde.

In diesem Fall erkennt IPv4sec zwei unabhängige GRE + IPv4-Pakete. In einer Standardkonfiguration ist eines dieser Pakete oft so groß, dass es nach der Verschlüsselung fragmentiert werden muss.

Der IPv4sec-Peer muss dieses Paket vor der Entschlüsselung neu assemblieren. Diese â€ždoppelte Fragmentierungâ€œ (einmal vor GRE und einmal nach IPv4sec) auf dem sendenden Router erhöht die Latenz und senkt den Durchsatz.

Die Reassemblierung erfolgt prozessgesteuert, sodass auf dem empfangenden Router bei jedem Auftreten eine CPU-Belastung auftritt.

Diese Situation kann vermieden werden, indem â€žip mtuâ€œ auf der GRE-Tunnelschnittstelle so niedrig eingestellt wird, dass es den Overhead von GRE und IPv4sec berücksichtigt (standardmäßig ist â€žip mtuâ€œ auf der GRE-Tunnelschnittstelle auf die tatsächliche Overhead-Byte-Zahl der Ausgangsschnittstelle MTU â€“ GRE eingestellt).

In dieser Tabelle sind die empfohlenen MTU-Werte für jede Tunnel-/Modus-Kombination aufgeführt, bei der davon ausgegangen wird, dass die physische Ausgangsschnittstelle eine MTU von 1500 hat.

| **Tunnel-Kombination** | **Erforderliche spezifische MTU** | **Empfohlene MTU** |
|---|---|---|
| GRE + IPv4sec (Transportmodus) | 1440 Byte | 1400 Byte |
| GRE + IPv4sec (Tunnelmodus) | 1420 Byte | 1400 Byte |

---

> **Hinweis**: Der MTU-Wert von 1400 wird empfohlen, da er die gängigsten GRE + IPv4sec-Moduskombinationen abdeckt. Auch gibt es keinen erkennbaren Nachteil, wenn ein zusätzlicher Overhead von 20 oder 40 Byte zugelassen wird. Es ist einfacher, einen Wert einzustellen, der fast alle Szenarien abdeckt, und sich diesen zu merken.

