Chapter 4
Network Layer : Data Plane

- توی این فصل ، به این که Data Plane و Control Plane به کدوم کار ها اطلاق میشن، می پردازیم.
- توی این فصل ، تمرکز روی Data Plane هست و :

1 - اصولی در رابطه با مدلی هایی که لایه ی شبکه به لایه های بالاترش سرویس میده، گفته میشه.

2 - تفاوت forwarding و routing گفته میشه.

3 - ساختار داخلی روتر رو بررسی می کنیم.

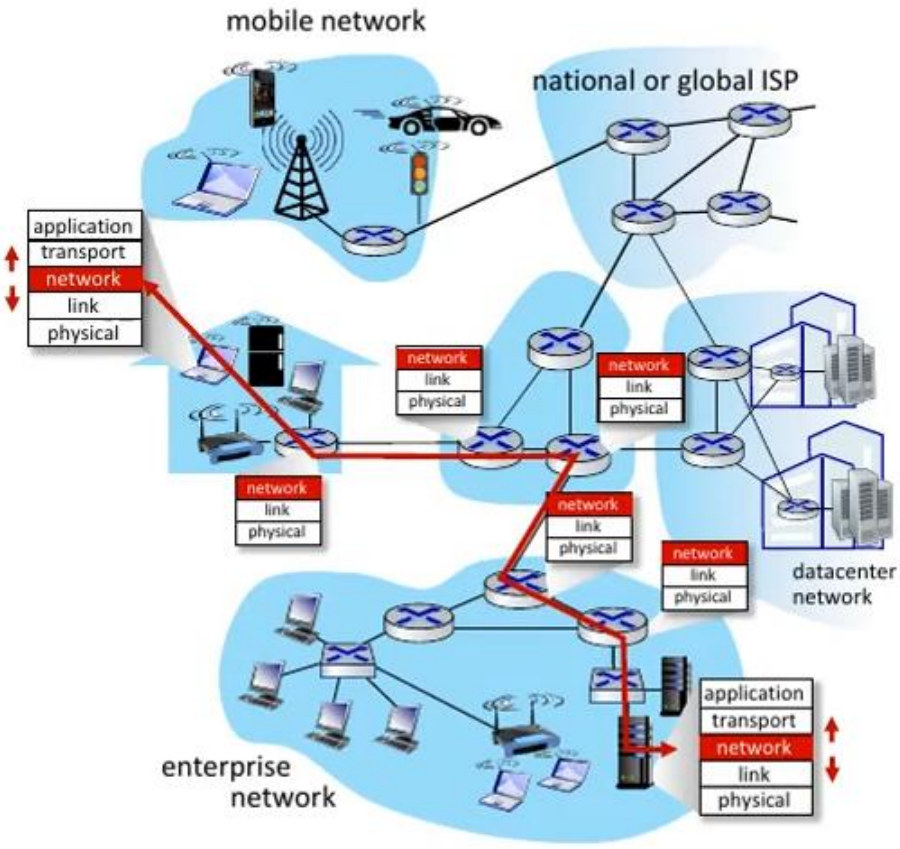
4 - نحوه ی آدرس دهی device های مختلفی که به شبکه متصل میشن رو بررسی می کنیم.(به نحوی که آدرس یکتا داشته باشن و و در عین حال پیچیدگی زیادی برای کارهایی که باید در روتر ها انجام بشه ، ایجاد نکنن.)

5 - با generalized forwarding و Internet architecture آشنا میشیم.

6 - با پروتکل IP که معروف ترین پروتکل لایه ی شبکه هست آشنا می شیم. همچنین با بحث NAT (Network Address

Translation) که به منظور برطرف کردن یه مشکل عملی(که کمبود آدرس های آی پی نسخه ی چهارم هست) می پردازیم.

