جزوه شبکه ۲

Session 1

عناصر اصلی شبکه ← ① سیستم‌های دانشگاه کامپیوترهای متصل به هم
به اصطلاح end system یا host می‌گیم
② لینک‌های ارتباطی Communication Links
به نرخ انتقال یا transmission rate bandwidth
پهنای باند می‌گیم
③ دستگاه‌هایی که وظیفه‌ی انتقال packet ها را دارند ← switch
router

اینترنت یک شبکه‌ای از شبکه‌ها است.

protocol یک کنترل کننده فرستادن و دریافت پیام‌ها را کنترل می‌کند
مثل TCP و IP و HTTP و ...
استانداردهایی که این protocol ها را تعریف می‌کنند
Request for comments یا RFC
Internet Engineering Task force یا IETF
device های شبکه برای این protocol ها را اجرا می‌کنند
اینترنت یک سرویس برای App های مختلف فراهم کرده مثل web و email و
game و ...
social network
زیرساختی که یک API به ما می‌دهد تا به
دستگاه‌های مختلف شبکه دانش پالتیم

protocol یک فرمت مشخصی را تعریف می‌کند و ترتیب بسته‌ها و پیام‌هایی
که ارسال می‌شوند یا دریافت می‌شوند را تعریف می‌کند و کار یا action ای که
در انتقال یا دریافت پیام باید انجام شود

① تعیین فرمت پیام ② ترتیب بسته‌ها در ارسال و دریافت
③ action ای که هنگام دریافت و ارسال بسته انجام می‌شود

همه‌ی ارتباطاتی که توی اینترنت وجود دارد بر پایه‌ی protocol ها است.

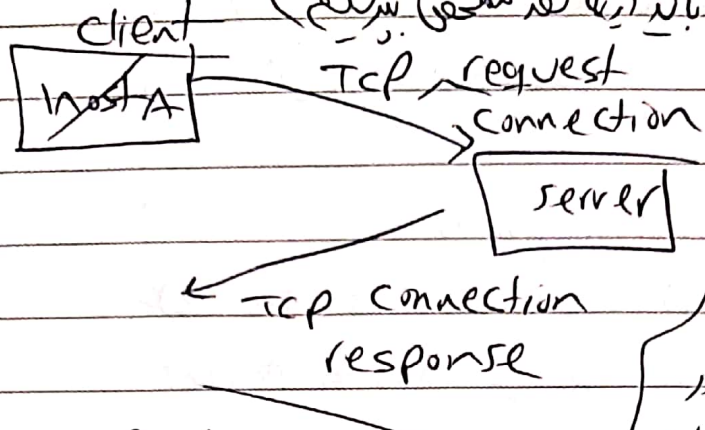
مثل ارتباط انسان‌ها
مثل سلام کردن انسان‌ها قبل از درخواست برای کاری ← مثلاً TCP ها هم قبل از

hirmandpaper

Scanned by CamScanner

درخواست Get یا Post باید به request و response دوبل کند

در واقع باید یک ارتباط بین دو تا دستگاه به مدار بسته بعد از این درخواست فایل صدا زده بشه (باید تضمین کنیم از طرف دستگاه می خاهیم اطلاعات بگیریم و تحویل بدهد ضایع نباشیم و اگه بدیم باید از طرف مقصد تحویل بدیم)



client → server

① host ← edge → server

② access network: با مدار بسته به data center ها

physical media: لینک های ارتباطی سیمی و بی سیم

③ core: router هایی که به هم وصل شده اند

شبکه ای از شبکه ها

switch ها و router ها

end system

برای وصل شدن سیستم های انتهایی به router ها ← مهم است که این دستگاه ساکن است یا در حال حرکت است و فاصله اش چقدر است و آیا ارتباط سیمی است یا بی سیم و آیا connection مشترک است بین چند دستگاه و آیا shared است یا dedicated است و این لینک چه مقدار عرض باند را (فضا) در اختیار دستگاه می گذارد و bandwidth

bits per second

با access network ها DSL

digital subscriber line ← ارتباطی که روی خطوط تلفن به مدار بسته می شه چون خطوط تلفن و چون از قبل ما شبکه های تلفنی را ساخته بودیم (سیم های قبلی بودند) و شبکه موجودی بوده ← می شه ارتباط اینترنت را روی این ساخت ← نیاز به شبکه جدید و متفاوتی نداشتیم ← برای همین از ADSL

hirmandpaper

Scanned by CamScanner

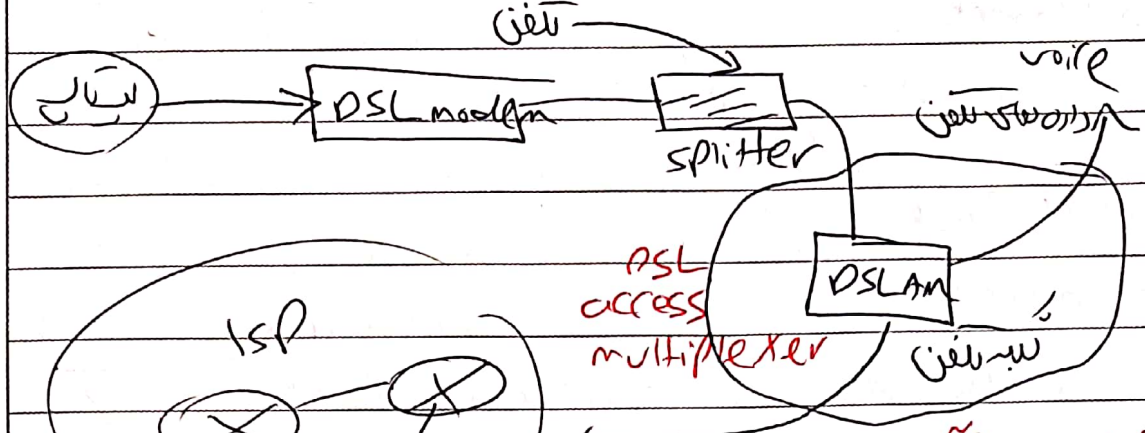
استفاده می‌کنه.

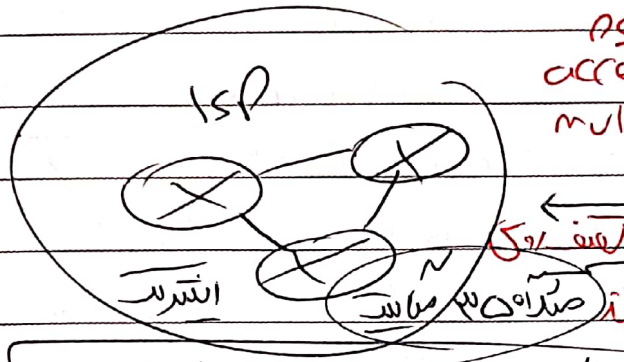
Internet ← میخواد data بفرستیم برای ← شبکه تلفن

voice ← ← ← ← ← DSL خط تلفن

rate ← آپلود ← کمتر از 2.5 Mbps (کمتر از 1Mbps معمولاً)

دانلود ← ← 24 ← ← 70 ← ←)



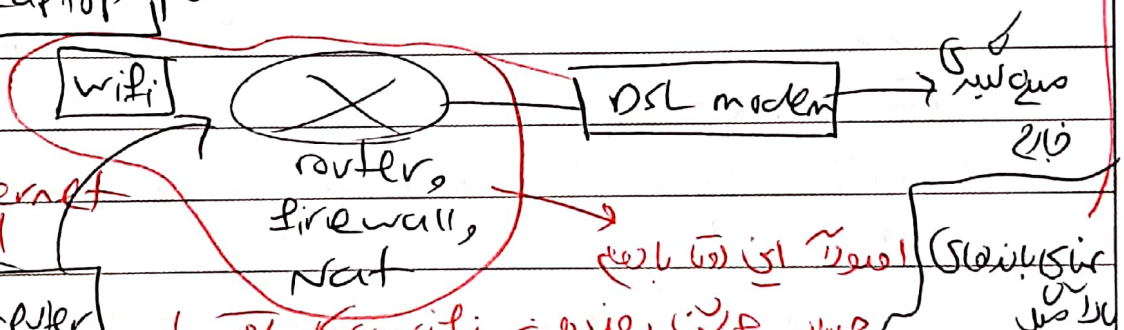


ظرفیت و پهنای باند صدا 70 است ولی تلفن روی حجم داده‌هایی که روی بدل می‌شه می‌گذارند

wireless access point از access network

استفاده می‌کنه wifi

Laptop wireless



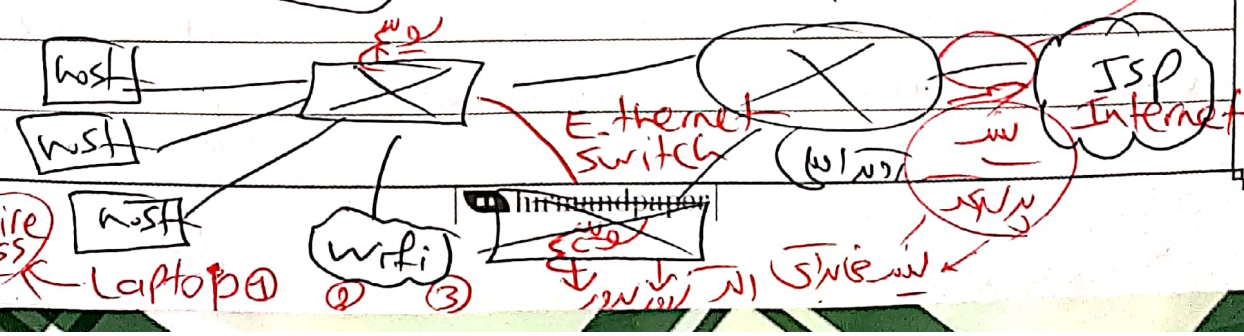
معمولاً این دو تا با هم حساب می‌شن یعنی هم wifi و هم کد را تبدیل می‌کنه

switch, wifi, modem (البته الان توی یه دستگاه ADSL هستند)

شبکه‌های enterprise مثل دانشگاه چی؟

به دفتر باند زیادی لازم دارند (ترافیک بین‌المللی می‌فرستند)

یه شبکه داخلی دارند که از switch ها و router ها تشکیل شده و یه روتر اصلی دارن که از طریق این می‌خواستم به اینترنت ISP وصل می‌شود



اگر LAN باشد local area network (شبکه‌های کوچک و محلی)

از wifi استفاده می‌کند

4-50 Mbps

شبکه‌های wireless access network

اگر WAN باشد wide area network

مثل 3G_4G و LTE

صداقت و بیع به

دلبستگی با امکان جابجایی و mobilibility

network core

داده‌ها از طریق access network به پورت‌ها و آدرس‌ها متصل می‌شوند

افرادی که یک host (end system) می‌خواهند به یکدیگر متصل شوند

بفرستند و در قالب packet متصل می‌شوند ← host ها پیام‌ها یا

message هایی که لایه‌های شبکه برای طرف مقابل را به packet تبدیل می‌کنند

روی هر یک آدرس مبدأ و مقصد نوشته شده هر یک به بخشی از دنیا را

و به آدرس داده شده دارد.

در core ← ① باید مسیرها را مشخص کنیم (چون بسته‌ای به

مقصد مشخص می‌آید ← باید تصمیم‌گیری کنیم از کدام مسیری باید به این مقصد برسد)

forward

② مسیری که مشخص شد، هر بسته را به این مسیر هدایت کنیم

اگر ایده‌ای داریم

host و فرستادن بسته‌های data

یک‌بار ① گرفتن پیام (message) از لایه application

② شکستن پیام به قسمت‌های کوچک (chunk) ← هر یک به اندازه‌ی

packet ها

L بیت است

یک access با rate R بسته را می‌فرستد

R = (rate _ انتقال بسته) یا (ظرفیت لینک) یا (پهنای باند لینک)

link bandwidth Link capacity

transmission delay

زمانی که نیاز است تا

تا فیزیکی انتقال بسته = L_bit بیت را به لینک تحویل دهیم

(bits) L

R (bits/sec) لینک

ارسال تا روی لینک

هر بیت $\frac{1}{R}$ ← بسته $\frac{L}{R}$ ← بسته

بعد از $\frac{L}{R}$ کامل ← همه‌ی بیت‌ها از host رفتن روی لینک ← اینکه

چه زمانی به مقصد می‌رسند فرق دارد. ← طول می‌کشد بسته‌ها به لینک ارسال بشوند

اگر فرض کنیم بین مبدا و مقصد یک نود میانی داریم، در packet switching

این یک که از مبدا ارسال شده، باید توی نود میانی هم

کامل دریافت بشه و ذخیره بشه و بعد دوباره به لینک بعدی ارسال شود

چرا packet switching این کار را می‌کند؟ ← store & forward

چون در هر یک ما، آدرس مقصد را می‌نویسیم و در نودهای میانی

بر اساس آدرس مقصد مشخص می‌کنیم از کدام مسیر می‌خواهیم نود بعدی

کدوم باشه. ← برای داشتن آدرس مقصد ← باید کل بسته دریافت بشه

خب چرا کل بسته؟ چرا همین کافی نیست برای فهمیدن آدرس مقصد؟ برای اینکه مطمئن

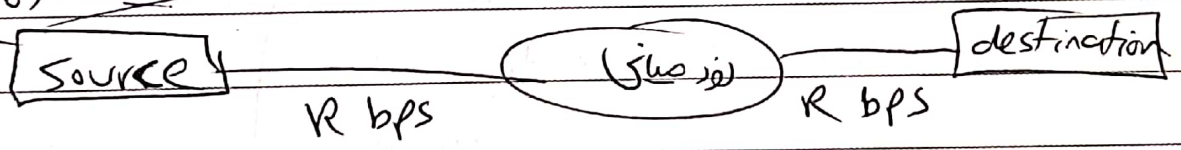
بشیم خطایی در دریافت بسته‌ها نداریم ← کنترل خطا برای کل بسته است.