---

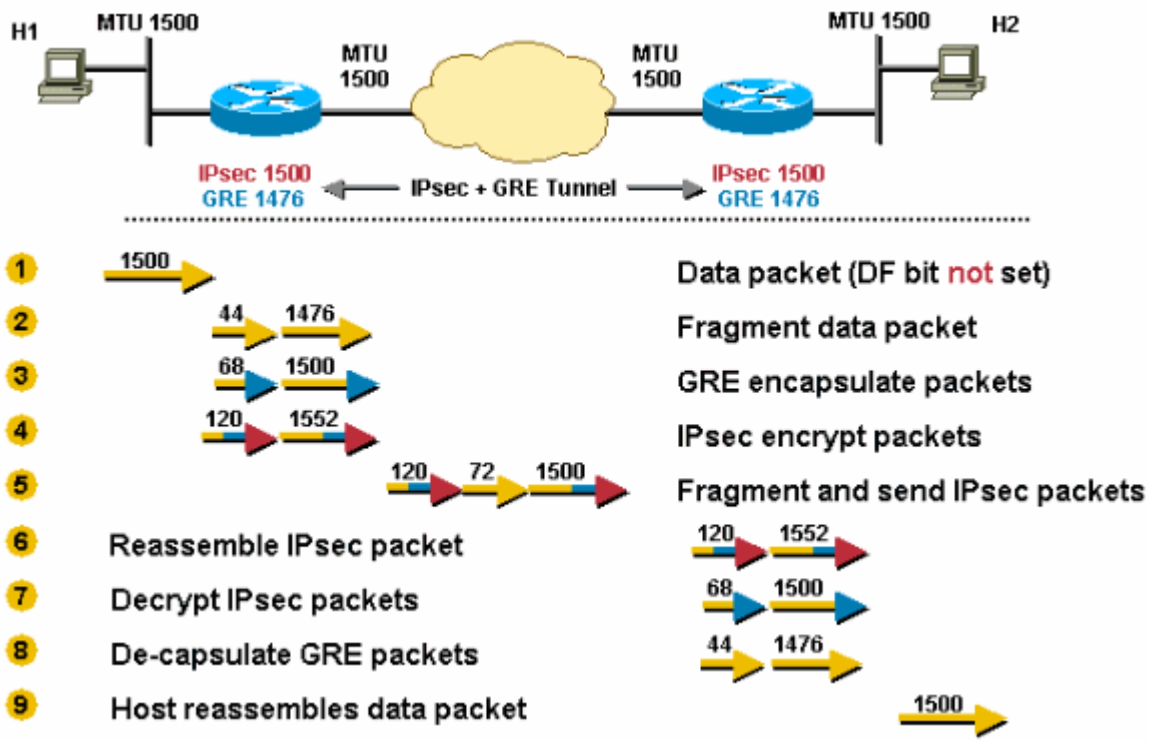
**Beispiel 9**

IPv4sec wird aufsetzend auf GRE bereitgestellt. Die MTU der physischen Ausgangsschnittstelle beträgt 1500, die IPv4sec-PMTU beträgt 1500 und die GRE-IPv4-MTU beträgt 1476 (1500 - 24 = 1476).

TCP/IPv4-Pakete werden daher zweimal fragmentiert, einmal vor GRE und einmal nach IPv4sec.

Das Paket wird vor der GRE-Kapselung fragmentiert, und eines dieser GRE-Pakete wird nach der IPv4sec-Verschlüsselung erneut fragmentiert.

Die Konfiguration von â€žip mtu 1440â€œ (IPv4sec-Transportmodus) oder â€žip mtu 1420â€œ (IPv4sec-Tunnelmodus) im GRE-Tunnel würde in diesem Szenario die Möglichkeit einer doppelten Fragmentierung ausschließen.

