

能ping通，TCP就一定能連通嗎？

程序員零距離 2022-09-30 13:06 發表於江蘇

以下文章來源於小白debug，作者小白



小白debug

答應我，關注之後，好好學技術，別只是收藏我的表情包。。

(給程序員零距離加星標，了解項目開發.)

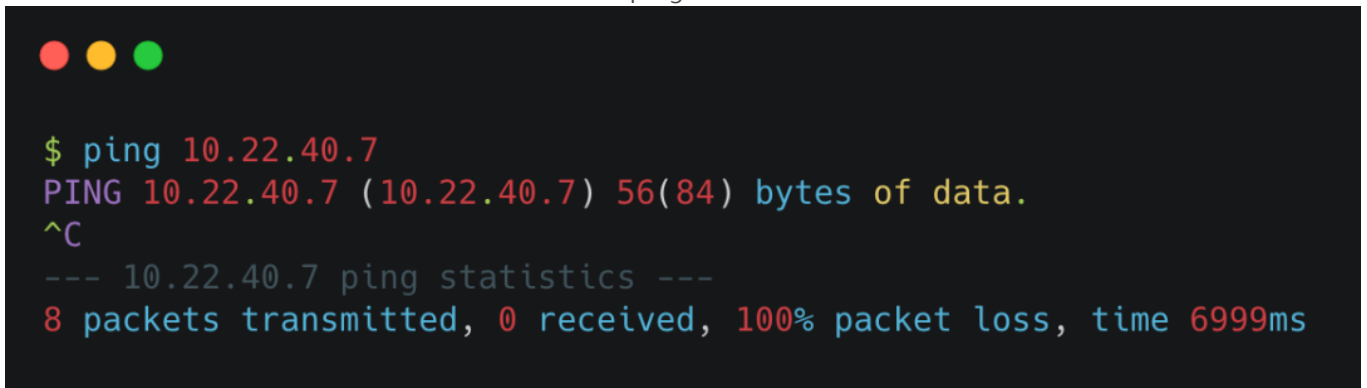


平時，我們要知道，自己的機器到目的機器之間，**網絡通不通**，一般會執行**ping命令**。

一般對於狀況良好的網絡來說，你能看到它對應的 **loss** 丟包率為 **0%**，也就是所謂的**能ping通**。如果看到丟包率 **100%**，也就是**ping不通**。



ping正常

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal shows a command prompt '\$' followed by the command 'ping 10.22.40.7'. The output shows 'PING 10.22.40.7 (10.22.40.7) 56(84) bytes of data.' followed by a carriage return '^C'. Then it shows '--- 10.22.40.7 ping statistics ---' and finally '8 packets transmitted, 0 received, 100% packet loss, time 6999ms'.

```
$ ping 10.22.40.7
PING 10.22.40.7 (10.22.40.7) 56(84) bytes of data.
^C
--- 10.22.40.7 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 6999ms
```