- برای این که ارتباط لایه ی شبکه با لایه های حمل و نقل و اپلیکیشن روشن بشه ، شکل روبرو رو در نظر می گیریم :



توی این شبکه فرض شده که process های لایه ی اپلیکیشن توی دو تا end system می خوان ارتباط داشته باشن ، در این صورت لایه ی شبکه ی سمت فرستنده ، سگمنت ها رو از لایه ی transport دریافت

می کنه و هدر های مخصوص این لایه ی رو به سگمنت اضافه می کنه و datagram حاصل رو با توجه به آدرس مقصد برای اولین روتری که در مسیر بین فرستنده و گیرنده هست ، ارسال می کنه.

در سمت گیرنده ، عکس این کار انجام میشه ؛ یعنی این که پس از دریافت datagram با توجه به اطلاعات هدر، قسمت سگمنت extract میشه و تحویل پروتکل لایه ی حمل و نقل (TCP یا UDP) داده میشه.

در کنار فرستنده و گیرنده ، پروتکل های لایه ی شبکه در همه ی روتر هایی که توی شبکه هستن، وجود دارن. روتر ها هدر های datagram هایی که دریافت می کنن رو بررسی می کنن و اون ها رو (با توجه به جدول forwarding) از پورت های ورودی به پورت های خروجی منتقل می کنن که به این عمل forwarding میگن.

در حالت کلی کارهای دیگه ای هم علاوه بر forwarding ممکنه انجام بشه. مثلا یه بسته ای ممکنه duplicate بشه و برای چندتا پورت خروجی ارسال بشه که به این عمل multiplexing میگن. یا ممکنه یه بسته block بشه یا دور ریخته بشه . (این کار توی دستگاه هایی که علاوه بر routing ، نقش filtering دارن می تونه انجام بشه.)

- لایه ی شبکه دوتا کار اساسی انجام میده :

1 - Forwarding : یعنی بسته هایی که از طریق یه پورت ورودی به یک روتر می رسن ، به یک پورت خروجی مناسب داخل روتر منتقل بشن. منظور از مناسب اینه که انتقال از یه پورت ورودی به پورت خروجی در راستای اون مسیری باشه که نهایتا بسته باید از فرستنده تا گیرنده طی کنه.

2 - Routing : اون مسیری رو مشخص می کنه که بسته ها باید از فرستنده تا گیرنده طی کنن. این مسیر نتیجه ی اجرای الگوریتم های routing هست.

به عنوان مثال ،این کارها شبیه کارهاییه که برای مسافرت از یه شهر به یه شهر دیگه انجام میشه . این که از یک چهارراه به یه چهارراه دیگه می رسیم یا از یک خیابان در یه چهارراه به یه خیابان دیگه در چهارراه دیگه ای میریم ، مثل کار forwarding عه . اون نگاهی که به نقشه می کنیم و مسیرمون رو قبل از حرکت انتخاب می کنیم هم مثل کار routing عه .

- کارهایی که در لایه ی شبکه انجام میشه رو به دو دسته ی data plane و control plane تقسیم می کنیم.

Data plane : اون کارهایی هستن که به ازای دریافت یه datagram به صورت محلی(local) در داخل هر روتر انجام میشه. مهم ترین کار توی این دسته ، همون forwarding هست که وقتی یه

datagram روی یه پورت دریافت میشه ، باید به پورت مناسب خروجی در داخل همون روتر منتقل بشه. (کارهای دیگه هم مثل block کردن یا duplicate کردن هم انجام میشه.)

Control plane :

یه سری کارها هستن که با توجه به شرایط کلی شبکه باید در موردشون تصمیم گیری بشه. شاخص ترینش ، routing عه.