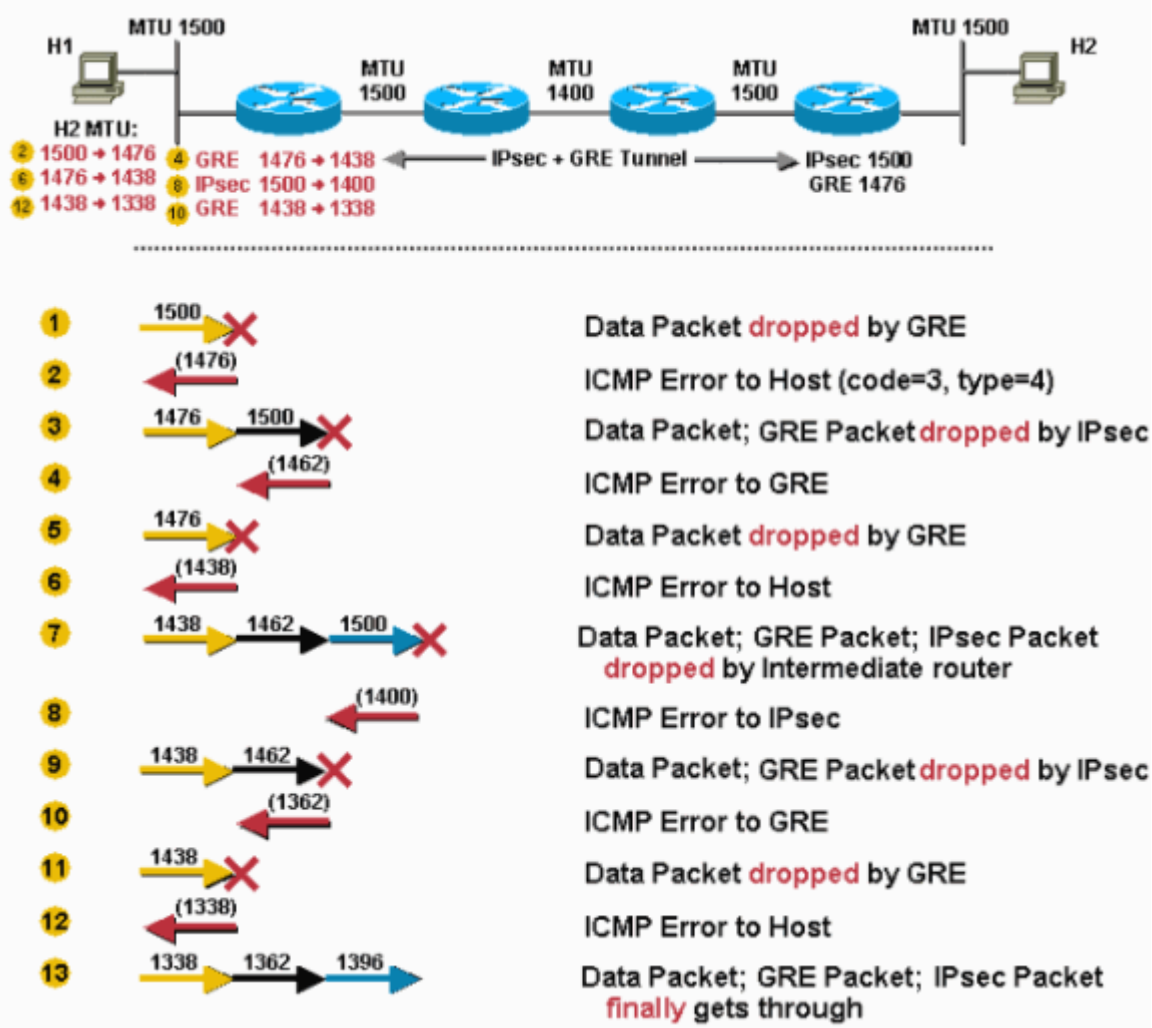



1. Der Router empfängt ein 1500 Byte großes Datagramm.
2. Vor der Kapselung fragmentiert GRE das 1500-Byte-Paket in zwei Teile, eines mit 1476 Byte (1500 - 24 = 1476) und eines mit 44 Byte (24 Byte Daten + 20 Byte IPv4-Header).
3. GRE kapselt die IPv4-Fragmente, wodurch 24 Byte zu jedem Paket hinzugefügt werden. Daraus ergeben sich zwei GRE + IPv4sec-Pakete mit 1500 (1476 + 24 = 1500) und 68 (44 + 24) Byte.
4. IPv4sec verschlüsselt die beiden Pakete, die jeweils 52 Byte (IPv4sec-Tunnelmodus) Kapselungs-Overhead hinzufügen, um ein 1552-Byte- und ein 120-Byte-Paket zu erhalten.
5. Das 1552 Byte große IPv4sec-Paket wird vom Router fragmentiert, da es größer ist als die MTU der Ausgangsschnittstelle (1500). Das 1552-Byte-Paket wird aufgeteilt in ein 1500-Byte-Paket und ein 72-Byte-Paket (52 Byte Nutzlast plus ein zusätzlicher 20-Byte-IPv4-Header für das zweite Fragment). Die drei Pakete mit 1500 Byte, 72 Byte und 120 Byte werden an den IPv4sec + GRE-Peer weitergeleitet.
6. Der empfangende Router reassembliert die beiden IPv4sec-Fragmente (1500 Byte und 72 Byte), um das ursprüngliche 1552-Byte-IPv4sec + GRE-Paket zu erzeugen. Das 120-Byte IPv4sec + GRE-Paket muss nicht verändert werden.
7. IPv4sec entschlüsselt sowohl das 1552-Byte- als auch das 120-Byte-IPv4sec + GRE-Paket, sodass zwei GRE-Pakete mit 1500 Byte und 68 Byte entstehen.
8. GRE entkapselt das 1500-Byte- und das 68-Byte-GRE-Paket, sodass zwei IPv4-Paketfragmente mit 1476 Byte und 44 Byte entstehen. Diese IPv4-Paketfragmente werden an den Ziel-Host weitergeleitet.
9. Host 2 reassembliert diese IPv4-Fragmente, sodass wieder das ursprüngliche 1500 Byte große IPv4-Datagramm erzeugt wird.

Szenario 10 ist ähnlich wie Szenario 8, außer dass es eine untere MTU-Verbindung im Tunnelpfad gibt. Dies ist ein Worst-Case-Szenario für das erste Paket, das von Host 1 an Host 2 gesendet wird. Nach dem letzten Schritt in diesem Szenario legt Host 1 die richtige PMTU für Host 2 fest, und für die TCP-Verbindungen zwischen Host 1 und Host 2 ist alles in Ordnung. TCP-Flows zwischen Host 1 und anderen Hosts (erreichbar über den IPv4sec + GRE-Tunnel) müssen nur die letzten drei Schritte von Szenario 10 durchlaufen.

In diesem Szenario wird der **tunnel path-mtu-discovery** wird im GRE-Tunnel konfiguriert, und das DF-Bit wird auf TCP/IPv4-Paketen festgelegt, die von Host 1 stammen.