با دیدن آدرس مقصد ← می‌فهمیم به کدام پورت خروجی باید ارسال کنیم

اگر N تا نود میانی داشته باشیم ← زمان ارسال $N \cdot \frac{L}{R}$

تا $\frac{L}{R}$ طول می‌کشد

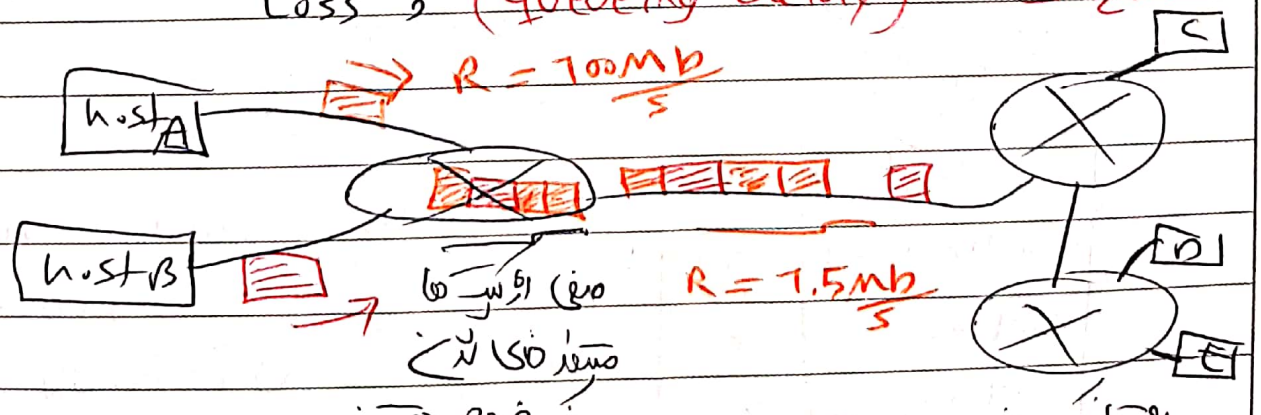
L بیت در هر پکت



end end delay = $\frac{2L}{R}$ (اگر propagation delay = 0)

زمانی که بسته طول می‌کشد تا به مقصد برسد

Loss و (queueing delay) تاخیر صف



صف از بسته‌ها

متفاوت شکل گرفتن

نیز خروجی هستند

ترافیک بین A و C

خروجی ارسال کنند هم ترافیک‌ها B و D

چون مسیر مشترک شده، همه بسته‌ها قرار است باید به پورت

hirmandpaper

Scanned by CamScanner

اگر rate < اینپوت پکت‌ها به لینک ۵ بیشتر از rate ← ارسال بسته

① پکت‌ها ۵ تشکیل صف می‌دهند و مسیر می‌کنند تا از طریق لینک ارسال شوند

برای یا صدت زمانی

② پکت‌ها می‌توانند drop شوند (lost شوند) اگر بافر پر بشه

عادی

در روتر باید بافر داشته باشه و حجم‌ش محدود است

یا بسته‌ای که توی صف قرار گرفته ۵ تا بیاد ته صف و بخاطر ارسال بسته ۵ ای خودش به زمانی طول می‌کشه ← برای تاخیر صف می‌گیم queueing delay

پس ممکن است تاخیر صف هم داشته باشیم

خصوصاً در core شبکه که حجم ترافیک بالاست ۵ تاخیر صف هم می‌تواند قابل توجه باشد

۲ بخش اصلی core شبکه

۲ تا کار متفاوت

① routing ← مسیری مبدأ به مقصد که پکت‌ها باید طی کنند رو تعیین می‌کنه

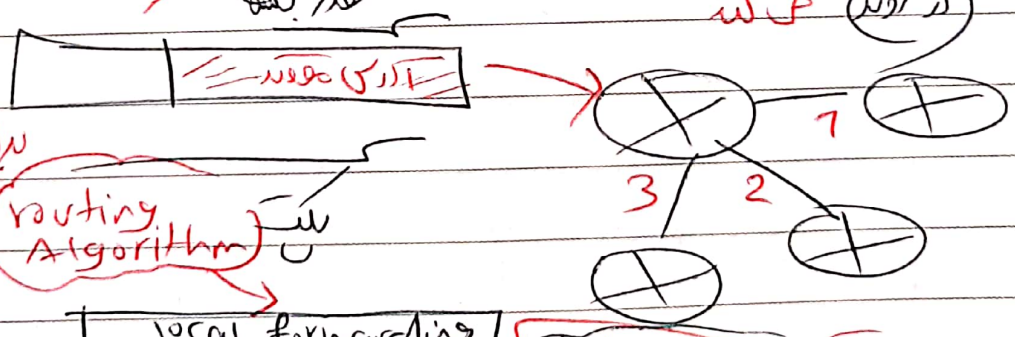
routing algorithms

② forwarding

بسته‌ای که میاد ۵ براساس آدرس مقصد و اطلاعاتی که از routing بدست اومده را به پورت مناسب ۵ هدایت می‌کنه

از پورت ورودی ۵ بسته را به پورت خروجی مناسب ۵ هدایت می‌کند

در روتر



برای

پکت

routing Algorithm

local forwarding table	
header value	output link
0100	3
0101	2
0111	2
⋮	⋮

شبکه باید در هر روتر ۵ عملیات routing را انجام می‌دهد

برای هر مقصدی ۵ چه مسیری مناسب است چه مقصدهای مختلف ۵ subnet هایی که به روتر ها وصل اند.

Scanned by CamScanner

اینترنت: شبکه‌ای از شبکه‌ها

end system ها با کمک ISP ها به Internet وصل می‌شوند

Internet service provider

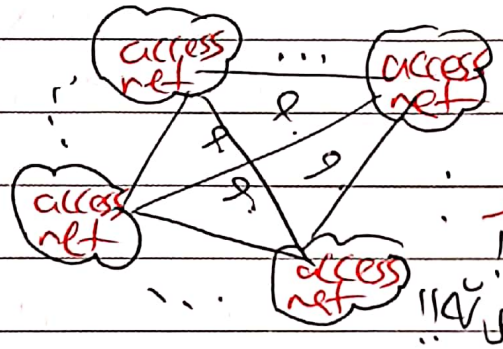
(residental) محلی ، company ، university

توسط Internet به املاک شرکت‌های خصوصی بوده است

یک کار اقتصادی و تجاری بوده

ISP ها در دنیای Access network شبکه ایجاد کرده‌اند. مثل شرکت‌هایی که سرویس ADSL ارائه می‌دهند. ISP ها با تکنولوژی‌های access متفاوت، سرویس می‌دهند

ISP ها باید به هم وصل باشند تا host های متفاوت بتوانند برای هم پیام ارسال کنند.

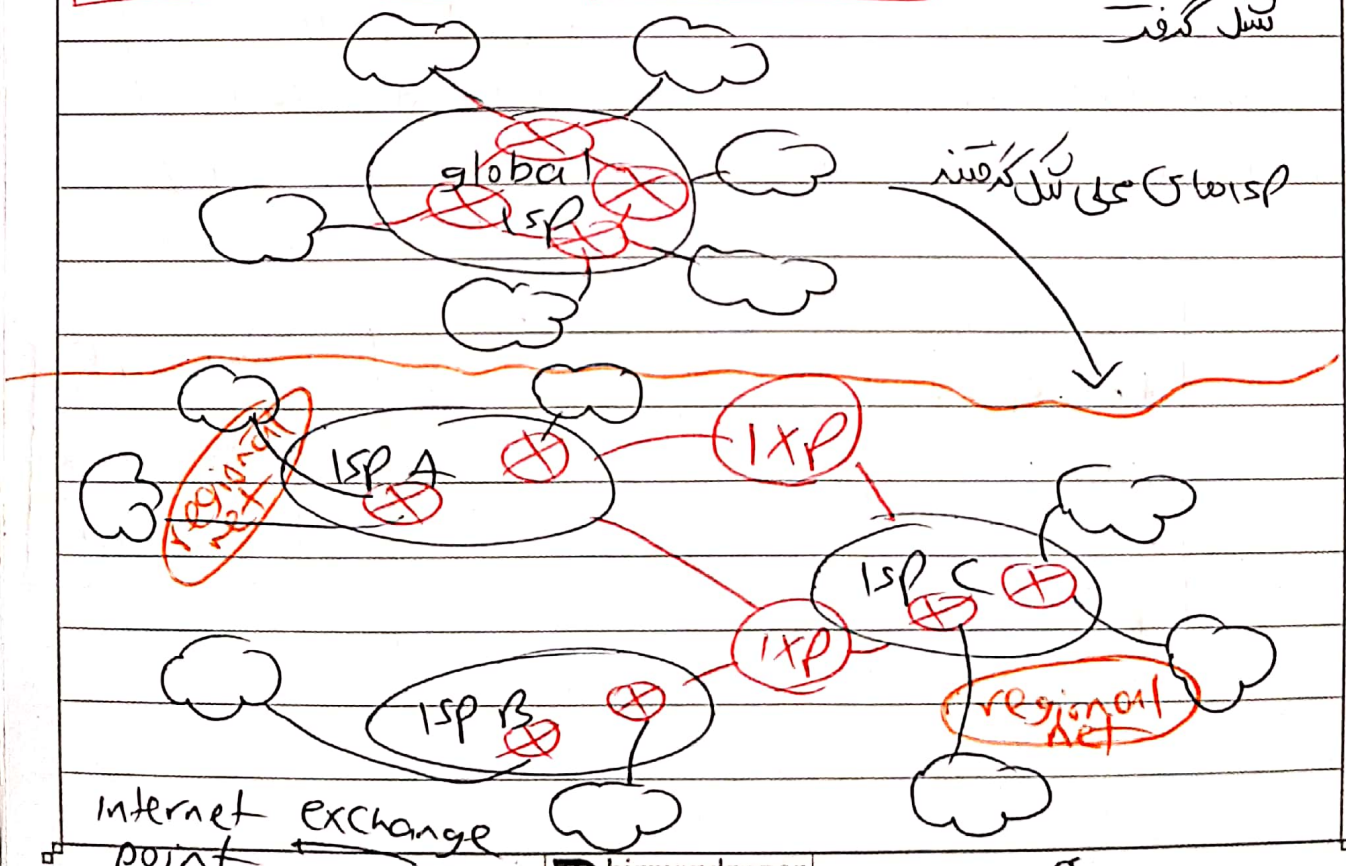


چطوری همه‌ی این ISP ها رو به هم وصل کنیم؟

می‌تونیم همه‌شون رو به هم یک به یک وصل کنیم!!

scale پذیر نیست چون $O(N^2)$ می‌شه!!

یه لایه‌ی شرکت اومدن زیرساخت ایجاد کردن تا access network ها بتوانند به هم وصل بشوند ← global ISP ها و regional ISP ها شکل گرفت



ISP های محلی شکل گرفتند

Internet exchange point

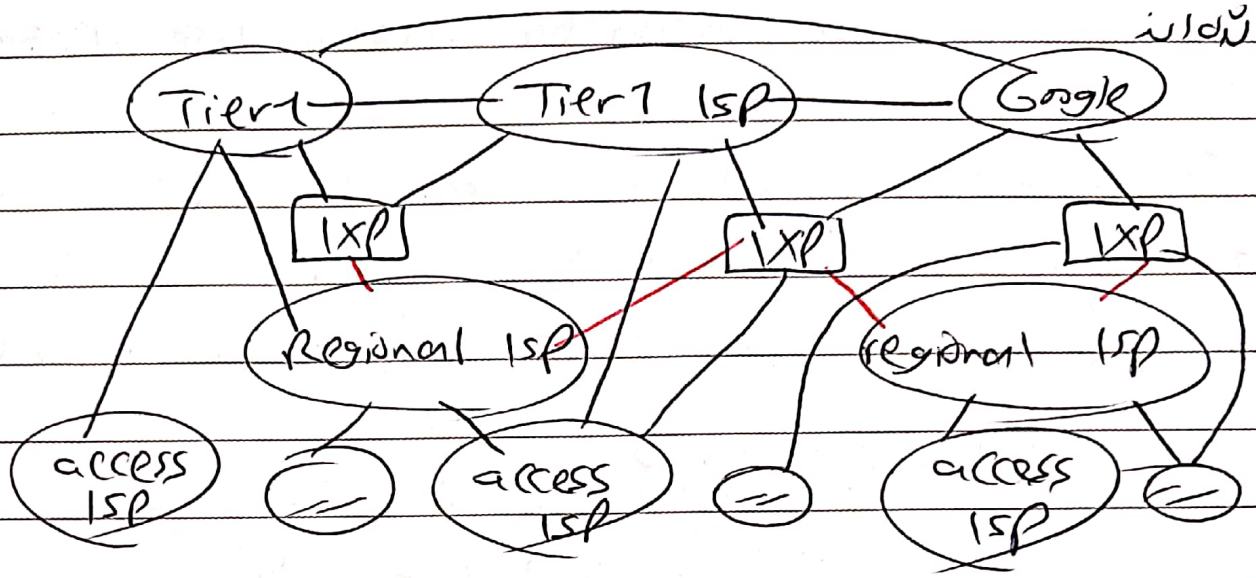
خود ISP های regional هم باید با هم ارتباط داشته باشند ← IXP به وجود آمد

Scanned by CamScanner

ISP های A و B و C توسط دیتای بالاتر مثل Google, Microsoft

به هم وصل می‌شوند ← بعضی Content provider network می‌گویند.

شبکه‌ای از شبکه‌ها ← بعضی ارتباط ISP ها هستند که در access و Core ایجاد شده‌اند



Session 2

دیتا لینک

پیام‌ها را در قالب segment از فرستنده برای گیرنده ارسال می‌کند

در سمت فرستنده segment ها را به datagram تبدیل می‌کند

encapsulate در واقع

در سمت گیرنده segment ها را تحویل دیتای transport می‌دهد

پروتکل‌های این دیتا در همه‌ی host ها و router ها هستند

روترها فقط فیلدهای هدر بسته‌های IP که به آن ارسال شده را بررسی می‌کنند.

application	end system	network
transport		data link
network		physical
data link		
physical		

★ کارهایی که پروتکل‌های مربوط به مسیریابی انجام می‌دهند

① forwarding

بسته‌ها که از پورت ورودی به یک node می‌رسد باید به یک پورت خروجی ارسال شود ← هدایت بسته‌ها، کار forwarding است

move packets

② routing

به کدام پورت خروجی باید ارسال شود؟ تصمیم‌گیری و مشخص کردن پورت مناسب با توجه به مقصد

determine routes

الگوریتم‌های مسیریابی

Data plane

local و برای هر پورت کار می‌کند ← forwarding انجام می‌شود

بسته‌ای که می‌رسد، با توجه به آدرسی مقصد که توی هدرش هست، به خروجی مناسب ارسال می‌شود

Control plane

منطقش network wide است

تعیین می‌کند چطوری یک datagram از مبدأ به مقصد برسد

از این چیزی آمد (مسیریابی بسته‌ها)

how datagram is routed?