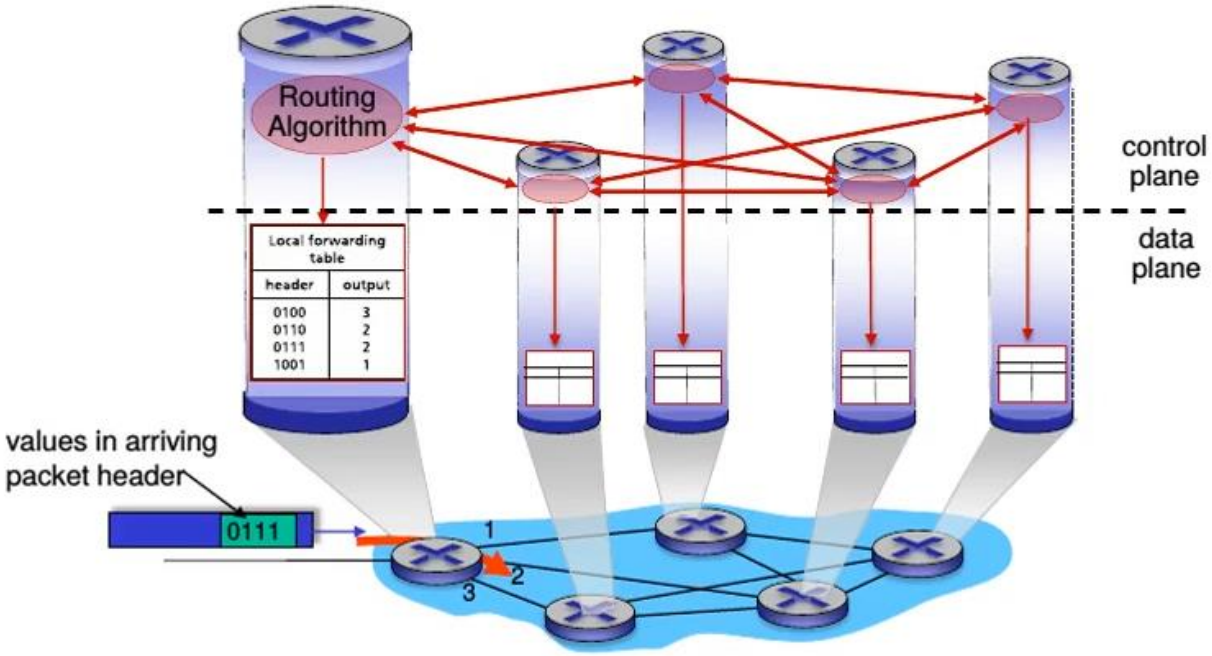
کار دیگه می تونه سیاست های filtering باشه.

به طور کلی کارهای مربوط به Control plane رو به دو روش می تونیم انجام بدیم :

1 - Traditional routing algorithms : روش غالب هست و خیلی جاها ازش استفاده می کنیم. داخل روتر ها توزیع شدن و روتر ها با استفاده از پیام های کنترلی مسیریابی بین فرستنده و گیرنده رو انجام میدن.