ping不通

那麼問題來了，假設我能ping通某台機器，那這時候如果我改用TCP協議去發數據到目的機器，也一定能通嗎？

或者換個問法，ping和tcp協議走的網絡路徑是一樣的嗎？

這時候第一反應就是不一定，因為ping完之後中間鏈路里的某個路由器可能會掛了（斷電了），再用TCP去連就會走別的路徑。

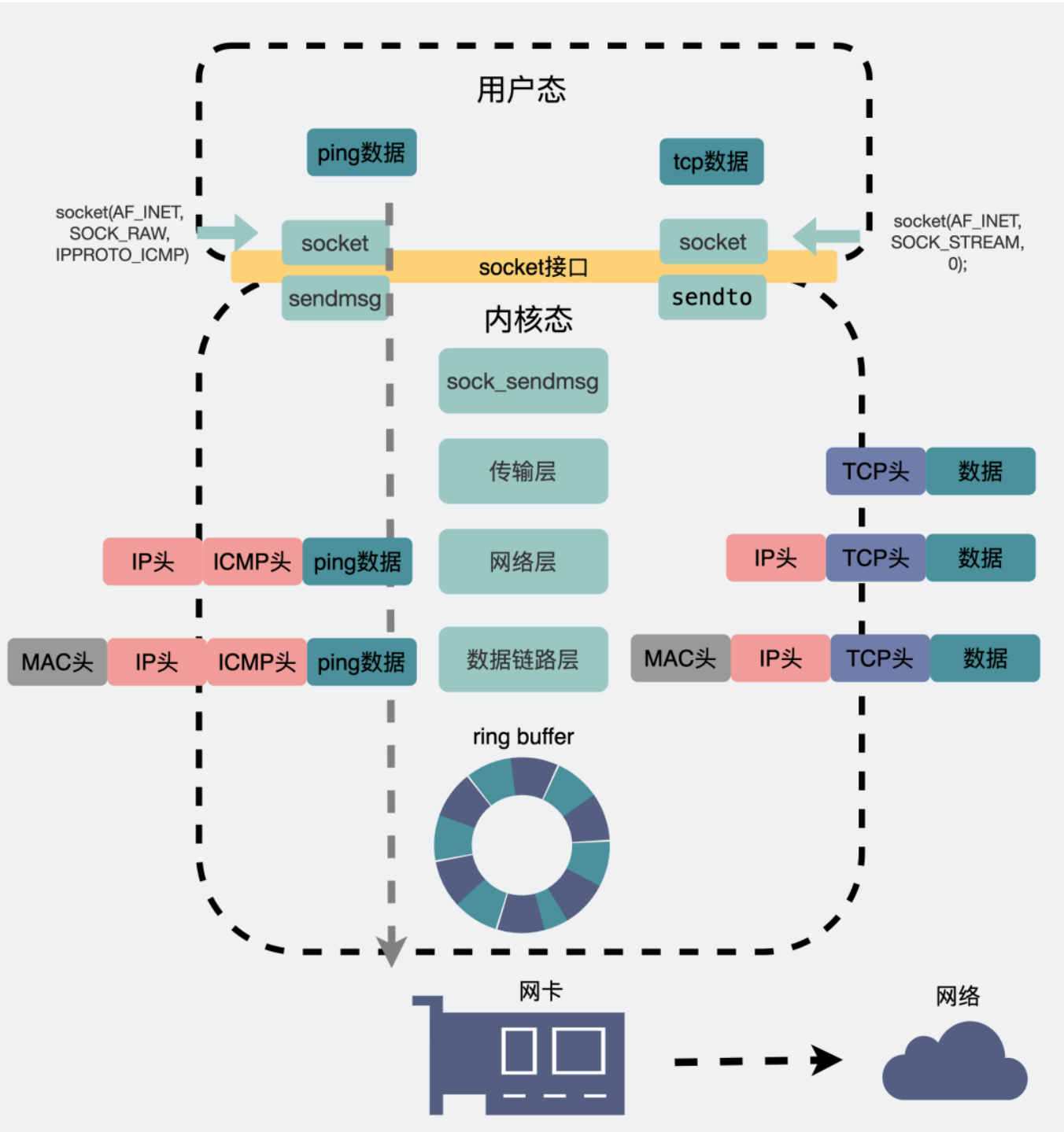
也沒錯。但假設，中間鏈路沒發生任何變化呢？

我先直接說答案。

不一定，走的網絡路徑還是有可能是不同的。

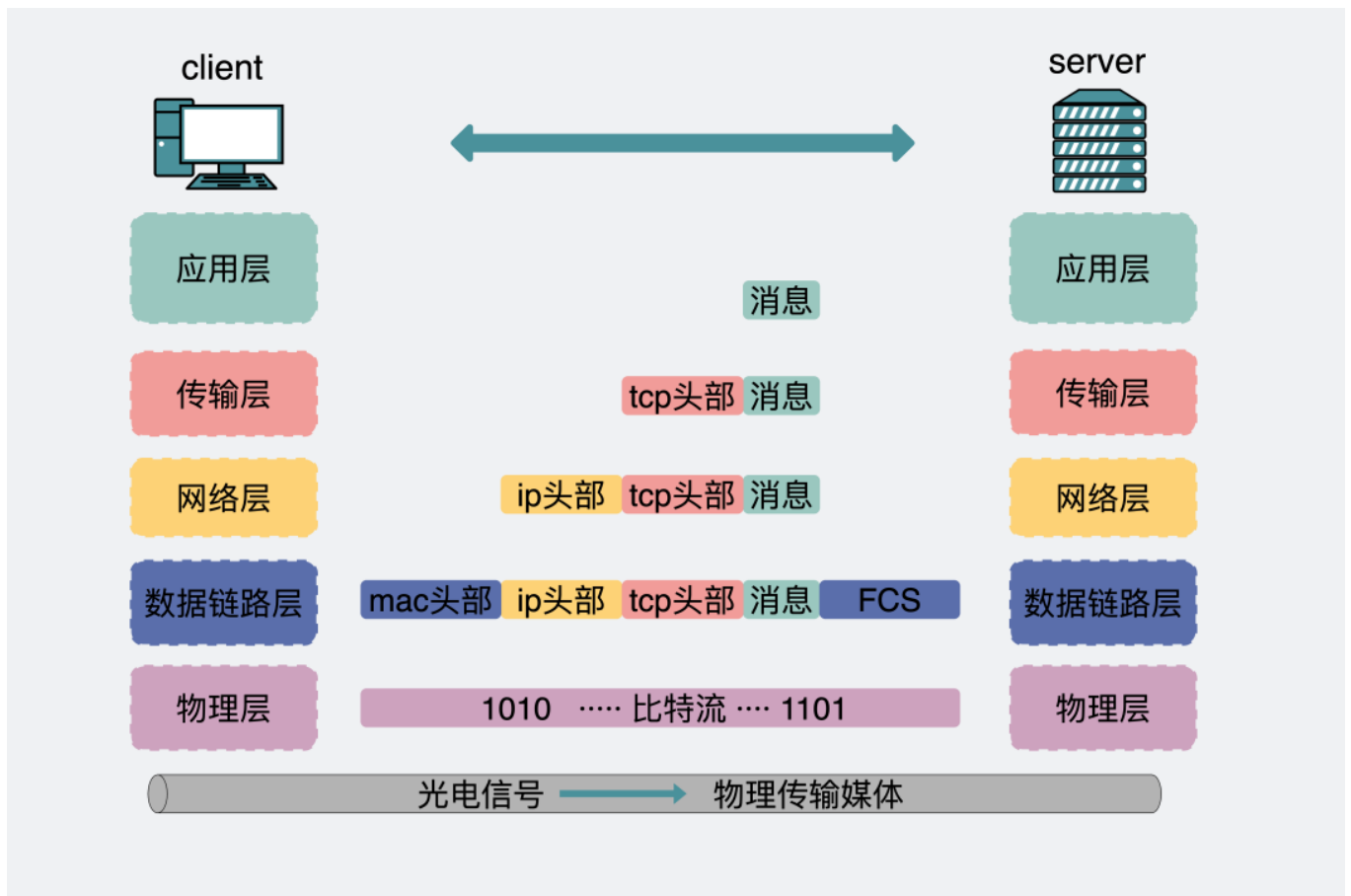
今天就來聊聊為什麼。

我之前寫過一篇《斷網了，還能ping通127.0.0.1 嗎？》,裡面提到過ping數據包和tcp數據包的區別。