برای هر link، مشخص می‌کند کدام پورت مناسب است؟

چطوری control plane این جدول را پر می‌کند؟

مسیریابی یا routing به وسیله‌ی routing protocols انجام می‌شود

① Link state

② distance vector

هدف: پیدا کردن بهترین مسیر برای ارسال بسته‌ها

hirmandpaper

Scanned by CamScanner

معیار بهترین مسیر چی است؟ Cost یک مسیر معیار است ← میتواند هزینه

پرداختی به ISP باشد، پرهیز از بودن مسیر یا مکث بودن ترافیک معیار باشد
تعداد hop های میانی یا عبوری باشد یا

فرمت datagram IP

۳۲ بیت

۳۲ bit source IP addr
" dest " "
data

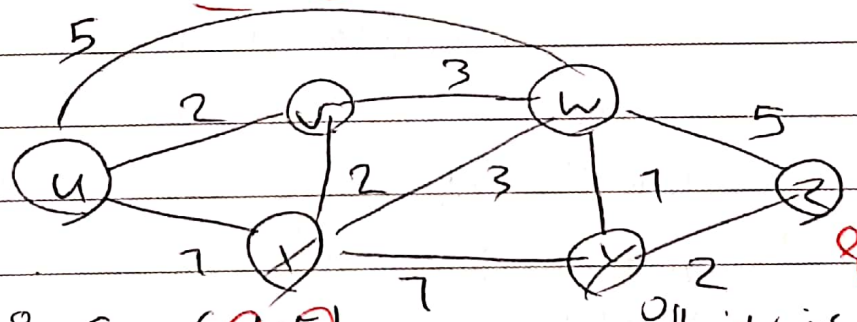
برای انتخاب مسیر در جدول ای که در مسیریاب وجود دارد

طولش متغیر است

مهولا یک segment TCP یا UDP است

* پروتکل های مسیریابی، مسیر را از روی گراف شبکه پیدا میکنند
گراف شبکه یک abstraction از topology شبکه است
که از node ها و Edge ها تشکیل شده

لینک های بین روترها — روترها



وزن یال ها Cost بین دو روتر — مبنا

graph: $G = (N, E)$

Cost با طول مسیر یا کیفیت باند باشد
نسبت مستقیم — عکس

N = set of routers = {u, v, w, x, y, z}

E = " " links = {(u,v), (u,x), (v,w), (v,x), (w,z), (w,y), (z,y), (w,x), (x,y), (u,w)}

بهترین مسیر مسیری که Cost لینک های مسیر کمترین باشد

Cost of path $(x_1, x_2, \ldots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)$

Cost کل مسیر — Cost لینک ها

hirmandpaper

Scanned by CamScanner

طبقه بندی الگوریتم های مسیریابی

سوال: ① اطلاعات global است یا decentralized؟

اطلاعات گراف شبکه و cost لینک ها

همه ی روترها: اطلاعات topology شبکه را دارند

الگوریتم های Link state

هر node فقط اطلاعات فوری و همسایه های را دارد و از کل شبکه اطلاعاتی ندارد (درحقیقت از اطلاعات را دارد)

هر node به بالای لینک همسایه هایش چه روترهایی هستند و فاصله اش تا همسایه ها چی است و با لینک همسایه ها و مسیرهایی را برای مقصدهای در شبکه دارند ← مقصدهایی مورد نیاز خودش را پیدا می کند

روتر فقط اطلاعات همسایه هایی که به صورت فیزیکی متصل هستند را می داند

همسایه ها و اطلاعات خودشان را با هم مبادله می کنند

مثال: الگوریتم های distance vector

سوال ② الگوریتم static یا dynamic؟

به طور ثابت و یک مسیری را مشخص می کنند و مسیرها خیلی آرام و کم در طول زمان عوض می شود (مثل تابلوهای ثابت در ورودی خیابان ها)

به بالای وضعیت شبکه، الگوریتم، مسیرها را تغییر می دهد. مثل تابلوهای الکترونیکی که مثلا (یکی از مسیر به بالای ترافیک) مدام (periodic) آپدیت می شود

اطلاعات از طریق advertis های شبکه از طریق روترها منتقل می شود

تغییرات و پاسخی به تغییرات cost لینک ها است.

الگوریتم Dijkstra ← Link state

روتر اطلاعات شبکه را دارد

الگوریتم های مسیریابی

distance vector

Berrman ford

hirmandpaper

Scanned by CamScanner

الگوریتم Dijkstra

از نوع Link state

همه node های شبکه، توپولوژی و گراف شبکه را می‌دانند

از طریق Link state broadcast

همه نودها، اطلاعات یکسان دارند

هر روتر، لینک‌هایی که به همسایه‌هایش هست و هزینه‌ها را لیست می‌کند و به همه روترهای شبکه می‌فرستد (broadcast می‌کند)

هر روتر، با وسیله همین broadcast، اطلاعات کل شبکه را بازسازی می‌کند

اطلاعات کل شبکه (گراف) را به صورت یک دیتابیس بدست می‌آورد

کم هزینه‌ترین مسیر را از یک نود (source) به همه نودهای دیگر، به کمک الگوریتم دوکسترا محاسبه می‌کند و به نودها می‌دهد

مقصد ← بهترین مسیر را پیدا می‌کند از دوکسترا با همه مقصدها ← یک درخت به کمک مسیاب که ریشه‌اش، خود این نود است و برگ‌هایش، مقصدهای دیگر هستند

نودهای دیگر

به کمک این درخت، اطلاعات forwarding table به کمک مسیاب

در هر نود، برای همه مقصدها مشخص می‌شود چه پورتی خوبه ولی توی جدول forwarding فقط همسایه‌هایش را می‌نویسد تا همه مقصدها را (می‌بیاد می‌کند ولی توی جدول می‌نویسد)

به صورت iterative است

بعد از K تا iteration، کم هزینه‌ترین مسیر را به K تا مقصد می‌داند

هزینه‌ی لینک از نود x به y = $c(x,y)$

اگر همسایه نباشند ← ∞ می‌گذاریم.

فاصله یا هزینه‌ی یک نود تا مقصد v = $D(v)$

نودی است که مسیر قبل از رسیدن به مقصد v از آن عبور می‌کند، $p(v)$ ← v

hirmandpaper

Scanned by CamScanner

N' : مجموعه نودهایی که توی درخت مسیر قرار گرفته‌اند.

این نودهایی که کمترین هزینه برای مسیرهای مختلف به صورت قطعی پیدا شده (مسیر مشخص شده)

الگوریتم

Initialization

$N' = \{u\}$

برای همه‌ی نودهای v

فاصله u تا v

اگر v با u همسایه است $\rightarrow D(v) = c(u,v)$

در غیر این صورت $\rightarrow D(v) = \infty$

Loop

w را پیدا کن که ① توی N' نباشد ② D(w) ی min باشد

w را به N' اضافه کن

کوتاه‌ترین فاصله (کمترین هزینه)

D(v) را برای نودهایی که (v) همسایه‌ی w هستند و توی N' نیستند به شکل زیر آپدیت کن

این کوتاه‌ترین مسیر به w

$D(v) = \min(D(v), D(w) + c(w,v))$

cost جدید به v | این قبلی | cost از u به v

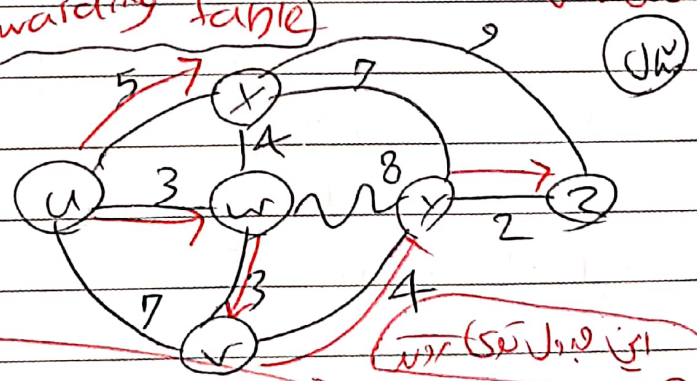
= از u تا v + از v تا w

تا همه‌ی نودها توی N'

از v تا w

قرار بگیرند

routing table یا forwarding table



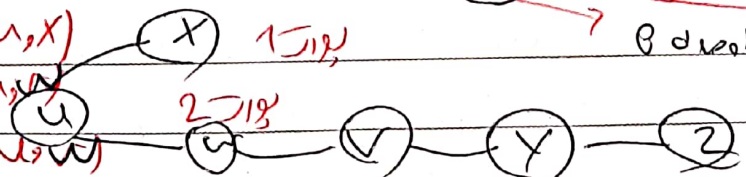
مثال

مسیر u تا z

پیدا کن کوتاه‌ترین مسیر از u به بقیه نودها

destination

مقصد	لینک
x	(u,x)
w	(u,w)
v	(w,v)
y	(w,y)
z	(w,y)



این جدول توی روتر در شبکه درخت بلند شده‌ی u

data plane

نوشته می‌شود

پس توی جدول forwarding u باید برای رسیدن به w (u,w) و برای رسیدن به x (u,x) بدهیم و برای رسیدن به y و z از (w,y) بدهیم.

hirmandpaper

Scanned by CamScanner

step	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

بگیریم بسته بیاد و بخواهد از w به z برود ← با u که صرف 6 به جدول که توی روتینگ هست نگاه می‌کند و می‌فهمد باید به روت 2 بسته را بفرستد
بعد که با روتر w رسید، w باید به جدول خودش نگاه کند و تصمیم بگیره ←
بسته

هر روتر با جدول خودش نگاه می‌کند بسته‌ها و دیتا را به مقصد می‌فرستد

کوتاه‌ترین مسیر از بین همسایه‌ها

Distance vector — الگوریتم‌های

dynamic programming — Bellman ford

هزینه‌ی کوتاه‌ترین مسیر از x به y

مقصد — نود x

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

این مسیر از نود x با x و v فرق می‌کند از همسایه‌های x باید عبور پیدا کنه ← ما باید بگیم از فرق کدوم همسایه‌ی x بهترین مسیر را دارم و اگه همسایه‌ها را با v نشان بدیم

روی v و min را پیدا می‌کنیم

هزینه از همسایه (v) تا مقصد y ← محاسبه می‌کنیم فرض می‌کنیم همسایه خودش محاسبه کرده ←

هزینه‌ی x تا همسایه‌ی v

همسایه باید این $d_v(y)$ را بهمان بدهد

$d_x(y)$ را اگر برای همه‌ی مقصدهای توی شبکه از نودی (مبدأ) x به لیست بیاریم

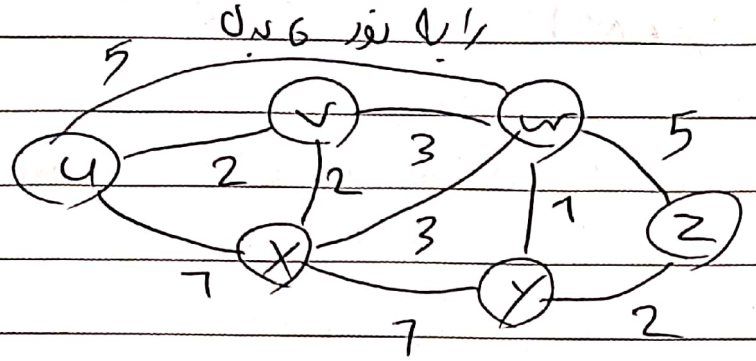
Scanned by CamScanner

forwarding table را می‌توانیم پر کنیم با کمک دو قدم بدست آمده

تنها اطلاعاتی که هر نود دارد ← هزینه فرستادن تا همسایه‌هایش

همسایه باید هزینه فرستادن تا مقصد را به نود u بدهد

مثال



فرض کن اطلاعات مثل $d_v(z)=5$ ، $d_x(z)=3$ ، $d_w(z)=3$ را به ما دادند.

می‌خواهیم کوتاه‌ترین فاصله از u به z را پیدا کنیم ← همسایه‌های u می‌شوند v و x و w ← باید کوتاه‌ترین مسیر این همسایه‌ها را تا z داشته باشیم که توی سوال داده و با فاصله‌ی u از همسایه‌هایی که داریم از گرافت جمع کنیم و min را پیدا کنیم ←

$$d_u(z)=\min\{c(u,v)+d_v(z),\ c(u,x)+d_x(z),\ c(u,w)+d_w(z)\}=$$

$$\min\{(2+5),(7+3),(5+3)\}=$$

$$\min\{7,4,8\}=4$$

u باید x را به عنوان همسایه انتخاب کند برای رفتن به z

اگر همین کار را برای مقصدهای دیگر و غیر از z انجام بدهیم (یعنی کوتاه‌ترین فاصله از u به نودهای دیگر) را پیدا کنیم ← می‌شود جدول مسیریابی u را برای همه مقصدها پیدا کرد

حذف x هم اطلاعاتش را از همسایه‌های مثل u می‌گیرد ← مجدداً به هم واکنش‌اند.

hirmandpaper

Scanned by CamScanner

$D_x(y)$ و تخمین از کمترین cost از نود x به y

همسایه‌ی و distance vector فوق را برای همه‌ی مقصدها دارد.

$D_x = [D_x(y) \text{ و } y \in N]$

node x و ① با توجه به اطلاعاتی که از همسایه‌هایش دارد $C(x,v)$

x و همسایه‌های v ② با توجه به distance vector همسایه‌های

برای هر همسایه به اسم v ← x و مقدار $D_v = [D_v(y) \text{ و } y \in N]$

را نگه می‌دارد.

پس با کمک ① و ② ← x و distance vector فوق را برای

همه‌ی مقصدها و آپدیت می‌کند

باید distance vector همسایه‌های

را داشته باشد ← هر همسایه باید را به x

داده باشد

عملکرد الگوریتم و ابتدا اطلاعات هیچ کدام از نودها واقعی نیست و فقط اطلاعات

همسایه‌ها نوع را دارند ولی اگر همسایه و اطلاعات را با هم رد و بدل کنند

همسایه‌های همسایه‌ها و ← اطلاعات نیز می‌شود

و را اگر یا نود b تا مسیر را می‌شناسد باید ببیند و کدوم کمتر است

وقتی x و یک DV جدید از همسایه دریافت می‌کند و DV فوق

را با کمک معادله‌ی $B-F$ و آپدیت می‌کند

$D_x(y) \leftarrow \min_v \{ C(x,v) + D_v(y) \}$

برای هر node ، $y \in N$

فاصله‌ی فوق تا همسایه

فاصله همسایه تا مقصد y

از x برای مقصد y

اگر تغییری در لینک و لینک‌های اتفاق بیفتد ← یا لینکی و قطع بشه یا node ای

down بشه ← الگوریتم باید دوباره اجرا بشه ← این را نود که فرستنده

قطع لینک یا همسایه‌ای node ای که down شده و تشخیص می‌دهد و تذکر

hirmandpaper

Scanned by CamScanner

می‌کند الگوریتم را دوباره اجرا می‌کند

ادامه الگوریتم — هر local iteration توالی ۲ مورد انجام می‌دهد: تغییرات در local link cost یا پیام آپدیت DV از همسایه

iterative و asynchronous

Distributed — هر نود به DV خود نگاه می‌کند که آیا کوتاه‌ترین مسیر است — اگر لازم باشد به همسایه‌ها به همسایه‌های خودش را آگاه می‌کند.

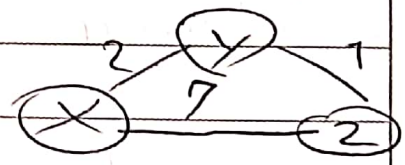
هر node: سعی کند تا یک تغییری در local link cost ها ایجاد بشه یا یک پیامی از همسایه بیاد — wait

recompute

همسایه‌ها را آگاه کن — اگر DV با هر مقصدی به خودش

iteration اول: هر نود DV خودش را به همسایه‌هایش می‌دهد — بعد از یک iteration x DV x و z را دارد.

بعد بر اساس این DV ها و فاصله‌ای که خودش با همسایه‌هایش دارد میاد DV خودش را تا مقصدها آپدیت می‌کند



توی اینجا: فاصله‌ی x تا y: 7 ، فاصله‌ی مستقیم x تا y

فاصله x تا z و z تا y: 2 $\longrightarrow$ $\min\{2 \to \cancel{7+7}\,8 = 2\}$

برای x تا z $\longrightarrow$ 7 فاصله مستقیم: 7

از طریق x تا y و z تا y: 2 $\longrightarrow$ $\min\{7, 3+4\} = 3$

$2+7=3$

DV اگر آپدیت بشه، همسایه‌ها می‌دهیم اگر نشد کاری نمی‌دهیم!!

اگر یک تغییری توی DV بدالگوریتم

یعنی مسیرهای را پیدا کرده‌اند.

hiriandpaper

Scanned by CamScanner

LS و node ها ← link E

$O(nE)$ تا بلکه مخابره

از نظر message complexity

DV و تبادل مقادیر بهترین مسیرها

زمان همگرایی و متغیر

ممکن است (نوسان) داشته باشد

مقایسه‌ی LS و DV

پیچیدگی LS و $O(n^2)$ ← پیچیدگی

DV و زمان همگرایی متغیر

ممکن است routing loop یا count_to_infinity

در شبکه‌های بزرگ LS و Dijkstra التقاوی شود

اگر روی یک مسیر وجود داشته باشد

در LS هر نود می‌تواند link cost ـ نادرست را advertise کند

هر نود فقط جدول خودش را محاسبه می‌کند

DV و node ها می‌توانند path cost ـ نادرست را advertise کنند

هر نود نوعی توافق بقیه دارد

(جدول) التقاوی می‌شه

error توی کل شبکه پخش می‌شه.

