Chapter 4
Network Layer: Data Plane

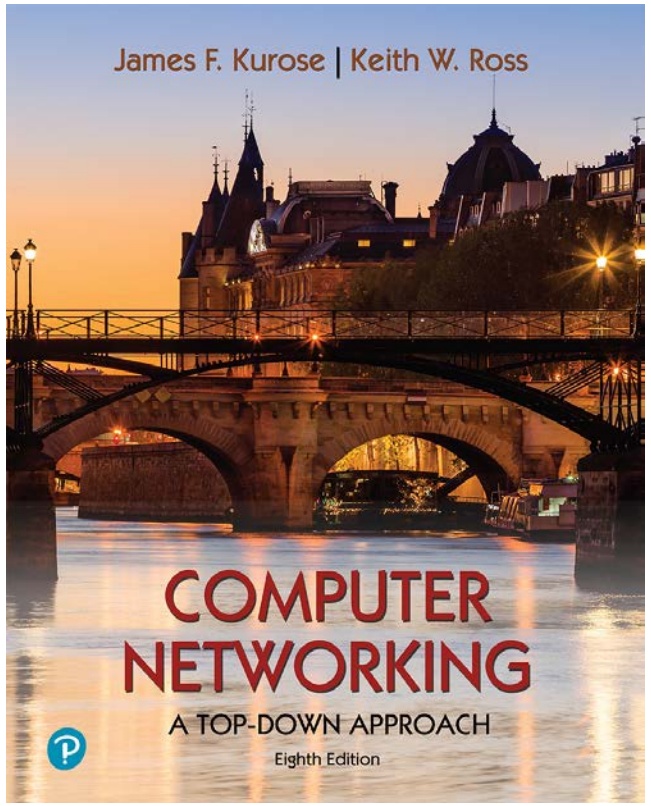
A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