**Beispiel 10**




- Der Router empfängt ein 1500-Byte-Paket. Dieses Paket wird von der GRE verworfen, weil die GRE das Paket aufgrund des festgelegten DF-Bits nicht fragmentieren oder weiterleiten kann und die Paketgröße die â€žip mtuâ€œ der Ausgangsschnittstelle überschreitet, nachdem der GRE-Overhead (24 Byte) hinzugefügt wurde.
- Der Router sendet eine ICMP-Meldung an Host 1, um ihm mitzuteilen, dass die Next-Hop-MTU 1476 (1500 - 24 = 1476) beträgt.
- Host 1 ändert seine PMTU für Host 2 auf 1476 und sendet die kleinere Größe, wenn er das Paket erneut sendet. Die GRE kapselt es und übergibt das 1500-Byte-Paket an IPv4sec. IPv4sec verwirft das Paket, weil die GRE das (festgelegte) DF-Bit aus dem inneren IPv4-Header kopiert hat, und mit dem IPv4sec-Overhead (maximal 38 Byte) ist das Paket zu groß, um über die physische Schnittstelle weitergeleitet zu werden.
- IPv4sec sendet eine ICMP-Meldung an die GRE, die angibt, dass die Next-Hop-MTU 1462 Byte beträgt (da maximal 38 Byte für die Verschlüsselung und den IPv4-Overhead hinzugefügt werden). Die GRE zeichnet den Wert 1438 (1462 - 24) als â€žip mtuâ€œ an der Tunnelschnittstelle auf.

---

- **Hinweis**: Diese Wertänderung wird intern gespeichert und ist in der Ausgabe des **show ip interface tunnel**<#>aus. Sie sehen diese Änderung nur, wenn Sie den **debug tunnel** aus.

---

- Wenn Host 1 das 1476-Byte-Paket das nächste Mal sendet, wird es von der GRE verworfen.
- Der Router sendet eine ICMP-Meldung an Host 1, die anzeigt, dass die Next-Hop-MTU 1438 beträgt.
- Host 1 senkt die PMTU für Host 2 und sendet ein 1438 Byte großes Paket erneut. Diesmal akzeptiert die GRE das Paket, kapselt es und leitet es zur Verschlüsselung an IPv4sec weiter.

- Das IPv4sec-Paket wird an den Zwischenrouter weitergeleitet und verworfen, da die MTU der Ausgangsschnittstelle 1400 beträgt.
- Der Zwischenrouter sendet eine ICMP-Meldung an IPv4sec, dass die Next-Hop-MTU 1400 beträgt. Dieser Wert wird von IPv4sec im PMTU-Wert der zugehörigen IPv4sec-SA aufgezeichnet.
- Wenn Host 1 das 1438-Byte-Paket erneut sendet, kapselt die GRE es und übergibt es an IPv4sec. IPv4sec verwirft das Paket, weil es seine eigene PMTU auf 1400 geändert hat.
- IPv4sec sendet einen ICMP-Fehler an die GRE, der anzeigt, dass die Next-Hop-MTU 1362 beträgt, und die GRE zeichnet intern den Wert 1338 auf.
- Wenn Host 1 das ursprüngliche Paket erneut sendet (weil eher keine Bestätigung erhalten hat), wird es von der GRE verworfen.
- Der Router sendet eine ICMP-Meldung an Host 1, die anzeigt, dass die Next-Hop-MTU 1338 (1362 - 24 Byte) beträgt. Host 1 senkt seine PMTU für Host 2 auf 1338.
- Host 1 sendet ein 1338-Byte-Paket erneut und kann diesmal endlich bis zum Host 2 durchdringen.

# Weitere Empfehlungen

Konfigurieren des **tunnel path-mtu-discovery** auf einer Tunnelschnittstelle kann GRE- und IPv4sec-Interaktionen unterstützen, wenn sie auf demselben Router konfiguriert sind.

Ohne die **tunnel path-mtu-discovery** -Befehls konfiguriert wurde, würde das DF-Bit immer im GRE-IPv4-Header gelöscht.

Dies ermöglicht die Fragmentierung des GRE-IPv4-Pakets, obwohl im gekapselten Daten-IPv4-Header das DF-Bit festgelegt war, was normalerweise eine Fragmentierung des Pakets nicht zulassen würde.