توی LS اگر یه جدول نودی تغییر کند ← broadcast می‌شه به همه‌ی نودها و توی فاصله‌ای که همه‌ی نودها آگاه بشوند، اگر شبکه بزرگ باشه، طول می‌کشد و نیست مسائل و مسیرها که امداح نشده‌اند، چون یه‌لحظه‌ی کیت‌ها از این نیت‌ها عبور می‌کنند (به خاطر مسیریابی جهانی) ← اطلاعات از دست میره.

DV ← اگر اطلاعات یک نود بروز نباشد یا غلط باشد، اطلاعات کل شبکه را می‌تواند به هم بریزد. ← آپدیت به همه نودهای شبکه هم الت

routing ← global است ← برای همه‌ی شبکه باید انجام بشه

session 3 — forwarding ← فقط در Router انجام می‌شه

local است

Data plane ← مسیری که forwarding را انجام می‌دهد

control plane ← جایی که routing انجام می‌شه

شرط اینکه routing درست انجام بشه ← اینه که همه‌ی روترها اطلاعات یکسانی داشته باشند ← مشارکت برای اینه که یکسان باشد اطلاعات

روترها نقشه داشته و باید توزیع شده

توی LS ← مشارکت از طریق دادن اطلاعات بین روترها باشد

بود و هر روتر به تنهایی الگوریتم را اجرا می‌کرد و درخت مسیرها را بدست می‌آورد و forwarding table خودش را set می‌کرد. توی LS اطلاعات لوزها باید یکسان باشد اگر یه لوز درخت را به طوری بدست بیاره و لوز بعدی یه طور دیگر ← یک قراره از یه مسیر بره و توی لوز دوم یه مسیر دیگری میره

در DV ← یک لوز path vector همسایه‌هایش را اعلام می‌کنه path vector خودش را هم به همسایه‌هایش می‌دهد ← اینجا هم مشارکت لازم است.

← در حالت سنتی control plane در همه‌ی روترها باید اجرا بشه یعنی distributed است در همه‌ی روترها.

control plane ← برای اجرای پروتکل‌های routing استفاده می‌شود

Data ← برای forward کردن بسته‌هاست.

دیدگاه جدید ← جدا کردن control plane ها ← روش SDN

software-defined networking

توی روش SDN ← control plane به صورت remote در سرورها پیاده‌سازی شده است.

(معماری Router) بیشتر از ۳ قسمت تشکیل شده

① port های ورودی و خروجی

② switching fabric ← شبکه ارتباطی بین پورت های ورودی و خروجی

از ویژگی های این شبکه یکی که از پورت ورودی میاد باید بتوانه به یک پورت خروجی ارسال شود (همه پورت های ورودی باید بتوانند به همه پورت های خروجی به بستر ارسال کنند)

③ Routing processor ← دستگاه کنترلی روتر

اجرای Routing protocol

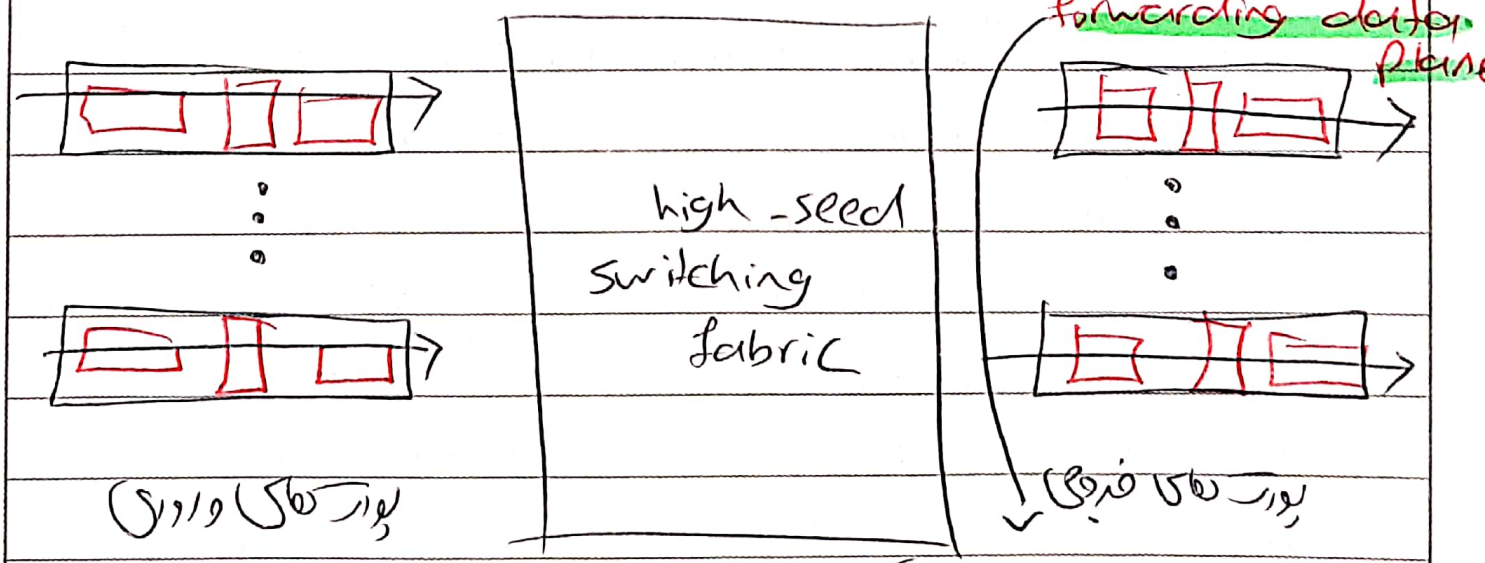
forwarding ← بین پورت های ورودی و خروجی از طریق switching fabric انجام میشه

نرم افزاری ← در حد میلی ثانیه

اجرای پروتکل های routing ← بدلیل کندی در حد

routing processor

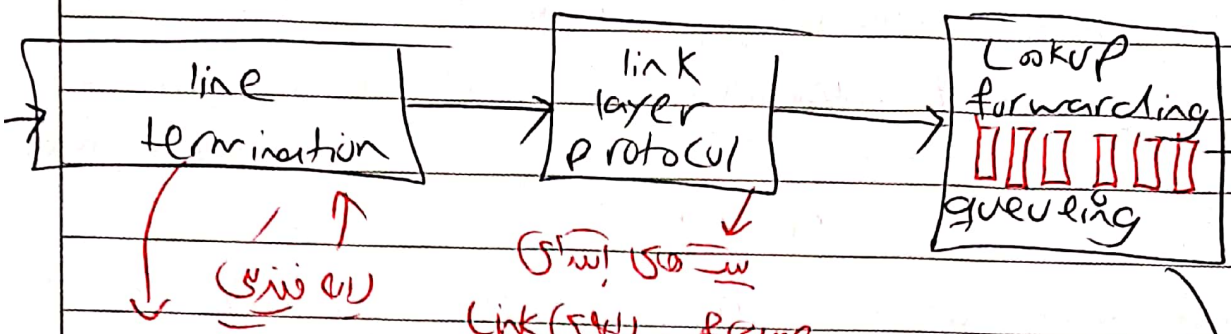
Control plane



سخت افزاری ← در حد نانو ثانیه

عملیات forward کردن فیلی سریع انجام میشه چون مقدار زیادی بسته باید در زمان کوتاهی ارسال شوند

(پردازش port ورودی)



لایه فیزیکی

bit level reception

لایه پیوند داده

پیت های ابتدای frame را تشخیص می دهد و frame ها را جدا می کند

یکی که توی frame هست و جدا می شود و به لایه بالاتر داده می شود

frame ها را برای لایه ۳ آماده می کند و کارهای لازم را انجام می دهد

کارهای

بررسی بسته لایه لینک مثل error control و سپس آدرس مقصد و

براساس lookup ها که توی جدول مسیریابی انجام می دهد و تشخیص می دهد از

طریق switch fabric به کدام پورت خروجی باید ارسال شود

هدف ۶ پردازش port ورودی در نرخ line speed انجام

بشه. چرا؟ چون تا یک بعدی نرسیده و یک بعدی باید فرستاده شده باشد

forwarding rate

هدف یا queueing ۶ اگر datagram ها ۶ سریعتر از نرخ ارسال

وارد switch fabric بشوند ← صف ایجاد می شود و باید اونجا بسته ها منتظر بشوند

* پردازش های پورت ورودی ۶ line speed انجام می شود ولی

ارسال به پورت خروجی ۶ بستگی به مسیریابی switch fabric می تواند تعداد

بسته ← پیت باید پهنا باند ← باید باشد داشته باشیم

segment = آی پی - payload) TCP

در هدر بسته لایه لینک

توی هدر TCP مشخص می شود

lookup براساس آدرس مقصد پیت است

داده ارسال می شود

آدرس آی پی ۳۲ بیتی

آدرس مقصد هم داریم اگر

در هدر آی پی

برای application ها

hirmandpaper

هدر TCP هم می خوایم (در اورد)

* توی یک firewall ← صف براساس آدرس مقصد تصمیم می گیریم جلو اش را بدهیم

Scanned by CamScanner

برای firewall ← همه داده‌های دیگر را هم نادیده می‌گیریم و به لایه‌های اوپر هم

تحویل می‌گیریم. سوال ← چرا برای forwarding این کار را نکنیم؟
یعنی وقتی بسته میاد، برای اینکه تصمیم بگیریم کجا ارسال بشه و چرا فقط کدوم بشه
به اینکه آدرس مقصد چیه؟ صدا به لایه Application و آدرس مقصد
تصمیم بگیریم. IP مقصد

TCP port number

چرا شماره پورت TCP هم الت؟ اگه از یک نود به مقصد دو تا مسیر داشته باشیم یکی گرون‌تر باشد ولی دقت و امنیت بیشتر و دیگری ارزان‌تر ولی احتمال گم شدن بسته بیشتر الت. ← ما میایم Application ای که تأخیر برای هم الت را از مسیر اولی می‌فرستیم ولی Application ای که گم شدن بسته برایش هم نیست را از مسیری که هزینه کمتری داره می‌فرستیم (یا تأخیر)

اگه فقط به مقصد نگاه کنیم نمی‌تونیم این تفاوت را قائل شیم

پس ۲ نوع forwarding داریم

destination_based forwarding (1) ←

forward فقط بر اساس آدرس IP مقصد

generalized forwarding (2)

بر اساس مجموعه‌ای از field های header ها

(2) دارای جامع‌تر (1) می‌باشد. بر اساس flow ها

Application ها یک سری‌های

ای که از یک مبدأ مشترک به مقصدی

میرن.

destination

مقصد	port

جدول مسیریابی

longest prefix matching ← طولانی‌ترین prefix ای که با آدرس مقصد

یک match بشه انتخاب می‌کنیم

به وسیله TCAM پیاده‌سازی

ternary content

addressable memories

hirmandpaper

Scanned by CamScanner

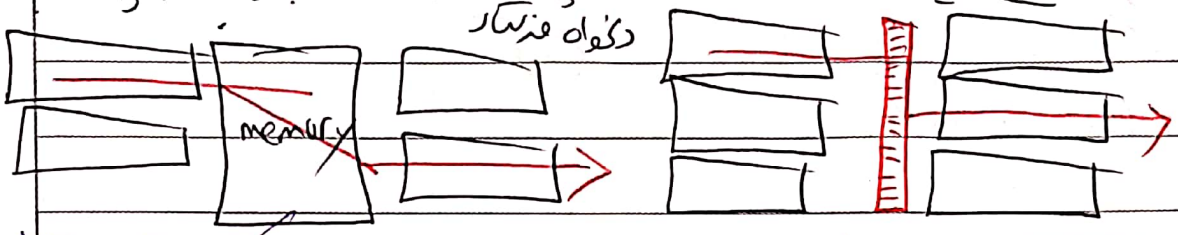
فرستادن پکت ها از پورت ورودی به خروجی

switching rate : نرخی که پکت ها می توانند از ورودی ها به خروجی ها ارسال بشوند

switching fabrics

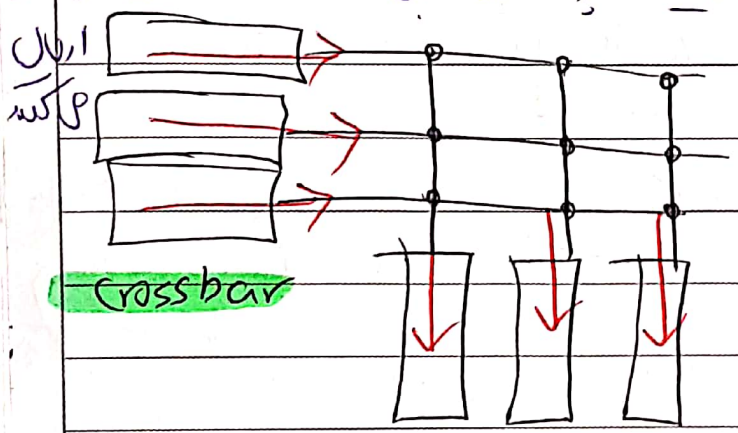
۳ نوع داریم :

از هر پورت ورودی دکول باید بشه به پورت خروجی دلخواه فرستاد



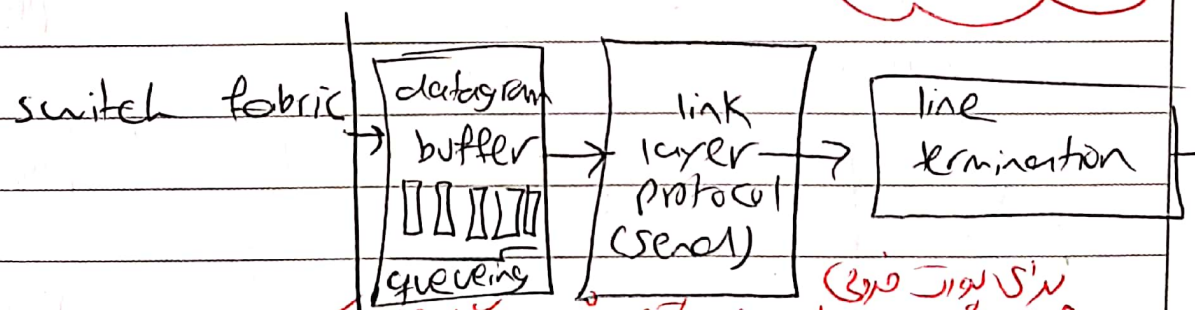
همه پورت های ورودی ، پکت را توی حافظه memory می نویسند و پورت های خروجی از حافظه می خوانند و ارسال می کنند

shared



اگر پورت های مقصد متفاوت باشند ضربت : در هم زمان ۳ تا پکت می توانند از ورودی ها به خروجی ها ارسال شوند ولی در مورد پایین ، نوبت هم است.

output ports



برای پورت خروجی

هم باید بفرستد و هم پکت دریافت کند

buffering ← وقتی نرخ می دهد که دیتاگرام ها از fabric سریع تر از نرخ ارسال (transmission rate) برسند.