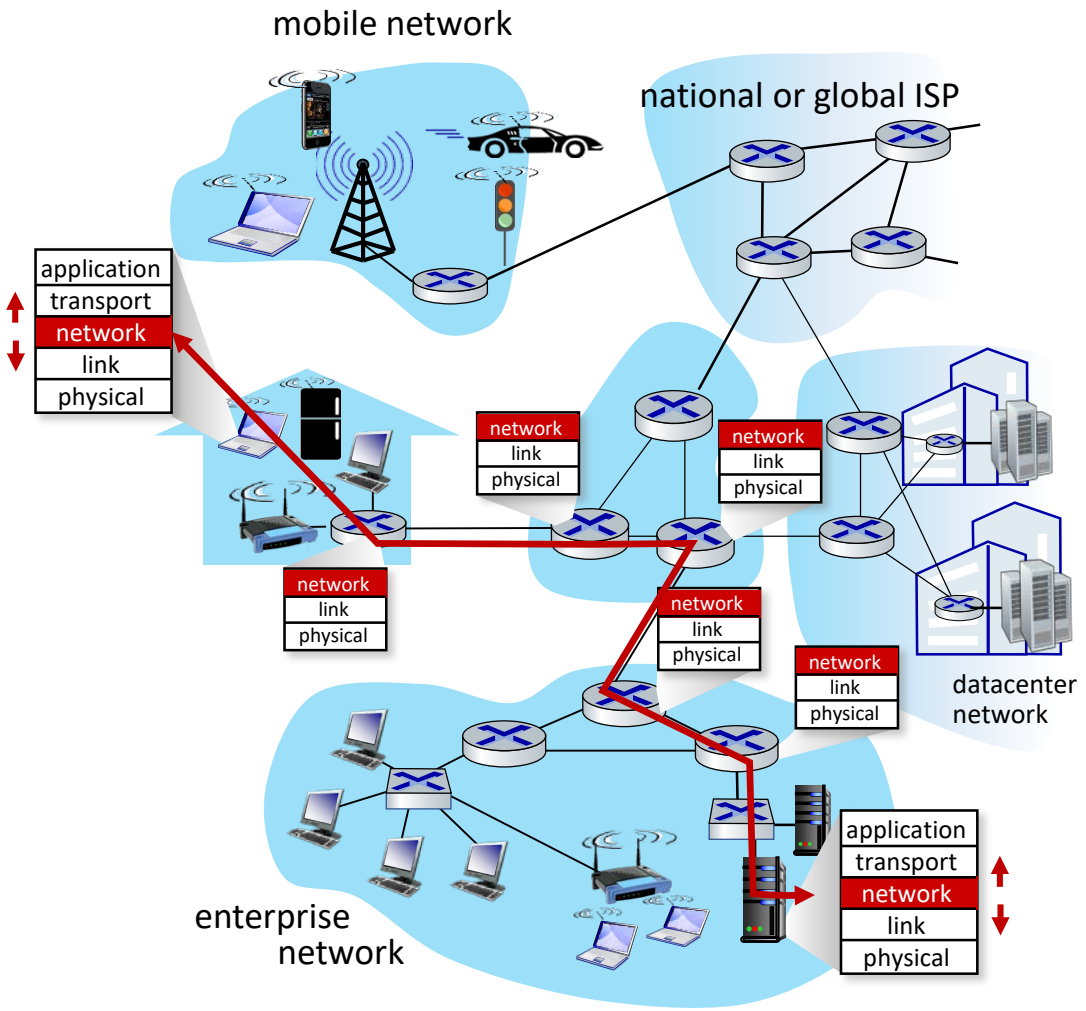


Computer Networking: A Top-Down Approach

8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Network-layer services and protocols

- transport segment from sending to receiving host
  - sender: encapsulates segments into datagrams, passes to link layer
  - receiver: delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- routers:
  - examines header fields in all IP datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



-
اینو قبلا گفته
وقتی که ساختار شبکه به این صورت است که یک مش داریم و بسته ما مسیرهایی که از طریق این مش پیدا میکنه باید به مقصد برسه و هر گره این مش باید تصمیم بگیره که قدم بعدی چی هست و این لازمش این است که تک تک نودهای توی مسیرمون مشارکت داشته باشند توی این کار

Two key network-layer functions

network-layer functions:

- *forwarding:* move packets from a router's input link to appropriate router output link
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination

forwarding

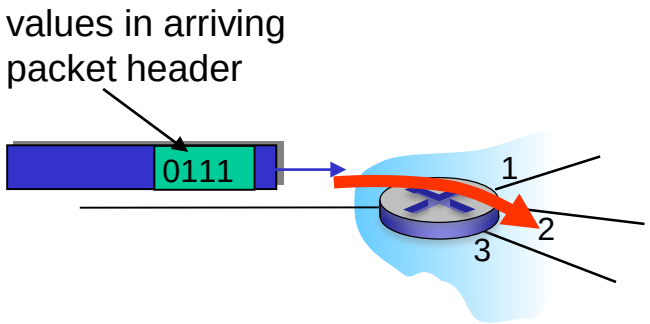
routing

-
که این کار در صفحه قبلی فورواردینگ است و روتینگ
فورواردینگ: وقتی پکت می رسه به یک روتر این روتر باید تصمیم بگیره که اینو به کدوم پورت بفرسته و این ارسال رو انجام بده و تصمیم گیری براساس یکسری اطلاعاتی که از قبل براش اماده شده ینی یک جدولی که بهش میگفتیم فورواردینگ تیبل که تصمیم گیری رو بر این اساس انجام میده
روتینگ: این تیبل از کجا میاد؟ روتینگ انجام میگیره و براساس یکسری الگوریتم هایی روتینگ انجام میشه و برای هر نود مشخص میشه که برای مقصد های مختلف توی اون نود کدوم مسیر خوبه که این میشه فورواردینگ تیبل
روتینگ یک کار گلوبال است که برای همه شبکه باید انجام بشه
و فورواردینگ فقط در اون روتر انجام میشه