Wenn die **tunnel path-mtu-discovery** auf der GRE-Tunnelschnittstelle konfiguriert:

1. Die GRE kopiert das DF-Bit vom Daten-IPv4-Header in den GRE-IPv4-Header.
2. Wenn das DF-Bit im GRE-IPv4-Header festgelegt ist und das Paket nach der IPv4sec-Verschlüsselung für die IPv4-MTU an der physischen Ausgangsschnittstelle "zu groß" ist, verwirft IPv4sec das Paket und benachrichtigt den GRE-Tunnel, seine IPv4-MTU-Größe zu reduzieren.
3. IPv4sec führt die PMTUD für seine eigenen Pakete aus, und wenn sich die IPv4sec-PMTU ändert (wenn sie reduziert wird), benachrichtigt IPv4sec die GRE nicht sofort, aber wenn ein anderes größeres Paket durchkommt, dann erfolgt der Prozess in Schritt 2.
4. Die GRE-IPv4-MTU ist jetzt kleiner, sodass alle Daten-IPv4-Pakete mit festgelegtem DF-Bit verworfen werden, die jetzt zu groß sind, und eine ICMP-Nachricht an den sendenden Host gesendet wird.

Die Fehlermeldung **tunnel path-mtu-discovery** hilft der GRE-Schnittstelle, ihre IPv4-MTU nicht statisch, sondern dynamisch mit der **ip mtu** aus. Es wird empfohlen, dass beide Befehle verwendet werden.

Die Fehlermeldung **ip mtu** wird verwendet, um Platz für den GRE- und IPv4sec-Overhead in Bezug auf die IPv4-MTU der lokalen physischen Ausgangsschnittstelle zu schaffen.

Die Fehlermeldung **tunnel path-mtu-discovery** -Befehl ermöglicht die weitere Reduzierung der IPv4-MTU des GRE-Tunnels, wenn es im Pfad zwischen den IPv4sec-Peers eine Verbindung mit einer niedrigeren IPv4-MTU gibt.

Folgendes können Sie tun, wenn Sie Probleme mit der PMTUD in einem Netzwerk haben, in dem GRE + IPv4sec-Tunnel konfiguriert sind.

Diese Liste beginnt mit der wünschenswertesten Lösung.

1. Beheben Sie das Problem, dass PMTUD nicht funktioniert, was normalerweise durch einen Router

oder eine Firewall verursacht wird, der bzw. die ICMP blockiert.
2. Verwenden Sie **ip tcp adjust-mss** an den Tunnelschnittstellen, sodass der Router den TCP-MSS-Wert im TCP-SYN-Paket reduziert. Dadurch können die beiden End-Hosts (der TCP-Sender und -Empfänger) Pakete klein genug verwenden, um eine PMTUD zu vermeiden.
3. Verwenden Sie richtlinienbasiertes Routing auf der Eingangsschnittstelle des Routers und konfigurieren Sie eine Routenübersicht, um das DF-Bit im IPv4-Header der Daten zu löschen, bevor das Paket zur GRE-Tunnelschnittstelle gelangt. Dadurch kann das Daten-IPv4-Paket vor der GRE-Kapselung fragmentiert werden.
4. Erhöhen Sie die â€žip mtuâ€œ auf der GRE-Tunnelschnittstelle auf den MTU-Wert der Ausgangsschnittstelle. Auf diese Weise kann das Daten-IPv4-Paket GRE-gekapselt werden, ohne es zuerst zu fragmentieren. Das GRE-Paket wird dann mit IPv4sec verschlüsselt und dann fragmentiert, um an die physische Ausgangsschnittstelle zu gelangen. In diesem Fall würden Sie **tunnel path-mtu-discovery** auf der GRE-Tunnelschnittstelle. Dies kann den Durchsatz drastisch reduzieren, da die Reassemblierung von IPv4-Paketen auf dem IPv4sec-Peer im Prozess-Switching-Modus erfolgt.

# Zugehörige Informationen

- **IP Routing-Support-Seite**
- **IPSec (IP Security Protocol)-Support-Seite**
- **RFC 1191: MTU-Pfaderkennung**
- **RFC 1063 Optionen für die IP-MTU-Erkennung**
- **RFC 791 Internetprotokoll**
- **RFC 793 Transmission Control Protocol**
- **RFC 879 Maximale TCP-Segmentgröße und verwandte Themen**
- **RFC 1701 Generic Routing Encapsulation (GRE)**
- **RFC 1241 Ein Schema für ein Internet-Kapselungsprotokoll**
- **RFC 2003 IP-Kapselung innerhalb von IP**
- **Technischer Support und Dokumentation für Cisco Systeme**

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.