اگر صف طولانی تر شد ← تاخیر صف ایجاد می شه

scheduling ← روال ها و پشته داریم برای مکاتبه تعیین است و تاخیر برای هم است (نباید معطل بشه) و پشته دیگری داریم که برای انتقال file است و تاخیر برای هم نیست ← وقتی صف تشکیل می شه و این دوتا صف مستقل هستند

برخلاف اینکه ممکن است file transfer ، زودتر اومده باشه ← ممکن است ما مکاتبه بشه

hirmandpaper

Scanned by CamScanner

تلفنی را نوبت نمی‌گیریم. چون delay برایمان مهم است

scheduling کردیم

در صف تشکیل بشه و بسته‌های از نوع‌های مختلف داشته باشیم و می‌تونه scheduler کند priority scheduling

* datagram ها packet ها می‌توانند گم بشوند بخاطر Congestion یا کمبود بافر

① ②

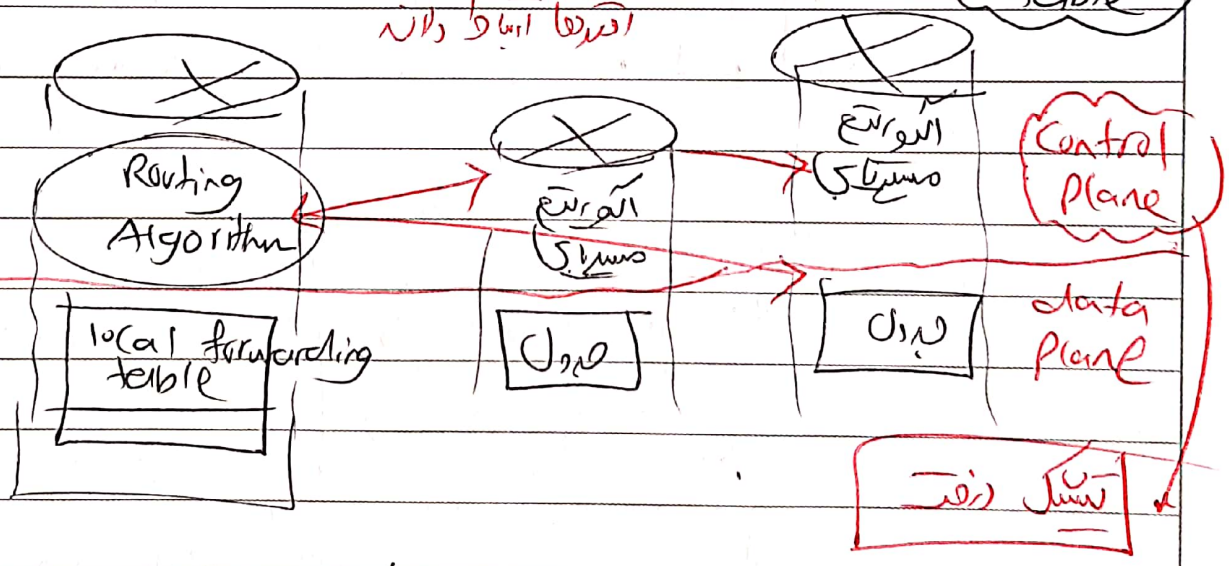
یک کسی بیاد performance را بالا ببره

مسائل data plane ← مدیریت ارسال و صف و scheduling و packet loss

دیدگاه کلی: هر روتر control plane خودش را دارد.

کارها و اجرای الگوریتم routing با مشارکت روترهای دیگر → خروجی: forwarding table

اتصال‌ها ارتباط دارند



روش جدید به جای اینکه یکی بیاد کل شبکه توی هر روتر جداگانه باشد، میاییم یک روتر می‌گذاریم و تعدادی یکی از این بیاد توی نرم‌افزاری می‌کنیم این روتر الگوریتم Dijkstra را به عنوان سورس برای همه روترها اجرا می‌کند بعد برای هر روتر، جدول‌شون را استخراج کند و بفرستد

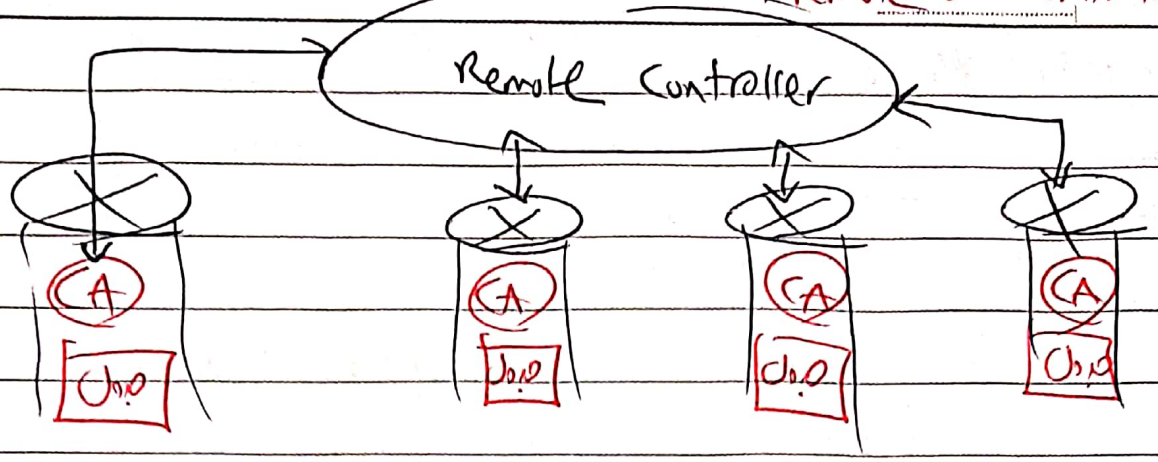
Controller میاد الگوریتم‌های مسیریابی را اجرا می‌کند

در هر روتر ← یک CA یا control Agent داریم. local

hirmandpaper

Scanned by CamScanner

remote ی controller



با این کار ← control plane را از data plane جدا کردیم

پس الان روترها عاملهای ارسال مسیرها فقط data plane را انجام می‌دهند

software defined networking SDN ←

Controller ی یا کنترلر الان ، الگوریتم‌ها نرم‌افزاری اجرا می‌شوند

← چون الان هر الگوریتم مسیریابی روی یک پردازنده باشد مثل dijkstra یا الگوریتم‌های بعد

روترها نقاط حساس و کلیدی هستند ← هم اند که قابل اعتماد باشند ← هزینه‌ها بالا و وقتی که به صورت distributed (روش سنتی) ها ی زیاد اند الگوریتم‌های مسیریابی را پیاده می‌کنیم ی کارشان حساس‌تر می‌شود ← وقتی روی لودر می‌بریم ی هزینه‌ها پایین‌تر می‌شود و چون پایه لودر داریم ← می‌توانیم به ی reliable کند ی از backup گرفت و بار روترها ی کمتر می‌شود و هزینه‌ها کم می‌شود.

* در SDN ی generalized forwarding داریم *

یعنی دیگر مدرها ی destination based نیست

هر روتر یک flow table دارد.

این جدول ی توسط یک controller مرکزی ایجاد می‌شود

در روترهای کنونی ی فقط local flow table

header (آدرس mac address)	counters	actions

فقط برای آدرس مقصد ها پورت مقصد ها یا پورت خروجی

فقط forward می‌کنیم

ی از ی action های مختلف داریم ←

↑ اضافه یزیکی در اول

مثال پورت آدرس ی پورت ورودی

hirmandpaper

Scanned by CamScanner

logically-centralized routing controller

control plane

data plane

local flow table

flow table را برای هر روتر ایجاد می‌کند و برای ما می‌فرستد و CA

که تمام روتر هست، flow table را ذخیره می‌کند در روتر

مزیت مهم SDN ← نرم‌افزاری بودن آن

openflow data plane

flow ← برای فیلدهای هدر مشخص می‌شود

مبدأ به بسته‌هایی که آدرس یکسان، آدرس مقصد یکسان دارند flow می‌شوند

packet handling ← قوانین لازم برای generalized forwarding

pattern ← match کردن فیلدهای هدرها

action ← برای بسته‌هایی که match شده‌اند

drop (1) ، forward (2) ، modify (3)

matched packet را ارسال به controller (4)

firewall در

priority ← یک مکانیزم است با چیزهای flow ها match شده

باید اولویت بدهیم که کدام match باید مبنا قرار بگیرد

counter ← packet ها، byte ها

برای کارهای مدیریتی و دارایی مفید است.

hirmandpaper

match + action ← قوانین

برای flow table

پروتکل است که controller و مسیریاب‌ها و اندرسها و ← openflow

ارتباط برقرار کند و جدول flow table ها را برایش فرستد

پروتکل کلی اجرای روی SDN

مثال ① src = 1.2.** , dest = 3.4.5* → drop

② src = *.*.*.* , dest = 3.4.*.* → forward(2)

همه آدرس‌ها — به پورت ۲ بفرست (لکسلی)

③ src = 10.1.2.3 , dest = *.*.*.* → send to controller

مبدأ یک آدرس — مقصد هر آدرسی باشد

به کنترلر می‌فرستیم

هر پکتی که از این آدرس بیاد

match و action را انجام می‌دهد

① Router ← longest prefix و match

match و action

forward out a link

② switch ← مقصد mac آدرس و match

find & forward و action

③ firewall ← TCP/UDP port و IP address و match

action و اجازه عبور یا منع

④ NAT ← port و آدرس IP و match

port و آدرس port rewrite و action

کوتاه کردن تعداد port

کلاس‌ها که openflow می‌تواند انجام بدهد

مثال و destination based forward ① (مثال)

Dest IP	Action
51.6.0.8	port 6

forward تنها پکتی که از پورت ۲۲ میاد

② forward و firewall

src IP	dest IP	TCP sport	TCP dport	
*	*	*	22	drop

hirmandpaper

Scanned by CamScanner

switch port	MAC Add.	MAC dest.	Eth type	VLAN ID	IP src	IP dst	...	action
*	*	*	*	*	128.119.1.1			drop

پکت هایی که توسط آدرس IP فرستاده می‌شوند را drop کن

(switch) (لایه ۲) ، destination based forwarding (A) *

switch port	MAC src	MAC dest	IP src	IP dst	IP Port	Action
*	22 BA 78 23 8 77 8	*	*	*	*	port3

frame هایی که برای (لایه ۲) هستند از MAC add فلان مبدا ← باید به پورت ۳ فرواردینگ

← اوتر می‌تواند switching (لایه ۲) را انجام بدهد برای MAC آدرس

session4

از 2005 شروع شد.

دیدگاه کنترل در هر اوتر انجام می‌شد

control plane ← دیدگاه امروزی SDN ← کنترل به صورت logically centralized است

باید توپولوژی کل شبکه باشد ← جداکردن control plane از اوتر

routing ← مسئله‌ای است که برای سطح شبکه است global

local نیست

در دیدگاه کلی ← اوتر یک لیست سیاه‌ها است که هم forwarding را انجام می‌دهد و هم مسیریابی

باید برای کارهایی مثل firewall ها و NAT box ها ← با کمی‌هایی جداگانه بشه انجام

hirmandpa

Scanned by CamScanner

چرا control plane را Logically Centralized کردیم؟

مزیت های SDN

① مدیریت آسان تر شبکه

② اجتناب از تنظیم های نامناسب روترها و انحراف ترافیک بسته ها

③ عیب یابی (دیباگ) آسان تر است

table based forwarding ②

openflow API

می توان روتر ها را برنامه نویسی کرد

controller و table را برای هر flow برنامه نویسی می کند

برنامه نویسی distributed و سخت تر است

③ open implementation برای control plane

generalized forwarding را انجام می دهند SDN در

routing

access control

load balance

... APP های کاربری

controller قرار می گیرند

Remote controller

جدا کردیم

control plane

data plane

CA

CA

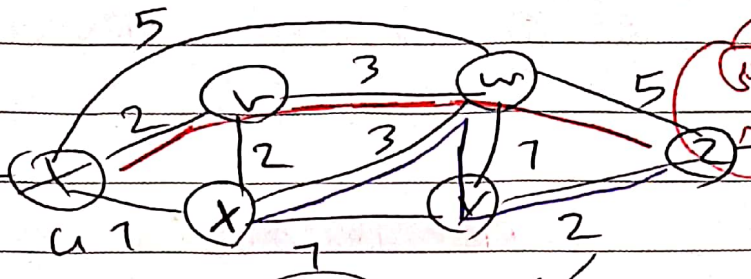
CA

CA

flow-based forwarding

قابل برنامه ریزی

مثال (Traffic Engineering) فرض کن شبکه به شکل کمان زیر است.



مشکلات پیکربندی مسیریابی در صورتی که ترافیک شبکه

① اگر مسئول شبکه بخواهد ترافیکی که از u به z است از طریق مسیر $\boxed{uvwz}$ بدهد و ترافیک بین $\boxed{x \text{ به } z}$ از طریق $\boxed{xwyz}$ بدهد

مسیریابی

برای الگوریتم های مسیریابی که کمترین هزینه را ایجاد می کنند تنها راه تغییر وزن لینک هاست تا الگوریتم ها مسیر را تغییر بدهند و یا اینکه الگوریتم متفاوتی انتخاب کنیم

② اگر بخواهیم ترافیک u به z را تقسیم کنیم به مسیرهای uvwz و uxyz باید چکار کنیم؟

Local balancing

اگر بخواهیم در واقع

با SDN می شود

با جدول های open flow می شود

یا باید الگوریتم جدید بنویسیم

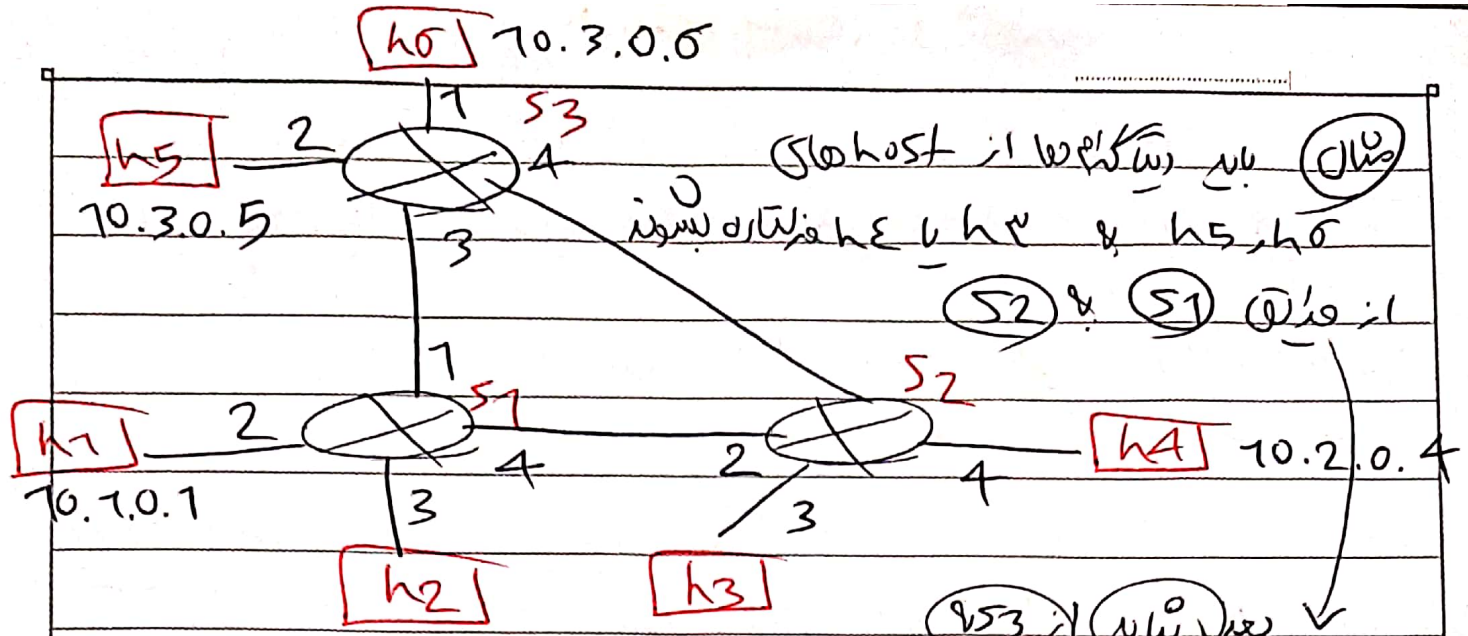
یا الگوریتم های قبلی را دستکاری می کنیم

③ برای ترافیک بین x و z در شکل x در وسط قرار گرفته با w و z در دو طرف (یعنی ۲ تا ترافیک از مسیرهای مختلف بین x و z به لینک مشترک می رسند) بازهم ارائه مسیرهای خوب باشد و به مسیر بهینه نباشد ← امکان پذیر نیست