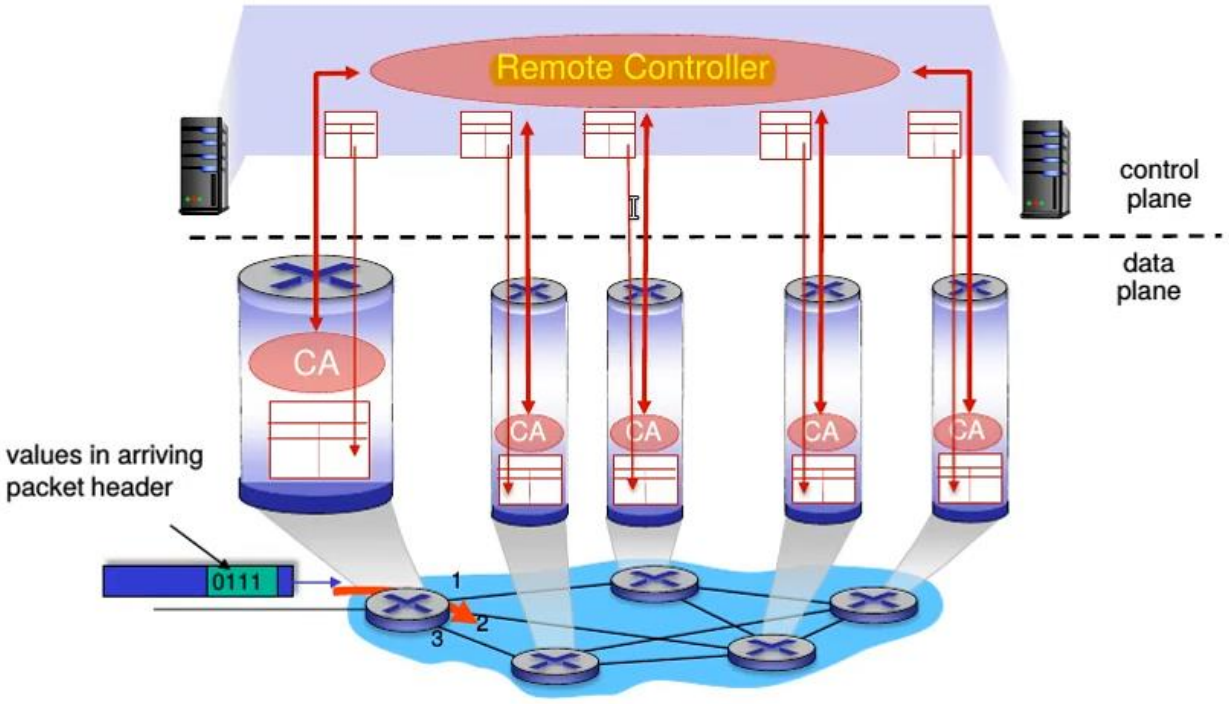
2 - Software-defined networking(SDN) : توی این روش ، کارهای کنترلی که توی روش اول انجام می شد، انجام نمیشه . توی یه سری سرور انجام میشه و بعد نتیجه در داخل روتر ها config میشه. مثلا مقادیر forwarding table هایی که روتر ها برای انجام کارهای data plane بهش نیاز دارن، با استفاده از الگوریتم هایی که داخل سرور ها اجرا میشن، تعیین میشه.

شکل زیر رویکرد روش سنتی(روش اول) رو نشون میده :



ما از بین کارهایی که توی control plane انجام میشن، به طور خاص ، روی الگوریتم های routing تمرکز می کنیم. الگوریتم های routing توی روش سنتی، در داخل همه ی روتر ها وجود دارن . این الگوریتم های توزیع شده از طریق رد و بدل پیام های کنترلی می تونن مسیر های بهینه بین هر روتر تا هر گیرنده ی داخل شبکه رو تعیین کنن و با استفاده از این اطلاعات مقادیر جدول forwarding رو تعیین می کنن.

شکل زیر رویکرد روش SDN رو نشون میده :



در رویکرد SDN ، الگوریتم های routing در داخل یک controller خارجی که به صورت فیزیکی از روتر ها جدا هست، تعیین میشه . نتیجه ی اجرای الگوریتم ها در جدول forwarding ذخیره میشه که بین روتر ها توزیع شده. همونطور که توی شکل هم هست، به جای این که توی هر روتری ، یه الگوریتم routing وجود داشته باشه ، ماژولی به اسم CA(Control Agent) هست که با یه پروتکل مخصوص با Remote Controller در ارتباطه و Remote Controller هم از طریق این ارتباط، جداول forwarding روتر های مختلف رو مدیریت و مقداردهی می کنه.

پس توی این مدل ، روتر ها با هم ارتباط مستقیم کنترلی ندارن و برای هم پیام کنترلی نمی فرستن و ارتباطشون از طریق Remote Controller هست.