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port



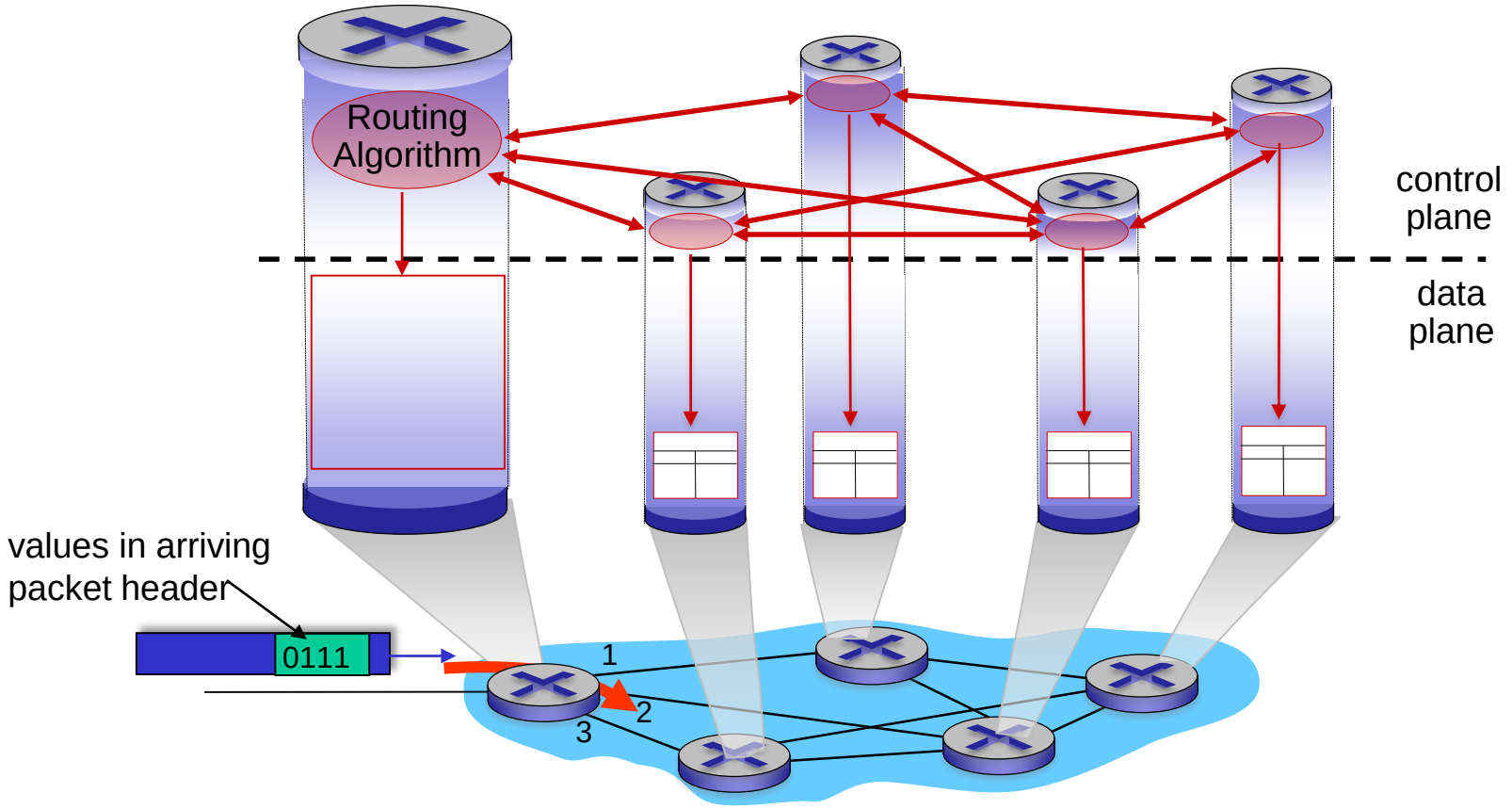
Control plane

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host

-
اون قسمتی که فورواردینگ رو انجام میده بهش دیتا پلن می گیم و در هر روتر بسته که می رسه به پورت ورودی به پورت خروجی مورد نظر فورووارد میشه که این کاور در دیتا پلن انجام میدیم و کنترل پلن اونجایی است که روتینگ انجام میشه و روتینگ برای کل شبکه انجام میگیره و توی اسلایدهای قبلی دوتا پروتکل گفتیم که هر دوی پروتکل مستلزم این بود که تک تک روترها مشارکت داشته باشند و توی یکی از این ها مشارکت این بود که اطلاعاتشون رو به بقیه میدادن (پروتکل لینک استیت) و هر روتر به تنهایی الگوریتم رو اجرا میکرد و مسیرها رو به دست میاورد و براساس اون فورواردینگ تیبل خودش رو درست میکرد منتها توی این پروتکل باید اطلاعات نودها یکسان باشه چون اگر اطلاعات نودها یکسان نباشه یک نود مسیر اون درخت رو یک جور به
دست میاره و یک نود دیگر یک جور دیگر در نتیجه پکت ما ممکنه توی نود بعدی که درخت متفاوته ممکنه یک مسیر دیگه رو بره پس شرط اینکه روتینگ درست انجام بگیره توی شبکه اینه که همه نودها اطلاعات یکسان داشته باشند پس این مشارکت برای این است که این اتفاق بیوفته و در پروتکل نوع دوم از اطلاعات هم دیگه استفاده میشه و براساس این اطلاعات مسیرها به دست میاد و این هم مستلزم این است که همه روترها مشارکت داشته باشند توی این کار - کنترل پلن باید توزیع شده روی روترها باشه و این وضعیت تا حالا توی شبکه بوده به صورت سنتی ینی کنترل پلن در همه روترها باید اجرا بشه پس روتر ما از دو تا plane تشکیل شده: Data plane و Control plane
Data plane میاد فورواردینگ رو انجام میده و Control plane هم برای اجرای پروتکل های روتینگ استفاده میشه
دیدگاه جدیدی که SDN است و اکنون داره پیاده سازی میشه و در این دیدگاه یه خورده فرق میکنه ینی ما کنترل پلن رو از دیتا پلن جدا میکنیم ینی میگیم این دوتا چیز مستقل هستند و دلیل نداره با هم جمع بشن توی یک روتر