بسته های قدرت و تنظیمی x و z مقصد z است از یک روت مشخص خارج می شوند به ازای جدول forwarding

hirmandpaper

Scanned by CamScanner



مثال: باید دیتاگرام‌ها از host های h5, h6 به h3 یا h4 فرستاده بشوند از مسیر S1 به S2

یعنی نباید از S3 به S2 بروند.

flow table ها را مسیری کانفیگ کنیم:

S3

کانفیگ S3:

match	action
IP SRC = 10.3.** IP DST = 10.2.**	forward(3)

۱ flow entry

S1

کانفیگ S1:

match	action
ingress port = 1 IP SRC = 10.3.** IP DST = 10.2.**	forward(4)

۱ flow entry

یک سمتی که از پورت ۱ اومدن

S2

کانفیگ S2 table S2:

match	action
ingress port = 2 IP DST = 10.2.0.3	forward(3)
ingress port = 2 IP DST = 10.2.0.4	forward(4)

۲ flow entry

Scanned by CamScanner

مستقل از control plane توسط داده‌ها می‌شود

data plane (switch ها)

forwarding با سخت‌افزار می‌تواند انجام شود

پروتکل برای ارتباط با controller: openflow

controller برای switch ها جابجا می‌شود و توسط controller نصب می‌شود

controller

نقش بیشتر عامل شبکه را دارد

مدیریت سخت‌افزار (پایه‌های data plane) و پایه‌های نرم‌افزاری

اطلاعاتی درباره وضعیت شبکه نگهداری می‌کند

network_control Applications

ارتباط و تعامل با applications های above با northbound API

تعامل با switch های below با southbound API

به صورت logically centralized در واقعیت به فاطر fault tolerance, performance, scalability به صورت distributed پیاده‌سازی شده‌اند (چند controller باشند که با هم ارتباط دارند)

brain

Load balancing مغز control شبکه

applications

کاربردی مثل routing, access control

این app ها با نرم‌افزار bundle نیستند

یعنی می‌توان آن‌ها را نصب کرد (از 3rd party ها)

communication layer

پایین‌ترین لایه ارتباط با switch ها

openflow, snmp

layer interface

لایه بالایی وجود interface برای ارتباط با App ها

network graph, restful API, Intent

لایه‌ای که نگهداری اطلاعات شبکه (لینک‌ها و نودها و ...)

link_state info, flow table, host info, switch info

SDN controller

stateful

state management layer

کپی از جدول flow های switch

hirmandpaper

Scanned by CamScanner

بین controller و switch چه کار کند

openflow

① تعریف format - flow table ها و generalized forwarding

② پروتکل ارتباطی controller با switch ها

③ تعریف انواع پیام های رد و بدل شده

④ از TCP استفاده می کند برای تبادل پیام ها

کنترلر داره کل شبکه رو مدیریت می کنه پس خیلی مهم است ارتباط با بالا به خاطر امنیت

کانکشن با client یا server

encryption هم می شه داشته باشیم

پیام ها ۳ تا کلاس دارند

① switch → controller

② controller → switch

③ symmetric

message ها

asynchronous

(miss)

پیام های controller to switch

① features

تبادل قابلیت های پورت ها

controller به کوئری می زند

برای گرفتن feature های switch و switch هم جواب می دهد

② configure

کنترلر کوئری می زند یا set می کند پارامترهای مورد نیاز برای کانفیگ switch

③ modify_state

state های توی flow های table را اضافه و حذف یا تغییر می دهد

④ packet_out

پیامی که توسط کنترلر encap و سپس به پورت داده می شود

به پورت گفته می شود پکت را با کدوم پورت بفرستد

Scanned by CamScanner

① packet_in : یک مدیری به پیغام میاره نمی‌دانم باید چکار کند ← می‌فرستد به کنترلر

پیام‌های switch_to_controller

② flow_removed : اگر یک entry از flow table پیغام حذف شده به controller اطلاع می‌دهد switch خودش اون entry را از جدول حذف کند ولی باید به کنترلر بگوید چون توی کنترلر یک کپی از این جدول‌ها هست و باید این جدول‌ها را تکرار کند و آپدیت کند)

③ port status : اگر تغییری توی پورت‌های switch اتفاق بیفتد ← به کنترلر با این نوع پیام اطلاع می‌دهد.

چالش‌های SDN ← سمت پوشش کنترل control plane

reliable ، dependable

secure ، distributed system ، performance ، scalable

robustness نسبت به خرابی‌ها ← اگر توی کنترلر یک مشکل ایجاد بشه چقدر خرابی دارد چون کنترلر برای شبکه مهم است

بحث امنیت

بحث تبادل بین controller ها (مدیریت خودرایی ارتباط‌ها)

Internet_scaling

session 5

توی پروتکل‌ها از الگوریتم مسیریابی برای محاسبات استفاده می‌کنه

تفاوت routing Algorithm ، routing protocol

کل مجموعه پیغام‌هایی که رد و بدل می‌شود تا الگوریتم بتونه اجرا بشه را می‌گوید

← توی الگوریتم Link state ← Link ها به ازای یک پروتکل باید رد و بدل بشن یا توی DV که هر روتر، stale DV خودش را به روتر همسایه می‌دهد ← به ازای یک پروتکل باید انجام بشه

تعریف می‌کند

اجرای دوباره این الگوریتم‌ها، تا چه فاصله‌ای زمانی باید انجام بشه؟ ← اینها را پروتکل

hirmandpaper

Scanned by CamScanner

(scalable routing) ← شبکه‌های واقعی flat نیستند

مسئله با روش scale: مسیرها و مقصدها وجود دارد ← اگر بخواهیم کل اینترنت را به عنوان یک شبکه واحد در نظر بگیریم در هر روتر ما در جدول مسیریابی ما تمامی مقصدها را ذخیره کرد.

مسئله با روش scalable نیست

اطلاعاتی که بین روترها (نودها) از link state ها باید تبادل شود

DV ها باید تبادل شوند

تعداد کل شبکه اینترنت‌ها باید broadcast شبکه = تعداد link state های هر روتر که باید ارسال شود × تعداد روترها

همه باید این را داشته باشند (هر روتر باید توپولوژی شبکه را داشته باشد)

administrative autonomy

شرکت‌ها و دولت‌ها می‌خواهند روی شبکه‌های خودشون تسلط داشته باشند و هر طوری که می‌خواهند آن را اداره کنند (خودمختاری داشته باشند)

در شبکه‌ی flat ما نمی‌توانیم این کار را کرد (نمی‌توانیم جهت‌های مختلف و مدیریت مستقل داشته باشند)

(راه حل به مسئله‌ی scalable routing)

autonomous system

AS ← اینترنت از قسمت‌های مستقل تشکیل شده ← region های به اسم domain یا AS ای که مستقل از دیگری‌ها است و به صورت داخلی اداره می‌شوند.

aka domain

intra_AS routing ← routing ما داخل یک AS بین host ها و روترها

intra_AS پروتکل ← همه‌ی روترهایی که در یک AS هستند باید مشترک و یکسان داشته باشند

ولی روترها در AS های مختلف می‌توانند پروتکل‌های intra AS متفاوتی داشته باشند

routing

inter_AS routing

gateway router ← در لبه‌های AS خودش ما

routing ما بین AS های مختلف

همه‌ی AS ها به روتر (ها) در AS های دیگر دارد.

hirmandpaper

Scanned by CamScanner

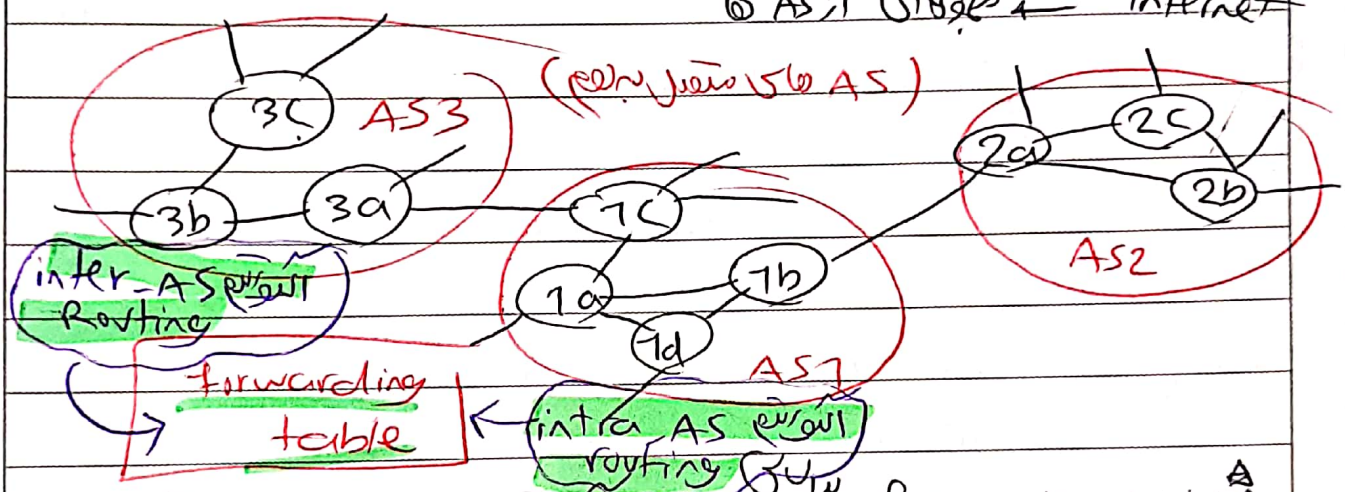
* gateway ها ← هم inter_domain routing را اجرا کنند هم intra_domain routing

به دو نوع کلی مسیریابی در AS های مختلف می‌توانند متفاوت باشد مثلاً دانشگاه مشهد از LA (link state) و دانشگاه اصفهان از (DV) استفاده کند

هر AS ← از طریق gateway به AS همسایه‌اش وصل می‌شود

inter AS ← به پروتکل استاندارد که همه AS ها باید همان را انجام بدهند

Internet ← مجموعه‌ای از AS ها



* forwarding table ← ① الگوریتم inter_AS و ② intra AS برای کانفیگ کردن

intra_AS routing ← entry هایی را مشخص می‌کند که مقصد درون AS است

inter_AS routing ← به همراه intra_AS entry هایی را مشخص می‌کند که برای مقصدهای خارجی (خارج از AS) است

* برای اتصال به همسایه‌ها ← از gateway

مقصد

فرض کن یک دوتر در AS1 به یک دیتاگرام دریافت می‌کند که خارج از AS1 است دوتر باید بگوید به یک دوتر gateway بفرستد و کدوم gateway ← inter_AS routing

① AS1 باید یاد بگیرد کدوم مقصدها از طریق AS2 و کدوم‌ها از طریق AS3 قابل دسترسی هستند. ② این اطلاعات را که به دست آورد به بقیه دوترها در AS1 بفرستد

hirmandpaper

Scanned by CamScanner

* gateway در AS ، اطلاعات پروتوکل (inter_AS) را که به لیست مسیریابی

آورده ، به همه روترهای داخلی propagate می‌کند و باید به لیست بیاید تا

مسیرها برای رسیدن به اون gateway بعدی در بین روترهای داخلی

صدا از (9.a) می‌خاد به (1b) برد که gateway الت ← از کدوم

مسیر برد که بعد باید بیاد

این کار پروتوکل intra_AS الت

دستگاه مقصدهای AS های دیگر ← پروتوکل inter_AS

gateway مربوط به اون مقصد ← intra_AS

(IGP) interior gateway protocol ←

Routing در داخل AS ها ← Intra_AS Routing

پروتوکل‌های معروف: ① RIP (Routing information protocol)

② OSPF (open shortest path first) ← تقریباً همه جا این استفاده می‌شه

③ IGRP ← interior Gateway Routing protocol

open source الت

IS_IS پروتوکل ← Link state الگوریتم ← OSPF پروتوکل

مسیر در الگوریتم‌های Link_state

از Dijkstra استفاده می‌کند

topology map در هر نود به لیست میاد

اونه ، link state ospf را به همه روترها در کل AS ،

تبلیغ می‌کند flood می‌کند

دیگر توی دیتای Application ، نیستند

ospf ، در خود دیتا کپسوله encapsulate می‌شوند IP ، دستگاه‌های توی

ویژگی‌های advanced ← security ، همه پیام‌ها message ، ospf

authenticate شده‌اند

multiple-same-cost paths

توی جدول مسیریابی

به جای یک پورت ، چند تا پورت صدا داشته باشیم

چند تا از بهترین مسیرهای که cost نزدیک به هم دارند را به همه

Load balancing را میسر می‌کند

hirmandpaper

Scanned by CamScanner

③ برای هر لینک، معیارهای متفاوتی به عنوان cost هاش می‌گذاریم به اسمش

TOS (type of service)

multiple cost metric

مثلاً لینک‌های satellite ← ترافیک حساس به تاخیر، هزینه زیاد

ترافیک‌های best effort ← هزینه کمتر (چون تاخیر برایش مهم نیست)

④ مهم‌ترین دیگری ← مسیریابی به صورت سلسله مراتبی دارد

hierarchical OSPF در داخل‌های بزرگ

تعداد نودها بالاست و حجم ترافیکی که AS ها اگر خیلی بزرگ باشند باید بین نودها رد و بدل بشه و جدول‌ها و link state ها ← خیلی بار می‌آرد

می‌آیم AS خودمان را به ناحیه‌های کوچکتر تقسیم کنیم و لینک‌هایی که باید الگوریتم مسیریابی مثل Dijkstra توش اجرا بشه را کوچکتر کنیم

تقسیم لینک‌ها (یک AS) به ۲ راه

① area of backbone

② بقیه لینک‌ها (area های دیگر) local area

hierarchical OSPF

الگوریتم Dijkstra توی هر area مستقلاً اجرا می‌شه

link state advertisment فقط در هر area انجام می‌شود ← overhead ترافیکی کمی ایجاد کند