- Network service model

- در مورد یک datagram سرویسی که لایه ی شبکه می تونه در اختیار قرار بده : ۱- ضمانت تحویل اون به گیرنده است(guaranteed delivery) ۲- میزان تاخیری که یک datagram متحمل میشه رو تضمین می کنه که از یه حدی بیشتر نشه.
- در مورد یک flow از datagram ها ، سرویس هایی که لایه ی شبکه می تونه بده اینه که ۱- ترتیب datagram ها بهم نمی ریزه ، ۲- یه پهنای باند حداقلی به اون flow تخصیص داده میشه. ۳- تاخیر بین datagram ها از یه حدی بیشتر نمیشه. (delay jitter)
- مهم ترین شبکه ای که توی اینترنت هست، از پروتکل IP استفاده می کنه. خود پروتکل IP به صورت اولیه در مورد هیچ کدوم از سرویس ها ضمانتی نمیده ؛ برای همین میگن مدل پروتکل IP ، best effort هست. (یعنی حداکثر تلاشش رو انجام میده ولی ضمانتی در کار نیست)
- در کنار پروتکل IP ، در شبکه های ATM لایه ی شبکه می تونه راجع به برخی از سرویس ها ضمانت بده ؛

به طور کلی دو مدل سرویس در شبکه های ATM وجود داره :

1 - Constant Bit Rate : در مورد همه ی پارامتر های Bandwidth ، Loss ، Order و Timing ضمانت میده.

2 - Available Bit Rate : در دو موردِ Minimum Bandwidth و Order ضمانت میده. اما در مورد Loss و Timing ضمانتی نمیده.

- دوتا پروتکل دیگه هم هست که یه جورایی extend شده ی پروتکل IP هستن ، یکی Intserv(Integrated Service) توی RFC 1633 ارائه شده و Diffserv(Differentiated Service) که توی RFC 2475 ارائه شده .

Intserv برای همه ی پارامتر ها میتونه گارانتی بده ولی Diffserv بعضی از پارامتر ها رو ممکنه که بتونه گارانتی کنه.

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

در مورد شبکه های real time که میخوان سرویس با کیفیت خوب ارائه بدن ، یکی از راه ها ، استفاده از این دوتا پروتکل هست.

- پروتکل IP که در اینترنت استفاده میشه مدلش best effort هست . در نگاه اول ممکنه ناامید کننده به نظر برسه ، ولی حقیقت اینه که پروتکل IP گسترش خیلی زیادی پیدا کرده . چرا؟

1 - پیاده سازی ساده ای داره ؛ که در این صورت استفاده از اون هم دامنه ی وسیع تری پیدا می کنه.

2 - پهنای باند بالای لینک هایی که توی core اینترنت مخصوصا توی ISP های tier1 و tier2 استفاده میشه و این، فشار رو از روی شبکه رو کم می کنه. چون حتی وقتی مدل best effort هست، باعث میشه عملکرد اپلیکیشن های real time به اندازه ی کافی خوب باشه.

3 - روش هایی که برای اپلیکیشن های real time استفاده میشه؛ مثال استفاده از replicated server ها یا CDN که باعث میشن سرور ها به کلاینت ها خیلی نزدیک باشن و ما بتونیم از چندین لوکیشن سرویس رو دریافت کنیم.

4 - Congestion control : اون سرویس هایی که elastic هستن و خیلی محدودیت های زمانی ندارن وقتی مکانیزم Congestion control اجرا بشه ، با توجه به این که ترافیک شبکه بالا هست،

rate ارسالشون پایین میاد؛ در نتیجه جا برای اپلیکیشن های real time باز میشه تا اون ها بتونن به معیار های زمانی خودشون دسترسی پیدا کنن.

هر چند همه ی این عوامل برای جبران best effort بودن لایه ی IP هستن ، ولی پاسخ به این سوال که آیا مدل best effort مدل خوبی برای اینترنت هست یا نه ، چالش بر انگیزه.