Per-router control plane

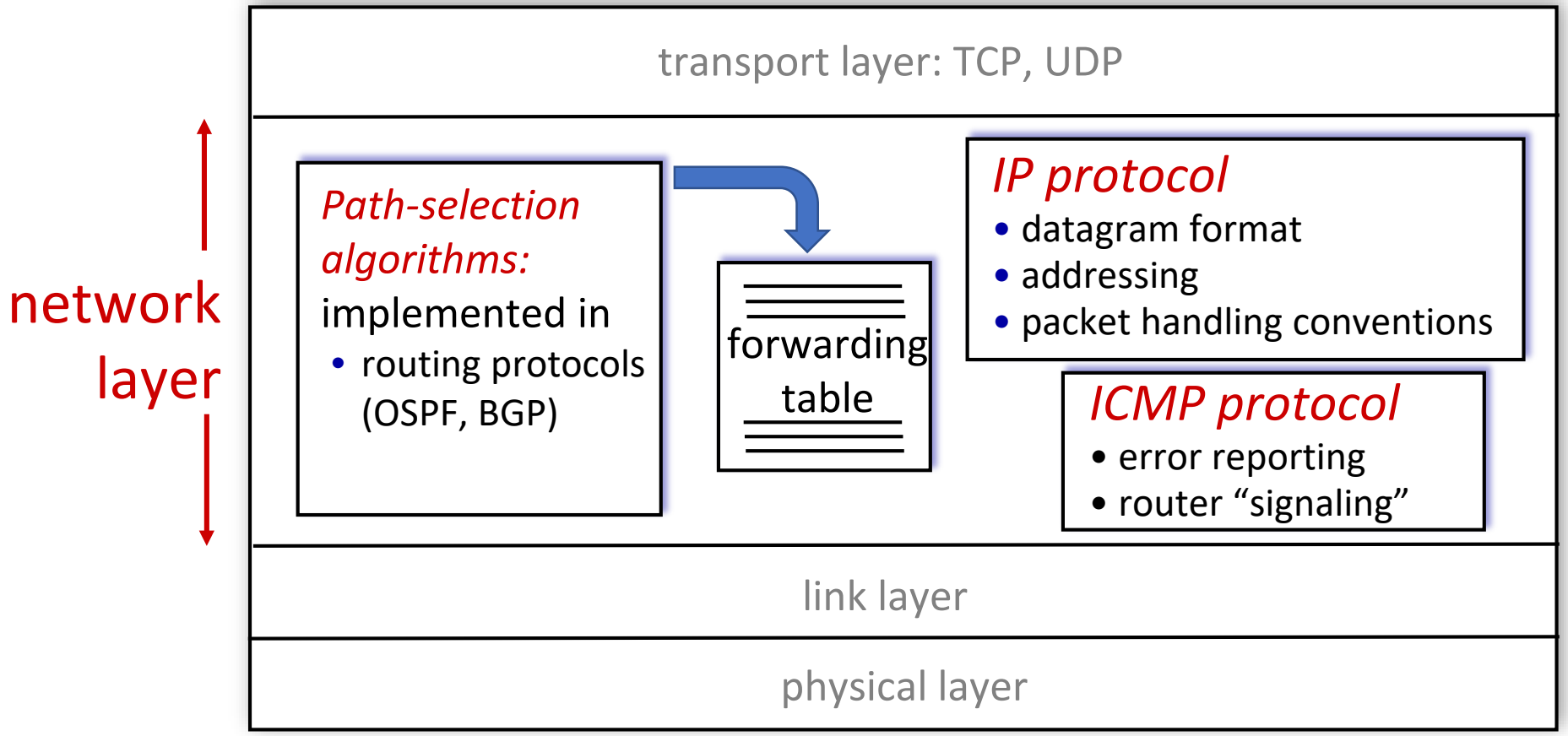
Individual routing algorithm components *in each and every router* interact in the control plane