مثال: مدام در هر area ← فقط اطلاعات گراف اون area را دارد

مسیرهای مربوط به مقصدهای توی area را می‌توانند مشخص کنند

سوال: جابجایی از یک subnet در یک area به subnet دیگری در یک area دیگر داشته باشیم؟

boundary router ← روترهایی که کل AS را با AS های دیگر وصل می‌کنند

ASBR

backbone router

backbone های area border routers

① روترهایی که داخل ناحیه‌های backbone هستند

② روترهایی که واسطه هستند با area های دیگر (هم خودشون area دارند هم area های دیگر)

Scanned by CamScanner

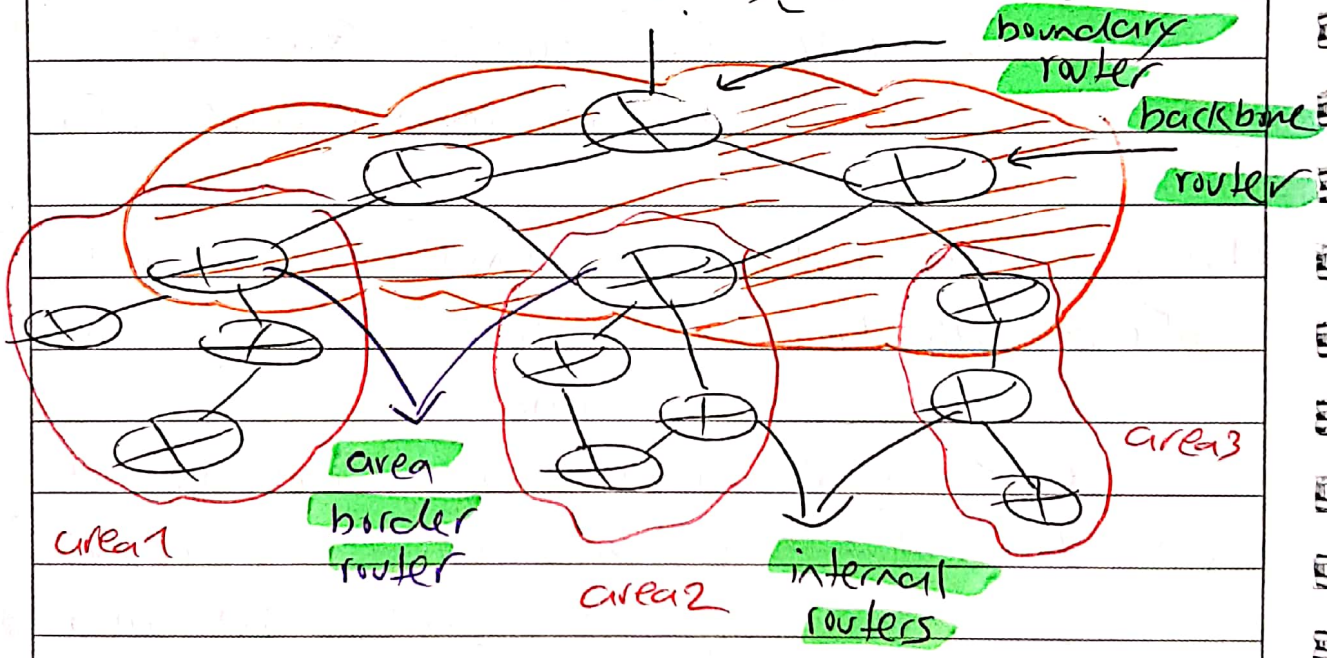
بقیه روترها در area های دیگر (به جز area 0 که backbone است)

به اسمش internal router می‌گیم

* وظیفه‌ی پیدا کردن مسیر از subnet های درون area به subnet در area های دیگر ← به عهده‌ی border router ها است

آلگوریتم Dijkstra در داخل area 0 هم بخشی از area 0 و backbone هم است

اجرا می‌کند ← روتر border در area 0 مدام مسیرهای داخل area 1 را دارد ← میاد اطلاعاتی که دارد را به border router ها در area های دیگر منتقل می‌کنه. اونا هم این اطلاعات را به internal router ها نشون می‌دهند ← مسیرهای اطلاعات در subnet های area های دیگر را بلد میاره



backbone router ← اجرای مسیریابی OSPF در backbone

boundary router ← به AS های دیگر وصل می‌شود

می‌تونیم بگیم که یک area می‌تونه چند تا روتر gateway برای area 0 داشته باشه در جدول مسیریابی روترهای بالاتر ← می‌تونند table های خیلی کوچیک شوند

یا در جدول‌های مسیریابی در داخل یک area 0 مسیرهای خارج از area می‌تونند default gateway داشته باشند ← اطلاعات مسیریابی بین داخل و خارج area

hirmandpaper

Scanned by CamScanner

Session 6

* هدف از پروتکل routing در نهایت چی هست؟
در داخل شبکه ← پروتکل IP اجرا می‌شود و بسته‌ها در قالب IP منتقل می‌شوند
که توی بسته‌های IP یک آدرس مقصد وجود دارد. ← پروتکل IP یک وظیفه دارد

بسته در داخل شبکه یک آدرس دارد و آدرس مقصد را نگاه می‌کند و روتر چون جدول مسیریابی را دارد که توش لیست subnet های دیگر را در جدول دارد و مشخص می‌کند برای هر subnet یک پورت خروجی مناسب است.
خیلی‌ها خودت جدول در هر روتر بدست آمده و به دیدنی الگوریتم‌های routing

پس برای یک subnet مقصد ← باید توی جدول یک entry در جدول داشته باشیم
آدرس مقصدی که از هدر آیپی خوانده می‌شود با entry در جدول چک می‌شود
و اگر match شد، به پورت خروجی مناسب ارسال می‌شود

برای subnet های داخل AS، الگوریتم routing اجرا می‌شود و در جدول اطلاعات را می‌نویسد مثلاً OSPF پروتکل، الگوریتم dijkstra را اجرا می‌کند

سوال: اگر subnet در یک AS دیگر باشد باید چکار کنیم؟
باید پروتکل مسیریابی مناسب انتخاب کنیم و مسیر مناسب را برای بسته‌ها پیدا کنیم و وارد جدول روترها بکنیم
آیا لازم است برای تک تک subnet ها توی شبکه، اطلاعات را توی جدول بنویسیم؟
خیلی ∞ و Border Gateway protocol
BGP ← پروتکل routing که بین AS ها اجرا می‌شود (inter-AS routing)
استانداردی است که توسط AS ها اجرا می‌شود.
در این روش subnet ها خودشون رو به بقیه شبکه advertise می‌کنند (اعلام وجود می‌کنند) → I am here
یک AS، مسیر مناسب برای subnet هایی که داخلش هستند را به همسایه‌هاش اعلام می‌دهد ← همسایه‌ها، اطلاعات دستیابی به این subnet ها را بدست میارن

hirmandpaper

Scanned by CamScanner

اطلاعات دسترسی subnet ها را از همسایه‌ها دریافت می‌کند و با بقیه همسایه‌ها

AS های دیگر

external eBGP

BGP

iBGP ← وقتی اطلاعات توسط روتر مرزی یک AS دریافت شد

به بقیه iBGP ها اطلاعات به روترهای داخلی AS منتقل می‌شود

وقتی اطلاعات به روتر لبه entry خودش را به بقیه می‌فرستد که دارد

پردازش می‌کند و جدول مسیریابی را بروز می‌کند.

Gateway router ← ارتباط eBGP و iBGP

ارتباط iBGP

Route establishment در BGP

اگر AS1 و AS2 بخواهند با هم communicate

کنند ← ① AS1 باید یک route به AS2 را اعلام کند

② AS2 باید route ای که از AS1 اومده را قبول کند

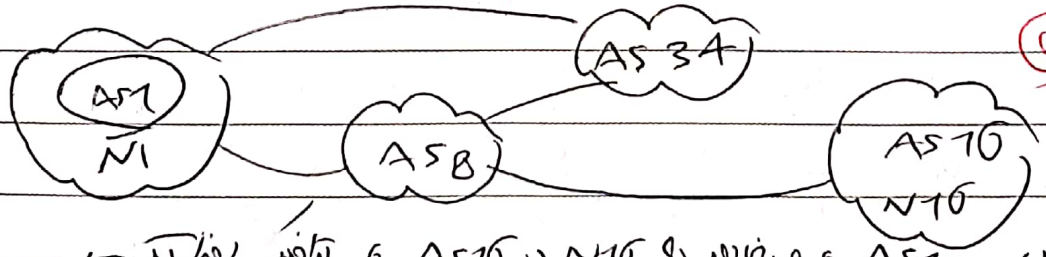
③ AS2 باید یک route به AS1 را اعلام کند

④ AS1 باید route ای که از AS2 اومده را قبول کند

تا ارتباط ۲ مرحله‌ای باشد

انجام می‌شود handshake

مثال



N1 در AS1 می‌خواهد به N16 در AS16 ترافیک بفرستد ←

① AS16 باید اطلاعات دسترسی N16 را به AS8 (همسایه‌اش) اطلاع بدهد

② AS8 باید مسیر اعلام شده را از AS16 قبول کند

③ AS8 باید N16 را به AS1 یا AS34 اعلام کند

hirmandpaper

Scanned by CamScanner

AS7 باید N16 را از AS8 یا AS34 بپذیرد

* نکته ← اگر flow بخواهد دوطرفه باشد (یعنی N16 هم بخواهد به N7 بسته بفرستد) باید N7 هم همین مراحل را طی کند

semi-permanent

peer ها

← ۲ تا روتر BGP، پیام‌های BGP را روی یک کانکشن دائمی TCP مبادله می‌کنند

BGP session

← بین ۲ تا روتر یک session برقرار می‌شود

← توی این session، به مقصدهای مختلف، path هایشون را advertise می‌کنند

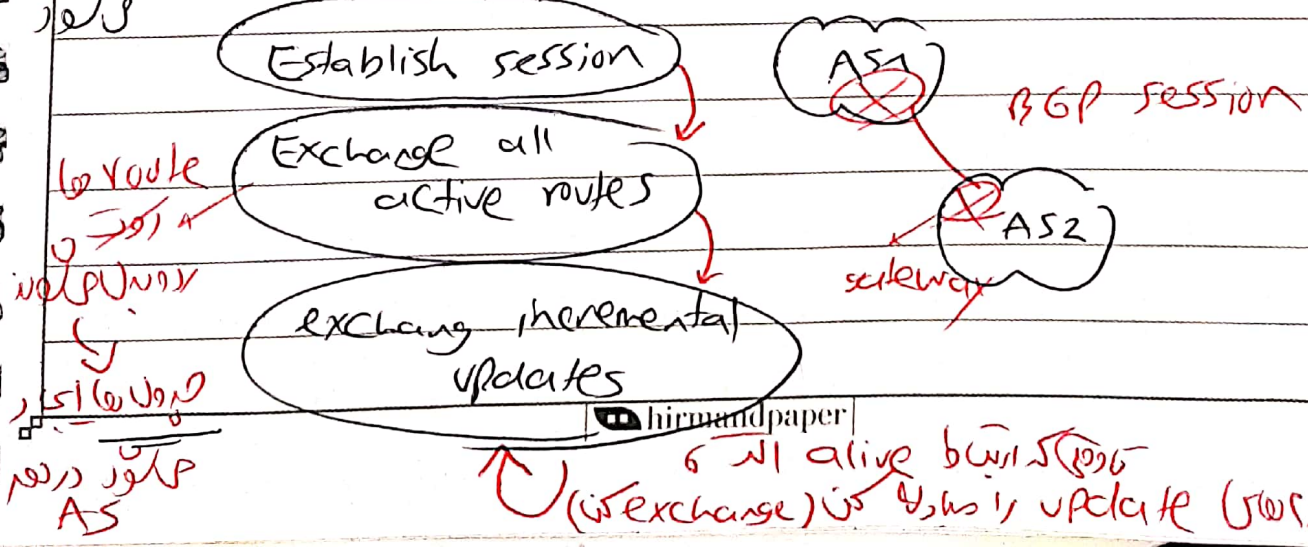
BGP یک پروتکل path vector است.

BGP advertisment

AS3, X

وقتی روتر 3a (که روتر gateway AS3 است)، مسیر AS3, X را به AS2 advertise می‌کند ← در واقع AS3 به روتر 2c که gateway AS2 است، قول می‌دهد که تمام بسته‌ها را از طریق X forward می‌کند

یعنی اگر AS2، بسته‌ای به مقصد X به AS3 بفرستد، به X روانه می‌شود



hirmandpaper

Scanned by CamScanner

① open ← ایجاد TCP بین ۲ تا روتر مرزی
Connection

② update ← مسیرهای جدید را advertise می‌کند
اعلام حذف مسیر

③ keepalive ← کانکشن را alive نگه می‌دارد با وجود اینکه update ای اتفاق نمی‌افتد
پیام‌ها از طریق کانکشن TCP بین دو تا peer رد و بدل می‌شوند

پیام‌های BGP
برای تایید کردن (ACK کردن) پیام‌های open التقاط‌شده

④ notification ← گزارش ارورهای موجود
بستن Connection

prefix + attributes = "routes" ← BGP Routes
آدرس subnet ای که مسیر دارد برای تبلیغ می‌کند
اطلاعاتی که در کنار prefix قرار می‌گیرند تا مسیر مناسب پیدا کنند

۲ تا attribute مهم

① AS_path ← لیست AS های که prefix advertisment ۶ از آن عبور کرده (AS های که از مسیر آن مقصد دیده می‌شود)

② next_hop
آدرس IP روتر مرزی اولین AS در مسیر
این مسیرها و توی message های اند که رد و بدل می‌شوند.

AS ای که دارد مسیر را به همسایه‌ها می‌فرستد

BGP message

مثال: prefix 6 138.16.64/22
AS_path 6 AS3 AS131 ;
next_hop 6 207.44.73.125

اول AS131 را دیده بعد AS3

آدرس آی‌پی روتر 3a

این آدرس توی AS131 قرار دارد و مسیری را تبلیغ می‌کند

AS3 روتر مرزی

اولین روتری که توی AS3 دیده می‌شود

hirmandpaper

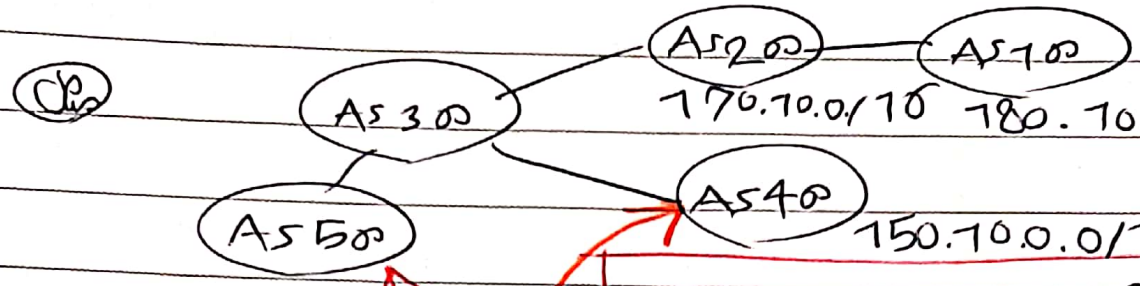
Scanned by CamScanner

دنباله‌ای از AS ها که مسیرهای از نقطه عبور کرده

AS path

هر AS ای که advertisment را دریافت می‌کند و قبول می‌کند وقتی می‌خواهد به همسایه‌ای بفرستد، AS خودش را هم به AS path اضافه می‌کند

مثال



180.10.0.0/16 300 200 100
170.10.0.0/16 300 200
150.10.0.0/16 300 400

180.10.0.0/16 300 200 100
170.10.0.0/16 300 200

مسیری که برای این subnet که AS100 است تبلیغ کرده

صرفاً این یک subnet ای توی AS100 است که مسیر رسیدن می‌ماند به AS400 و اینکه از AS100 برمی‌ماند به AS200 بعد به AS300

300 ← 200 ← 100

آدرس 150.10.0.0/16 یک subnet ای است که توی AS400 وجود دارد و اگر AS500 بخواهد بفرستد باید مسیر AS400 به AS300 را برود.