ping和TCP發消息的區別

我們知道網絡是分層的，每一層都有對應協議。



五層網絡協議對應的消息體變化分析

而這網絡層就像搭積木一樣，上層協議都是基於下層協議搭出來的。

不管是ping（用了ICMP協議）還是tcp本質上都是基於網絡層IP協議的數據包，而到了物理層，都是二進制01串，都走網卡發出去了。

如果網絡環境沒發生變化，目的地又一樣，那按道理說他們走的網絡路徑應該是一樣的，什麼情況下會不同呢？

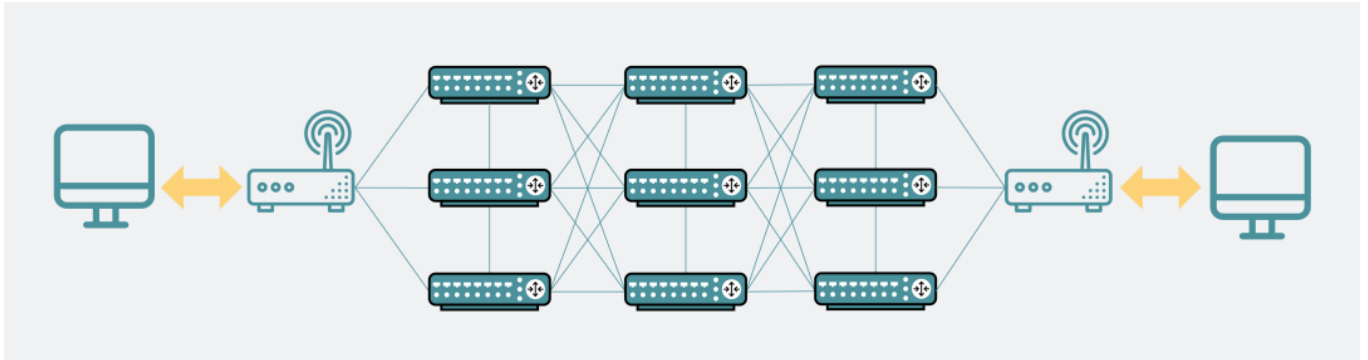
我們就從路由這個話題聊起吧。

網絡路徑

在我們的想像中，當我們想在兩台機器之間傳輸數據。本機和目的機器之間會建立一條連接，像一條管道一樣，數據從這頭到那頭。這條管道其實是我們為了方便理解而抽象出來的概念。