-
در دیدگاه سنتی هر روترمون که دیتاپلن رو انجام میده کنترل پلن خودش رو داره و کار این کنترل پلن اجرای روتینگ الگوریتم است با مشارکت روترهای دیگر در سطح شبکه و خروجیش میشه فورواردینگ تیبل
خیلی مهمه اطلاعاتشون یکسان باشه
توی دیدگاه جدید: کنترل پلن و دیتاپلن از هم تفکیک میشه حالا سوال اینه که چرا اینا باید پهلوی هم باشن؟ کنترل پلن اون درخت رو تشکیل میشه - فرض میکنیم این لینک استیت رو بین روترها رد و بدل کردیم و همه روترها این دیتابیس لینک استیت رو دارند و از اون گراف شبکه رو می سازن و در حالت درست اگر این کار درست انجام شده باشه گرافی که تک تک روترها به دست میارن یکسان است و بعد الگوریتم دایکسترا هر کدوم اجرا میکنن تا به دست بیاد اون جدول حالا اگر همین کارو یک سروری بیاد انجام بده که همین دیتابیس که کپیش توی همه روترها است یک کپی بفرستیم به اون سرور و اون سرور الگوریتم دایکسترا رو برای همه نودها اجرا بکنه و بعد این سرور بیاد برای هر روتر جدولش رو استخراج بکنه و بفرسته براش : این دیدگاه جدیدی است که توی این دیدگاه جدید یک کنترلر داریم و پروتکل روتینگ و الگوریتم هارو این کنترلر اجرا میکنه و توی هر روتر یک CA داریم پس کنترل پلن رو از دیتاپلن جدا کردیم و دیتاپلن باید توی هر روتر بمونه و کنترل پلن این که مسیر مناسب چی هست نیاز نیست توی روترها انجام بشه پس روترها فقط دیتاپلن رو انجام میدن