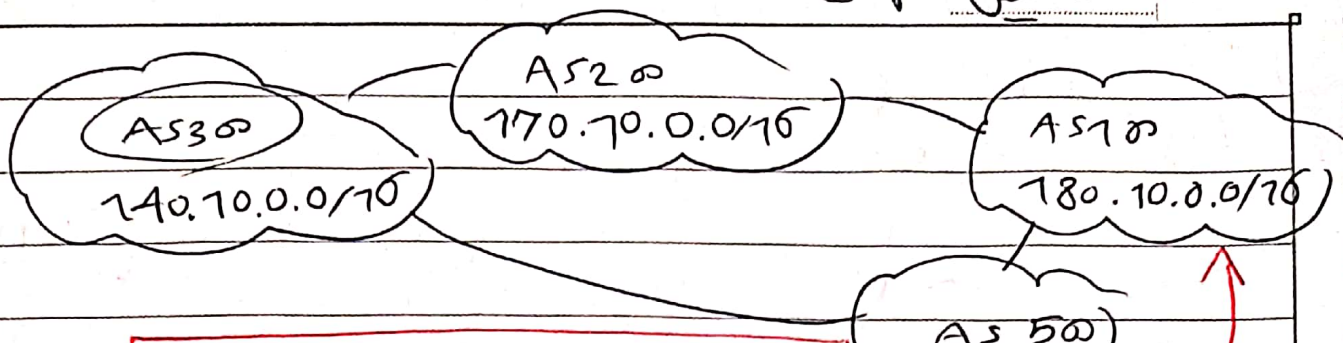
حال، چرا AS هایی که در مسیر دیده می‌شوند هم توی مسیر قرار می‌گذاریم؟

Apply policy ② Loop detection ①

وقتی برای یک AS یک advertisment می‌رسد و می‌خواهد قبول کند، بداند بسته‌هایی که از طریق این مسیر دریافت می‌شوند از چه AS هایی عبور می‌کنند ← برای اعمال سیاست‌ها که بسته‌ها از کدام AS ها عبور کنند جلوگیری از رخ دادن Loop در مسیر

Scanned by CamScanner

مثال: تشخیص loop



180.10.0.0/16	300	200 100
170.10.0.0/16	300	200
140.10.0.0/16	300	

140.10.0.0/16	500	300
170.10.0.0/16	500	300 200

AS500 ، این مسیرها را با همسایه‌هایش می‌فرستد ← AS100 ، مسیرهایی که برای AS200 و AS300 هست را قبول می‌کند و تبلیغ می‌کند

ولی مسیری که برای خودش بوده را قبول نمی‌کند چون خودش (AS100) را توی مسیر می‌بیند ← رد می‌کند

نوع از رالت یا حب

Next hop ← با قرار دادن next hop در مسیر، به همسایه می‌گیم اگر بخواهید بسته‌ای را در این مسیر بفرستید باید آن را به این آدرس بفرستید.

هر کی next hop خودش را توی مسیر می‌گذارد ← هر AS به یک next hop ، مسیر (path) به AS قبلی خودش ، set up می‌کند

next hop ، قدم بعدی را برای ارسال بسته نشون می‌دهد

قدم به قدم برای انتقال بسته در مسیر، انتخاب می‌شوند

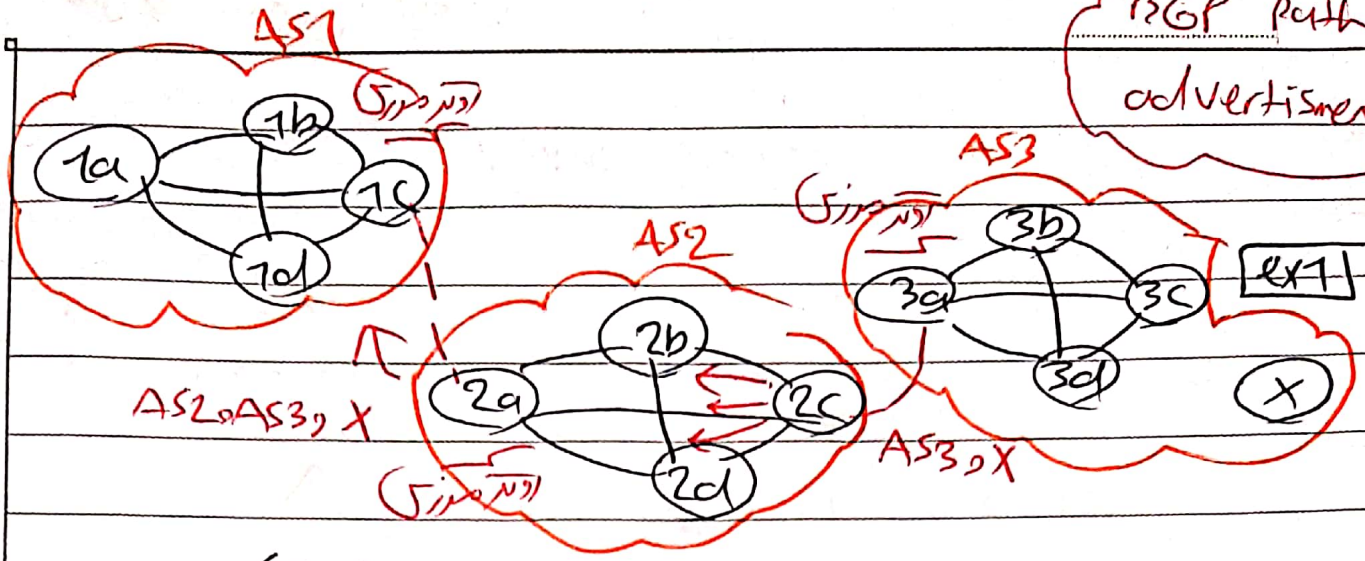
BGP دیگر نیازی ندارد که وارد توپولوژی فیزیکی داخلی AS ها بشه و مسیر داخلی هر AS را با IGP می‌دانند

در هر AS ، IGP ، مدیریتی مقصدی که next hop است ، مسیر داخلی خودش را برای اینکه بسته‌ها به این مسیر بفرستند تبلیغ می‌کند

hirmandpaper

Scanned by CamScanner

BGP path advertisement



ex1

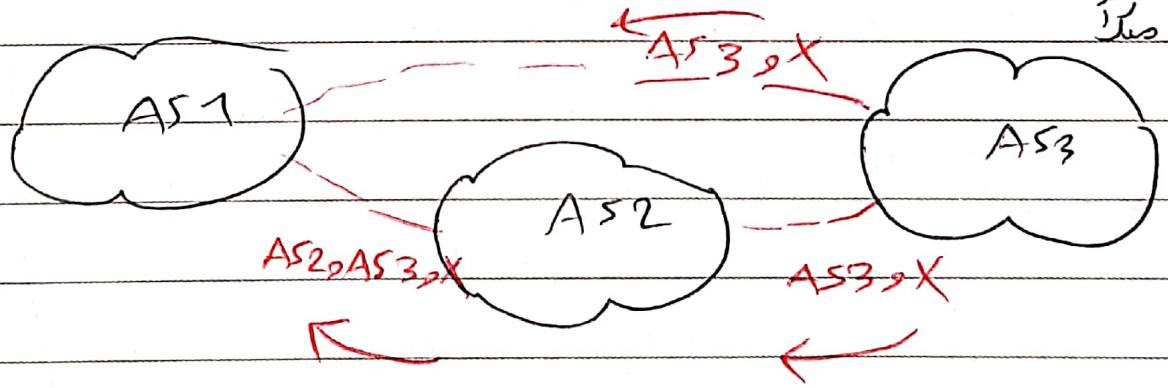
eBGP

① روتر مرزی 2c در AS2 مسیر X و AS3 را با eBGP از AS3 (روتر 3a) می‌گیرد

② بر اساس policy ای که AS2 دارد روتر 2c در AS2 مسیر X و AS3 را قبول می‌کند و با iBGP به همه‌ی روتر‌ها در AS2 منتشر می‌کند

③ بر اساس سیاست AS2 روتر 2a به کمک eBGP مسیر X و AS3 و AS2 را به روتر 1c در AS1 می‌فرستد

ex2 روتر gateway ها می‌توانند مسیرهای مختلفی برای مقصد دریافت کنند



پس تا مسیر درست برای advertise کردن X به AS1

① روتر 1c در AS1 مسیر X و AS3 و AS2 را یاد گرفت

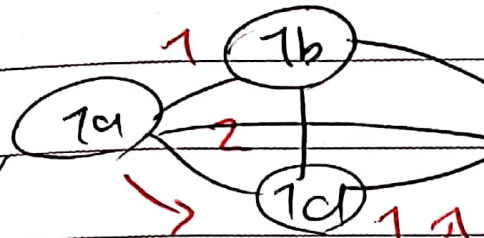
② روتر ~~~ و ~~~ AS3 و X ~~~

بر اساس policy ← روتر 1c مسیر AS3 و X را انتخاب می‌کند و مسیر را در AS1 به کمک iBGP advertise می‌کند

Scanned by CamScanner

اطلاعات مسیر برای یک subnet که در یک AS دیگری وجود دارد در روتر.

AS شماره همسایگی set می‌کند



روترهای داخلی یک AS چگونه entry مربوط به subnet X که در AS2 بود را پر می‌کنند؟ در جدول مسیریابی‌شان

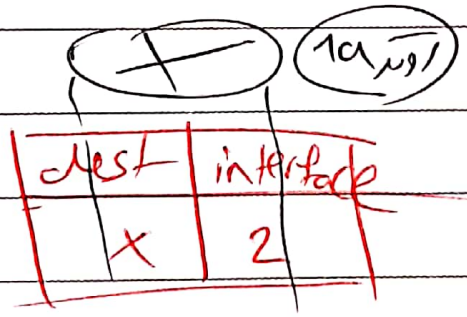
روتر 1d می‌داند که مسیر مربوط به X از طریق 1c است ← به الگوریتم routing داخلی

روتر مرزی در AS1 به دست می‌آورد که کدام مسیر برای رسیدن از 1d به 1c مناسب‌تر است؟

dest	interface
X	1

به الگوریتم پروتکل intra-domain مثلاً اگر OSPF باشد، به دست می‌آید

روتر 1a به الگوریتم OSPF و برای مقصد 1c بهترین مسیر تشخیص داده می‌شود ← پورت 2 و



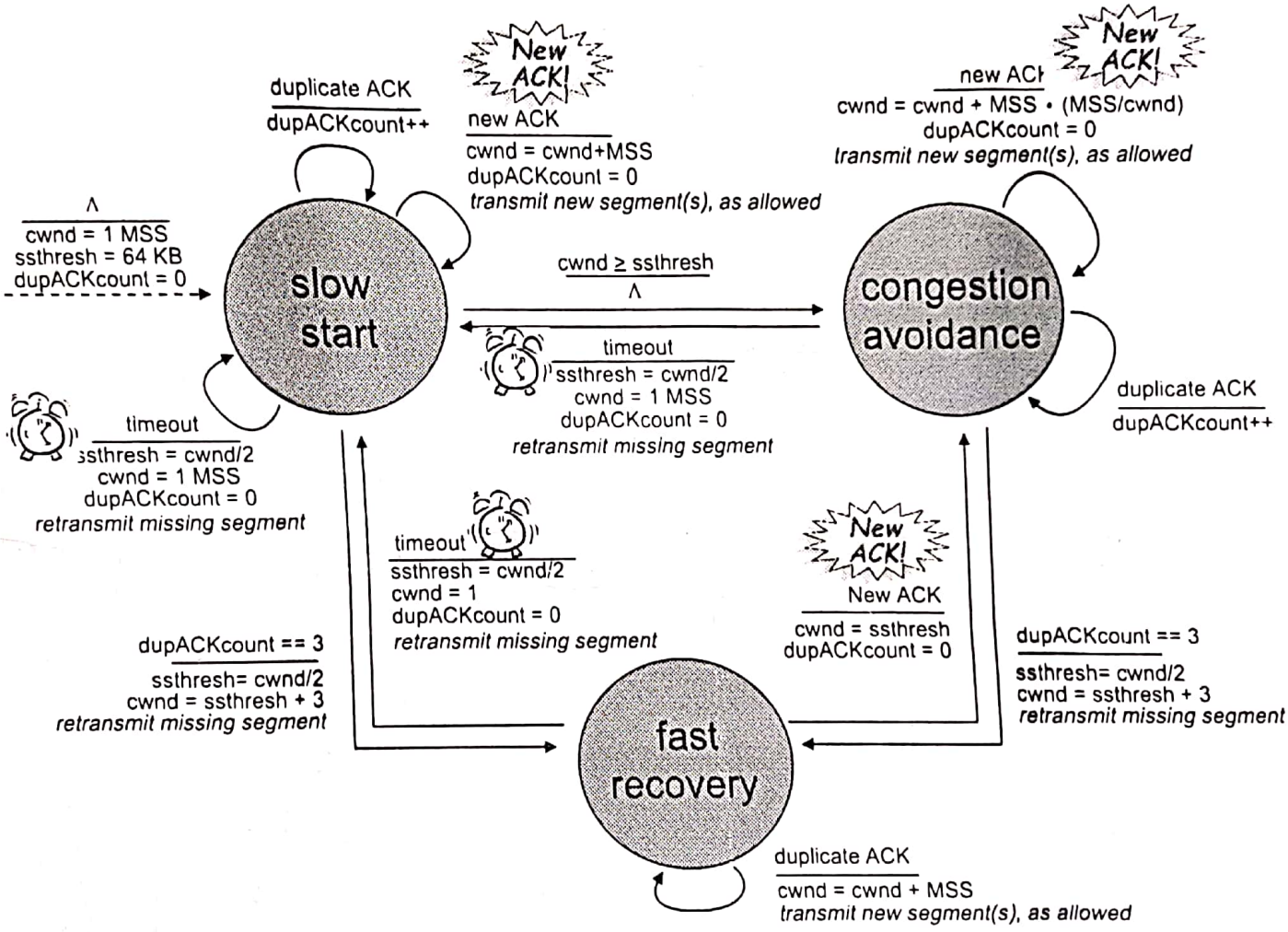
dest	interface
X	2

برای پر کردن اطلاعات جدول مسیریابی یک روتر در یک AS ← برای subnet هایی که در AS های دیگر قرار دارند ← هم از BGP استفاده می‌کنیم هم پروتکل مسیریابی مثل OSPF که داخل AS اجرا می‌شود

این ۲ تا پروتکل و اطلاعات مسیرها را مشخص می‌کنند ← entry مناسب در جدول ایجاد می‌کنند.

Scanned by CamScanner

Summary: TCP Congestion Control



Scanned by CamScanner