實際上，我們將數據包從本地網卡發出之後，會經過各種**路由器（或者交換機）**，才能到達目的機器。

這些路由器數量眾多，相互之間可以互連，連起來之後就像是一張大網，所以叫**"網絡"**可以說是非常的形象。



路由器構成的網絡

考慮到交換機有的功能，路由器基本上都支持，所以我們這邊只討論路由器。

那麼現在問題來了，**路由器收到數據後，怎麼知道應該走哪條路徑，傳給哪個路由器？**

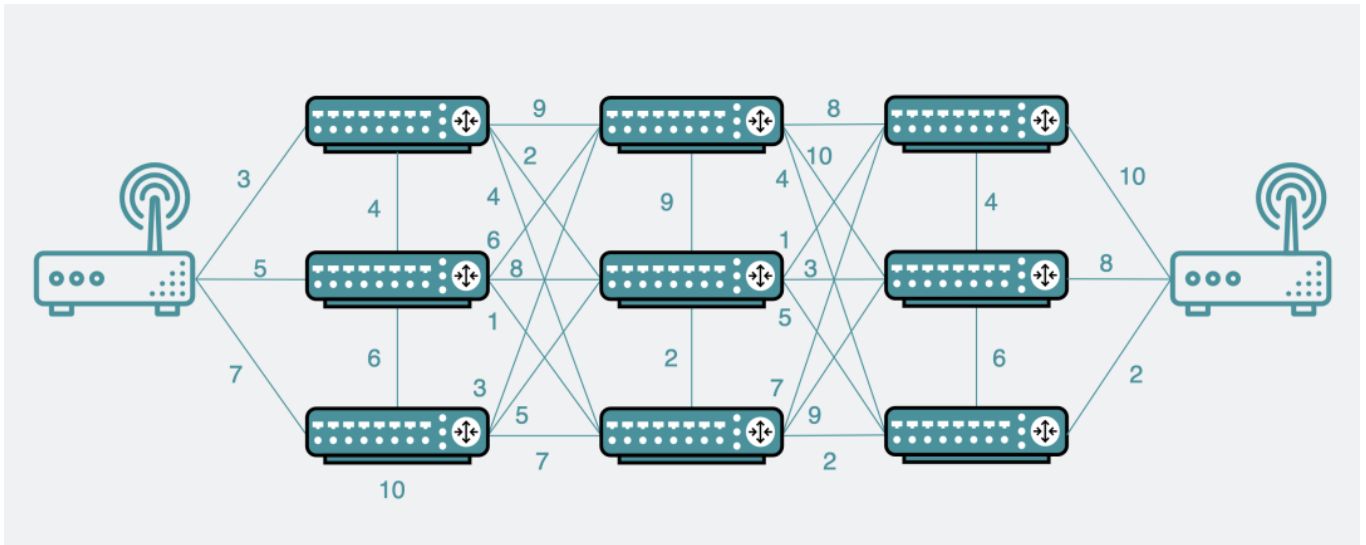
路徑由什麼決定？

在上面的那麼大一張網絡中，隨便一個路由器都有可能走任何一個路徑，將數據發到另外一個路由器上，

但路由和路由之間距離，帶寬啥的可能都不同。

於是就很需要知道，**兩點之間走哪條路才是最優路徑。**

於是問題就變成了這樣一個**圖狀結構**。每條邊都帶有**成本或權重**，算這上面**任意兩點的最短距離**。



路由器和Dijkstra

這時候想必大家回憶壓不住要上來了。

這題我熟，這就是大學時候刷的[Dijkstra算法](#)。菊花廠的OJ筆試題集裡也經常出現，現在終於明白為什麼他們家的筆試題裡圖類題目比別的大廠貌似要多一些了吧，因為菊花廠就是搞通信的，做路由器的老玩家了。

路由表的生成

基於 **Dijkstra** 算法，封裝出了一個新的協議，**OSPF** 協議（**O** pen **S** hortest **P** ath **F** irst, 開放最短路徑優先）。

有了OSPF，路由器就得到了網絡圖裡自己到其他點之間的**最短距離**，於是就知道了**數據包要到某個點，該走哪條最優路徑**。

將這些信息匯成一張表，也就是我們常說的**路由表**。

路由表裡記錄了到什麼IP需要走什麼端口，以及走這條路徑的成本（**metric**）。

可以通過 **route** 命令查看到。

```
$ route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          gateway          0.0.0.0          UG    0      0      0 eth0
link-local        0.0.0.0          255.255.0.0      U     1002   0      0 eth0
172.52.14.0       0.0.0.0          255.255.240.0    U      0      0      0 eth0
```

route表

路由表決定數據包路徑

數據包在發送的過程中，會在網絡層加入目標地址IP。

路由器會根據這個IP跟路由表去做匹配。

然後路由表，會告訴路由器，什麼樣的消息該轉發到什麼端口。

舉個例子。



通過路由表轉發數據

假設A要發消息到D。也就是 **192.168.0.105/24** 要發消息到 **192.168.1.11/24**。

那麼A會把消息經發到路由器。

路由器已知目的地IP **192.168.1.11/24**，去跟**路由表**做匹配，發現 **192.168.1.0/24**，就在e2端口，那麼就會把消息從e2端口發出，（可能還會經過交換機）最後把消息打到目的機器。

當然，如果路由表裡找不到，那就打到**默認網關**吧，也就是從e1口發出，發到IP **192.0.2.1**。**這個路由器的路由表不知道該去哪，說不定其他路由器知道。**

路由表的匹配規則

上面的例子裡，是只匹配上了路由表裡的一項，所以只能是它了。

但是，條條大路通羅馬。實際上能到目的地的路徑肯定有很多。

如果路由表裡有很多項都被匹配上了，會怎麼選？

如果多個路由項都能到目的地，那就優先選**匹配長度更長**的那個。比如，還是目的地 **192.168.1.11**，發現路由表裡的**192.168.1.0/24**和**192.168.0.0/16**都能匹配上，但明顯**前者匹配長度更長**，所以最後會走**192.168.1.0/24**對應的轉發端口。

但如果兩個表項的匹配長度都一樣呢？

那就會看生成這個路由表項的**協議**是啥，選優先級高的，優先級越高也就是所謂的**管理距離**（**AD**，**A**dministrative **D**istance）越小。比如說優先選**手動配**的靜態（**static**）路由，次優選**OSPF**動態學習過來的表項。