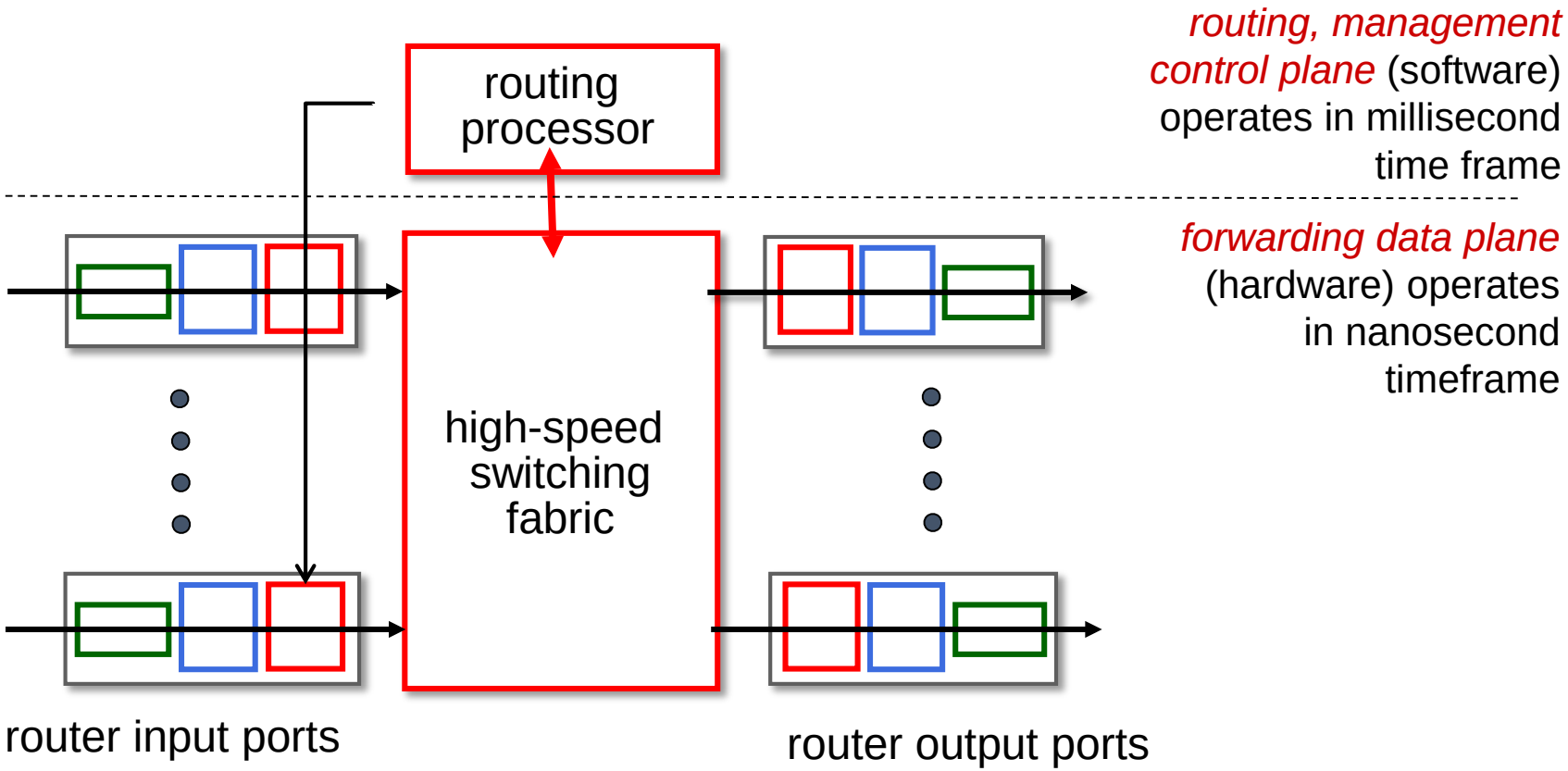
Network Layer: Internet

host, router network layer functions:



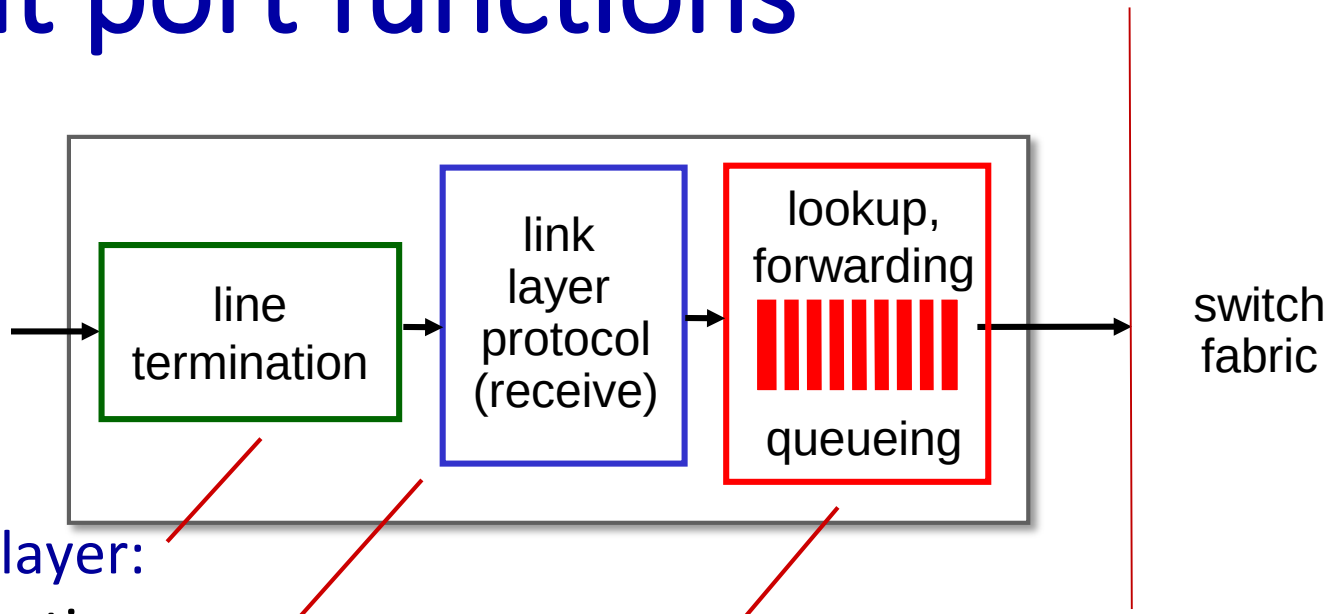
Router architecture overview

high-level view of generic router architecture:



-
معماری روترها:
یک روتر از سه قسمت تشکیل شده:
1- یکی پورت های ورودی و خروجی هستند و چون فانکشن پورت ورودی و فانکشن پورت خروجی متفاوت است ما اینارو جدا از هم نشون میدیم اینجا ولی در واقعیت توی روتر یک پورت داریم که از دو قسمت ورودی و خروجی تشکیل شده : پس این روتر تعدادی پورت داره که توی شکل نشون داده
2- switching fabric است که یک شبکه ارتباطی بین پورت های ورودی و خروجی است ینی در حالت کلی یک شبکه ارتباطی بین پورت ها است و از طریق این شبکه ارتباطی یک پکتی که از پورت ورودی میاد باید بتونه به هر یک از پورت های خروجی ارسال بشه که طبیعی است ینی کار روتر این است که یک بسته که میاد باید ببینه به کدوم خروجی ارسال بشه و اون بفرسته به اون پورت خروجی پس چجوری می فرسته با این شبکه ارتباطی که بهش switching fabric می گن چون همه پورت های ورودی باید بتونن به همه پورتهای خروجی بسته ارسال کنند
3- routing processor در واقع هسته کنترلی روتر ما است که یکسری کارهایی رو انجام می ده و مهمترین کاری که انجام میده بحث روتینگ است که در واقع اون روتینگ پروتکلی که گفتیم اینجا اجرا میشه و فورواردینگ کجا انجام میشه ؟ بین پورت های خروجی و ورودی از طریق switching fabric انجام میشه
به این ترتیب control plane و data plane رو میشه به این صورت جدا کرد توی روتر
عملا routing processor همون control plane رو تشکیل میده و کارهایی مختلفی انجام میده از جمله روتینگ و عمدتا نرم افزاری است و سرعت بالا است در حد میلی سکنت چون کارهایی اینجا انجام میشه در زمان های طولانی تری می تونن اتفاق بیوفتن از جمله روتینگ
و فورواردینگ پلین در حد نانوسکنت است چون هر پکت که می رسه این باید در یک پکت تایم فوروارد بشه به پورت خروجی و توی این بخش عملیات سریعتر انجام میشه ینی بخش forwarding data plane که قسمت پایین است چون تعداد زیادی پکت در یک زمان کوتاه می رسن و باید فوروراد بشن با سرعت line مون
control plane پروتکل های روتینگ رو اجرا میکنه و بعد اون tree رو به دست میاره و بعد براساس اون tree فورواردینگ تیبل هارو می سازه و در اختیار پورت های ورودی قرار میده.. ادامه صفحه بعدی..

Input port functions



physical layer:
bit-level reception

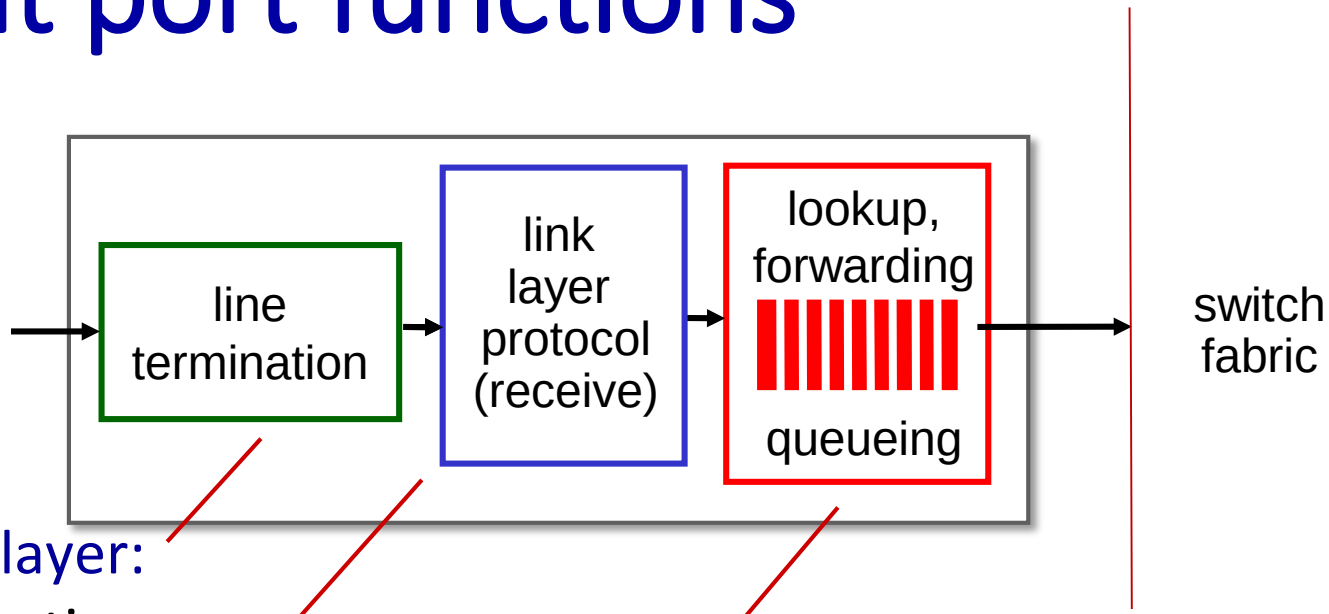
link layer:
e.g., Ethernet
(chapter 6)