如果還是相同，就看**度量值metrics**，其實也就是**路徑成本cost**，成本越小，越容易被選中。

路由器能選的路線有很多，但按道理，最優的只有“一條”，所以到這裡為止，我們都可以認為，對於同一個目的地，**ping**和**TCP**走的路徑是相同的。

但是。

如果連路徑成本都一樣呢？也就是說有多條最優路徑呢。

那就都用。

這也就是所謂的等價多路徑，ECMP (E qual C ost M ulti P ath)。

我們可以通過 `traceroute` 看下鏈路是否存在等價多路徑的情況。

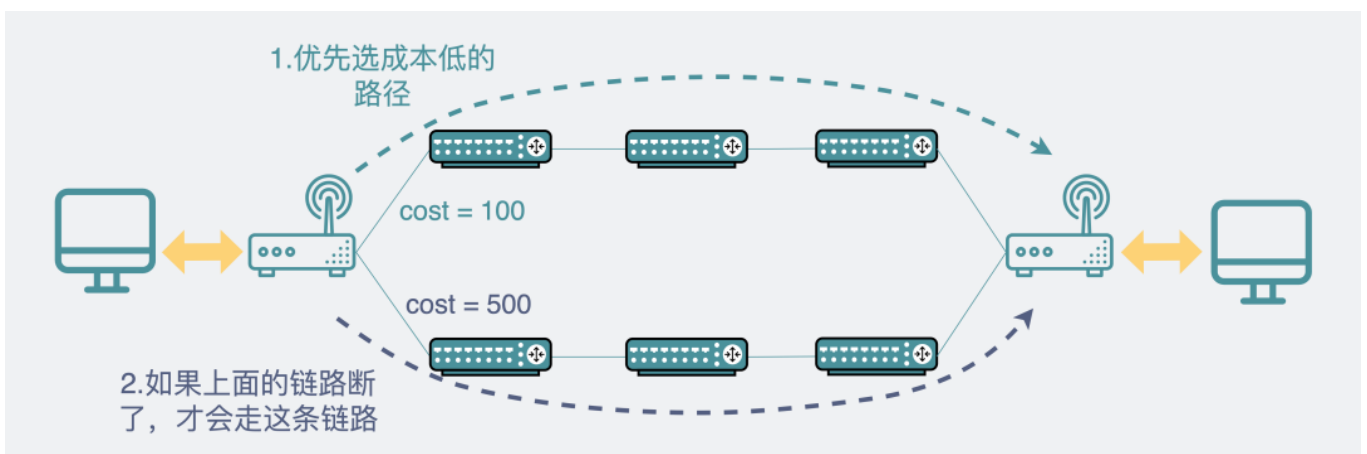
```
$ traceroute baidu.com
traceroute to baidu.com (39.156.66.10), 30 hops max, 60 byte packets
 1  10.90.180.26 (10.90.180.26)  1.584 ms  10.90.181.26 (10.90.181.26)  1.946 ms  10.90.180.26 (10.90.180.26)  1.570 ms
 2  11.73.1.101 (11.73.1.101)  5.363 ms  11.73.1.137 (11.73.1.137)  5.916 ms  5.785 ms
 3  11.95.58.57 (11.95.58.57)  2.159 ms  11.228.249.50 (11.228.249.50)  2.185 ms  11.228.249.18 (11.228.249.18)  8.647 ms
 4  103.52.72.134 (103.52.72.134)  3.083 ms  10.102.236.94 (10.102.236.94)  3.438 ms  157.119.195.94 (157.119.195.94)  3.445 ms
 5  42.120.241.41 (42.120.241.41)  3.686 ms  10.54.37.234 (10.54.37.234)  3.648 ms  10.54.37.198 (10.54.37.198)  2.088 ms
 6  . (117.131.11.89)  4.194 ms  3.914 ms . (117.131.11.85)  3.555 ms
 7  . (120.204.35.233)  3.814 ms . (120.204.36.5)  4.958 ms . (120.204.35.241)  4.517 ms
 8  221.183.53.225 (221.183.53.225)  5.143 ms * 221.183.53.229 (221.183.53.229)  11.066 ms
 9  221.183.37.133 (221.183.37.133)  29.109 ms  221.183.37.217 (221.183.37.217)  26.858 ms *
10  * 221.183.53.178 (221.183.53.178)  29.149 ms *
11  39.156.27.1 (39.156.27.1)  28.735 ms  29.367 ms  39.156.27.5 (39.156.27.5)  29.780 ms
12  39.156.27.5 (39.156.27.5)  29.289 ms  29.558 ms *
13  * * *
```

可以看到，中間某幾行，有好幾個IP，也就是說這一跳裡同時可以選好幾個目的機器，說明這段路徑支持ECMP。

ECMP有什麼用

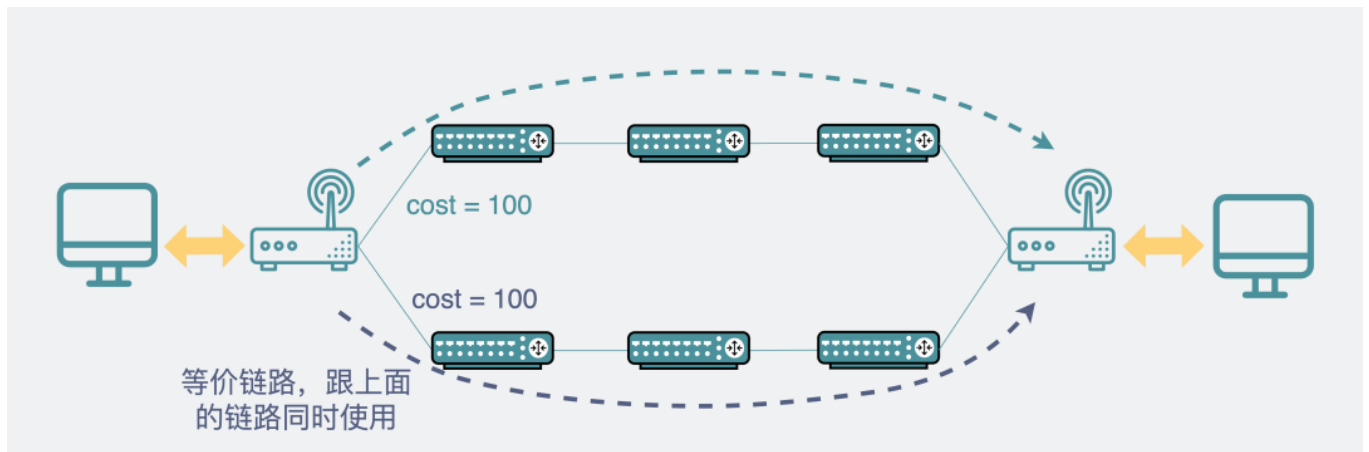
利用等價多路徑，我們可以增加鏈路帶寬。

舉個例子。



從A點到B點，如果這兩條路徑成本不同，帶寬都是 **1千兆**。那數據包肯定就選成本低的那條路了，如果這條路出故障了，就走下面那條路。但不管怎麼樣，**同一時間，只用到了了一條路徑**。另外一條閒置就有些浪費了，有沒有辦法可以利用起來呢？

有，將它們兩條路徑的成本設置成一樣，那它們就成了等價路由，然後中間的路由器開啟**ECMP**特性，就可以同時利用這兩條鏈路了。帶寬就從原來的 **1千兆** 變成了 **2千兆**。數據就可以在兩條路徑中隨意選擇了。



利用ECMP可以同時使用兩條鏈路

但這也帶來了另外一個問題。**加劇了數據包亂序**。

原來我只使用一條網絡路徑，數據依次發出，如無意外，也是依次到達。

現在兩個數據包走兩條路徑，先發的數據包可能後到。這就亂序了。

那麼問題又又來了。

亂序會有什麼問題？

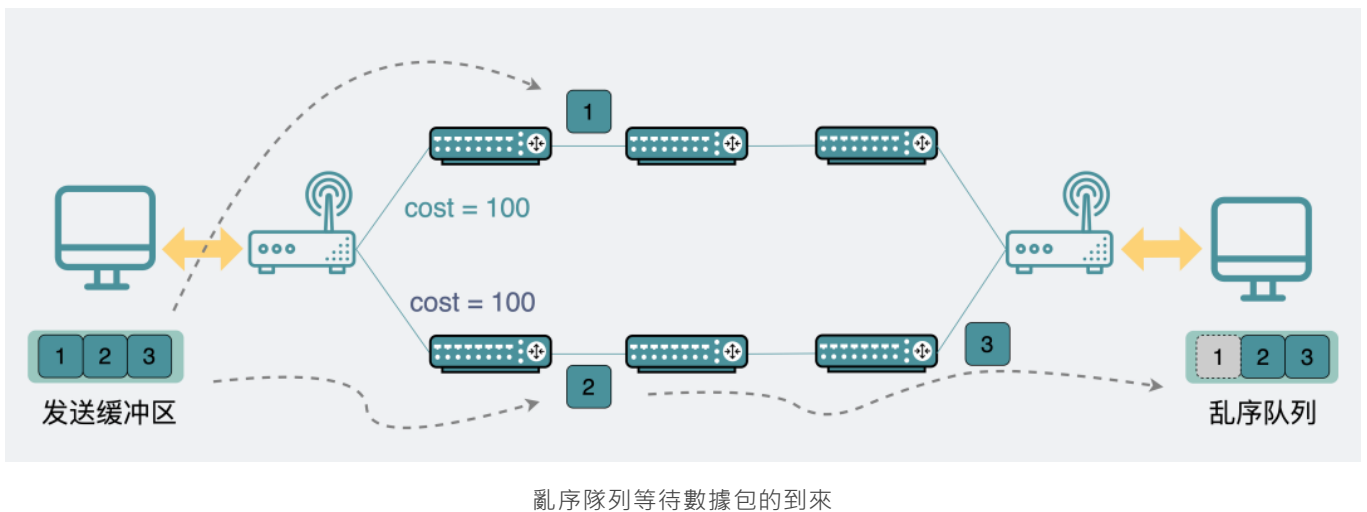
對於我們最最最常使用的TCP協議來說，它是個可靠性網絡的協議，這裡提到的**可靠**，不僅是保證數據要能送到目的地，還要保證**數據順序**要跟原來發送端的一樣。

實現也很簡單，**TCP為每個數據包（segment）做上編號**。數據到了接收端后，根據**數據包編號**發現是**亂序數據包**，就會扔到**亂序隊列**中對數據包進行排序。如果前面的數據

包還沒到，哪怕後面的數據包先到了，也得在亂序隊列中一直等，到齊後才能被上層拿到。

舉個例子，發送端發出三個數據包，编号1, 2, 3，假設在傳輸層 2和3 先到了，1 還沒到。那此時應用層是沒辦法拿到 2和3 的數據包的，必須得等 1 來了之後，應用層才能一次性拿到這三個包。因為這三個包原來可能表示的是一個完整的消息，少了1, 那麼消息就不完整，應用層拿到了也毫無意義。

像這種，由於前面的數據丟失導致後面的數據沒辦法及時給到應用層的現象，就是我們常說的TCP隊頭阻塞。



亂序發生時 2和3 需要待在亂序隊列中，而亂序隊列其實用的也是接收緩衝區的內存，而接收緩衝區是有大小限制的。通過下面的命令可以看到接收緩衝區的大小。

```
# 查看接收缓冲区
$ sysctl net.ipv4.tcp_rmem
net.ipv4.tcp_rmem = 4096(min) 87380(default) 6291456(max)
# 缓冲区会在min和max之间动态调整
```

亂序的情況越多，接收緩衝區的內存就被佔用的越多，對應的接收窗口就會變小，那正常能收的數據就變少了，網絡吞吐就變差了，也就是性能變差了。

因此，我們需要盡量保證所有同一個TCP連接下的所有TCP包都走相同路徑，這樣才能最大程度避免丟包。

ECMP的路徑選擇策略

當初開啟ECMP就是為了提升性能，現在反而加重了亂序，降低了TCP傳輸性能。

這怎麼能忍。

為了解決這個問題，我們需要有一個合理的路徑選擇策略。為了避免同一個連接裡的數據包亂序，我們需要保證同一個連接裡的數據包，都走同樣的路徑。

這好辦。我們可以通過連接的**五元組**（發送方的**IP**和**端口**，接收方的**IP**和**端口**，以及通信**協議**）信息定位到唯一一條連接。



五元組

然後對五元組信息生成哈希鍵，讓同一個哈希鍵的數據走同一條路徑，問題就完美解決了。



五元組映射成hash鍵



根據五元組選擇ECMP路徑

TCP和Ping走的網絡路徑一樣嗎

現在我們回到文章開頭的問題。

對於同樣的發送端和接收端，TCP和Ping走的網絡路徑一樣嗎？

不一定一樣，因為五元組裡的信息裡有一項是通信協議。ping用的是ICMP協議，跟TCP協議不同，並且ping不需要用到端口，所以五元組不同，生成的哈希鍵不同，通過ECMP選擇到的路徑也可能不同。

TCP和Ping的五元组差异					
TCP和Ping 是否一致	发送方IP	发送方Port	协议	接收方IP	接收方Port
	一致	不一致	不一致 TCP vs ICMP	一致	不一致

TCP和ping的五元組差異

同樣都用TCP協議，數據包走的網絡路徑一樣嗎

還是同樣的發送端和接收端，同樣是TCP協議，不同TCP連接走的網絡路徑是一樣的嗎？

跟上面的問題一樣，其實還是五元組的問題，同樣都是TCP協議，對於同樣的發送端和接收端，他們的IP和接收端的端口肯定是一樣的，但發送方的端口是可以隨時變化的，因此通過ECMP走的路徑也可能不同。

不同TCP连接的五元组差异					
五元组 是否一致	发送方IP	发送方Port	协议	接收方IP	接收方Port
	一致	不一致	一致	一致	一致

不同TCP連接的五元組差異

但問題又來了。

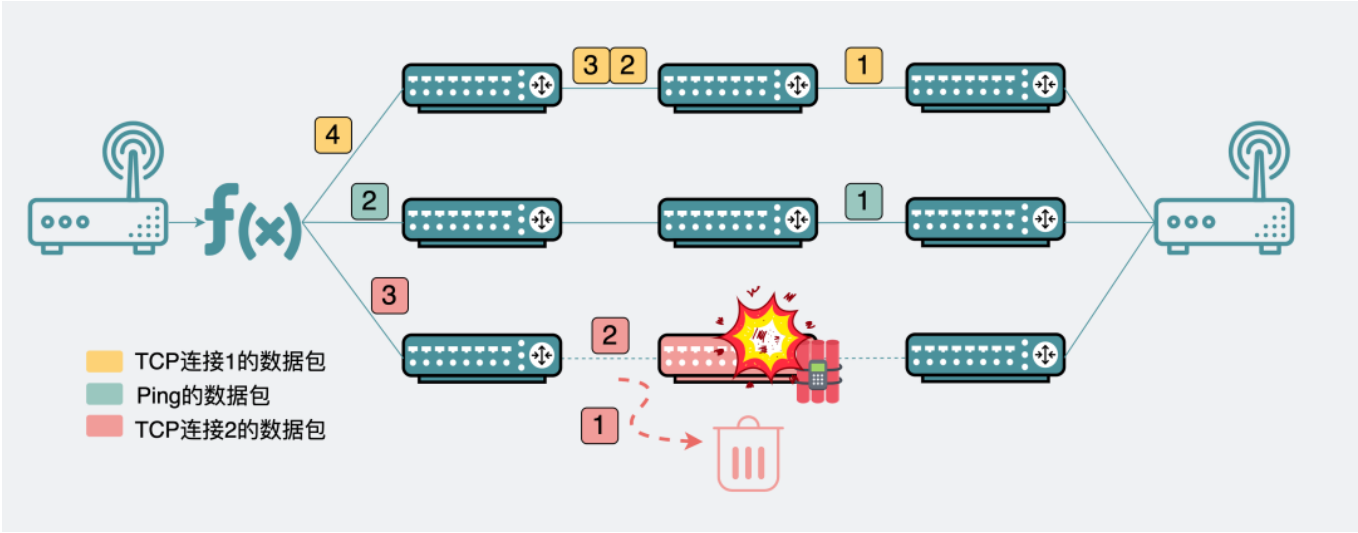
我知道這個有什麼用呢？我做業務開發，又沒有設置網絡路由的權限。

利用這個知識點排查問題

對於業務開發，這絕對不是個沒用的知識點。

如果某天，你發現，你能ping通目的機器，但用TCP去連，卻偶爾連不上目的機器。而且兩端機器都挺空閒，沒什麼性能上的瓶頸。實在走投無路了。

你就可以想想，會不會是網絡中用到了 ECMP，其中一條鏈路有問題導致的。



ping能成功但部分TCP連接失敗

排查方法也很簡單。

你是知道本機的IP以及目的機器的IP和端口號的，也知道自己用的是TCP連接。

只要你在報錯的時候打印下錯誤信息，你就知道了發送端的端口號了。

這樣五元組是啥你就知道了。

下一步就是指定發送端的端口號重新發起TCP請求，同樣的五元組，走同樣的路徑，按理說如果鏈路有問題，就肯定會復現。

如果不想改自己的代碼，你可以用nc命令指定客戶端端口看下能不能正常建立TCP連接。

```
nc -p 6666 baidu.com 80
```

-p 6666 是指定發出請求的客戶端端口是 6666，後面跟著的是連接的域名和80端口。

tcp.port == 6666									
No.	Time	Source	Destination	Protocol	Length	Calculated window size	Bytes in flight	Time since request	Info
566	16.543700	192.168.31.170	110.242.68.66	TCP	78	65535			6666 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=92
568	16.577119	110.242.68.66	192.168.31.170	TCP	78	8192			80 → 6666 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=3
569	16.577224	192.168.31.170	110.242.68.66	TCP	54	262144			6666 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
780	26.617879	110.242.68.66	192.168.31.170	TCP	54	24704			80 → 6666 [FIN, ACK] Seq=1 Ack=1 Win=24704 Len=0
781	26.618031	192.168.31.170	110.242.68.66	TCP	54	262144			6666 → 80 [ACK] Seq=1 Ack=2 Win=262144 Len=0
782	26.618100	192.168.31.170	110.242.68.66	TCP	54	262144			6666 → 80 [FIN, ACK] Seq=1 Ack=2 Win=262144 Len=0
783	26.652964	110.242.68.66	192.168.31.170	TCP	54	24704			80 → 6666 [ACK] Seq=2 Ack=2 Win=24704 Len=0

通過nc成功建立tcp連接

假設用了 6666端口 的五元組去連接總是失敗，改用 6667或其他端口 卻能成功，你可以帶著這個信息去找負責網絡的同事。

總結

- 路由器可以通過OSPF協議生成路由表，利用數據包裡的IP地址去跟路由表做匹配，選擇最優路徑後進行轉發。
- 當路由表一個都匹配不上時會走默認網關。當匹配上多個的時候，會先看**匹配長度**，如果一樣就看**管理距離**，還一樣就看**路徑成本**。如果連路徑成本都一樣，那**等價路徑**。如果路由開啟了ECMP，那就可以同時利用這幾條路徑做傳輸。
- ECMP可以提高鏈路帶寬，同時利用五元組做哈希鍵進行路徑選擇，保證了同一條連接的數據包走同一條路徑，減少了亂序的情況。
- 可以通過traceroute命令查看到鏈路上是否有用到ECMP的情況。
- 開啟了ECMP的網絡鏈路中，TCP和ping命令可能走的路徑不同，甚至同樣是TCP，不同連接之間，走的路徑也不同，因此出現了連接時好時壞的問題，實在是走投無路了，可以考慮下是不是跟ECMP有關。
- 當然，**遇到問題多懷疑自己，要相信絕大部分時候真的跟ECMP無關**。

參考資料

《網絡排查案例課》——極客時間

- END -

文章精選

- 1、三大運營商集體發出警告：這些iPhone買了也用不了！
- 2、抓https 加密數據，偷偷摸摸爽得很！
- 3、靈動島來了！這才是小米今年最好看的手機...
- 4、張總讓我去維護一團屎山，我去了.....
- 5、因創始人一句話，小眾好用的瀏覽器悲劇了
- 6、讀懂HikariCP一百行代碼，多線程就是個孫子！
- 7、這些網站太會整活，內容簡直震撼！
- 8、以羊了個羊為例，淺談小程序抓包與響應報文篡改



更多精彩等待你的發現



點分享



點點贊

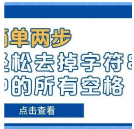


點在看

喜歡此內容的人還喜歡

簡單兩步，輕鬆去掉字符串中的所有空格

明日IT部落



1.8w 字的SQL 優化大全

Python開發者



寫一句VBA代碼，讓Excel自動發郵件~

PowerOffice