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- input port queuing: if datagrams arrive faster than forwarding rate into switch fabric

-
ادامه صفحه قبل:
یک بسته که به یک پورت ورودی می رسه این بسته در لایه فیزیکال receiver است ینی لینکمون رو داریم ینی بسته به این که کانال و لینکمون چی هست یک receiver مناسب باید اینجا باشه ینی بیت ها و صفر و یک ها رو دریافت میکنه و تشخیص میده
و بعد این ها رو به لایه link layer میده و لایه لینک ابتدای و انتهای فریم های لایه لینک رو تشخیص میده و فریم هارو جدا میکنه و بعد هدر لایه لینک ینی فریم لایه دو رو بررسی میکنه و کارهایی که پروتکل لایه دو باید انجام بده مثل Ethernet و کارهای لایه دو انجام میشه - و در روترها لایه سه هم داریم و لایه سه چیه؟ این پکتیه که توی این فریم هست ینی توی بخش لایه
لینک و این استخراج میشه و پکت داده میشه به قسمت لایه نت ورکمون (همون مستطیل قرمزه میشه) که این عمدتا میاد هدر بسته لایه شبکه رو بررسی می کنه و از جمله مهمترین کاری که اینجا هست اینه که ادرس مقصد رو نگاه میکنه و براساس ادرس مقصد و lookup که توی فورواردینگ تیبل انجام میده تشخیص میده به کدوم پورت خروجی ارسال بشه از طریق این switching fabric
برای مستطیل قرمز: کار توی این مستطیل باید توی لاین اسپیت انجام بشه چون تا پکت بعدی نرسیده این پکت باید پردازش و ارسال بشه برای یک پورت ورودی همه کارهایی که باید توی این سه مرحله انجام بشه انجام میشه و میاد می رسه به اینکه ارسال بشه به پورت خروجی ینی می رسه به switching... و اینجا بسته به اینکه معماری fabric مون چی هست ممکنه کانتکشن پیش بیاد بعضی از سوییچ فبریک ها به صورتی هستند که هر لاین کاردی در هر پکت تایم می تونه یک پکت بفرسته به هر خروجی دلخواه ولی بعضی از لاین کاردها معماریشون به گونه ای است که برای یک پورت خروجی یکسان یک پکت نمیشه در یک بار فرستاد مثلا الان دوتا پکت از ورودی داریم که میخوان برن به پورت خروجی مثلا شماره 2 هر دو نمی تونن بفرستن یکیش می فرسته و اون یکی نمی تونه بفرسته پس مستطیل قرمز باید اینو بافر بکنه پس پروسس پورت ورودی لاین اسپیت انجام میشه ولی ارسال به پورت خورجی بسته به این که معماری سوییچ می تونه موکول بشه به بعد پس پکت باید توی مستطیل قرمز بمونه و این ینی اینجا باید بافر داشته باشیم و این یعنی این که اینجا باید صف تشکیل بشه چون ممکنه دفعه های بعدی هم این اتفاق بیوفته پس اینجا بحث صف هم داریم

Input port functions



physical layer:
bit-level reception

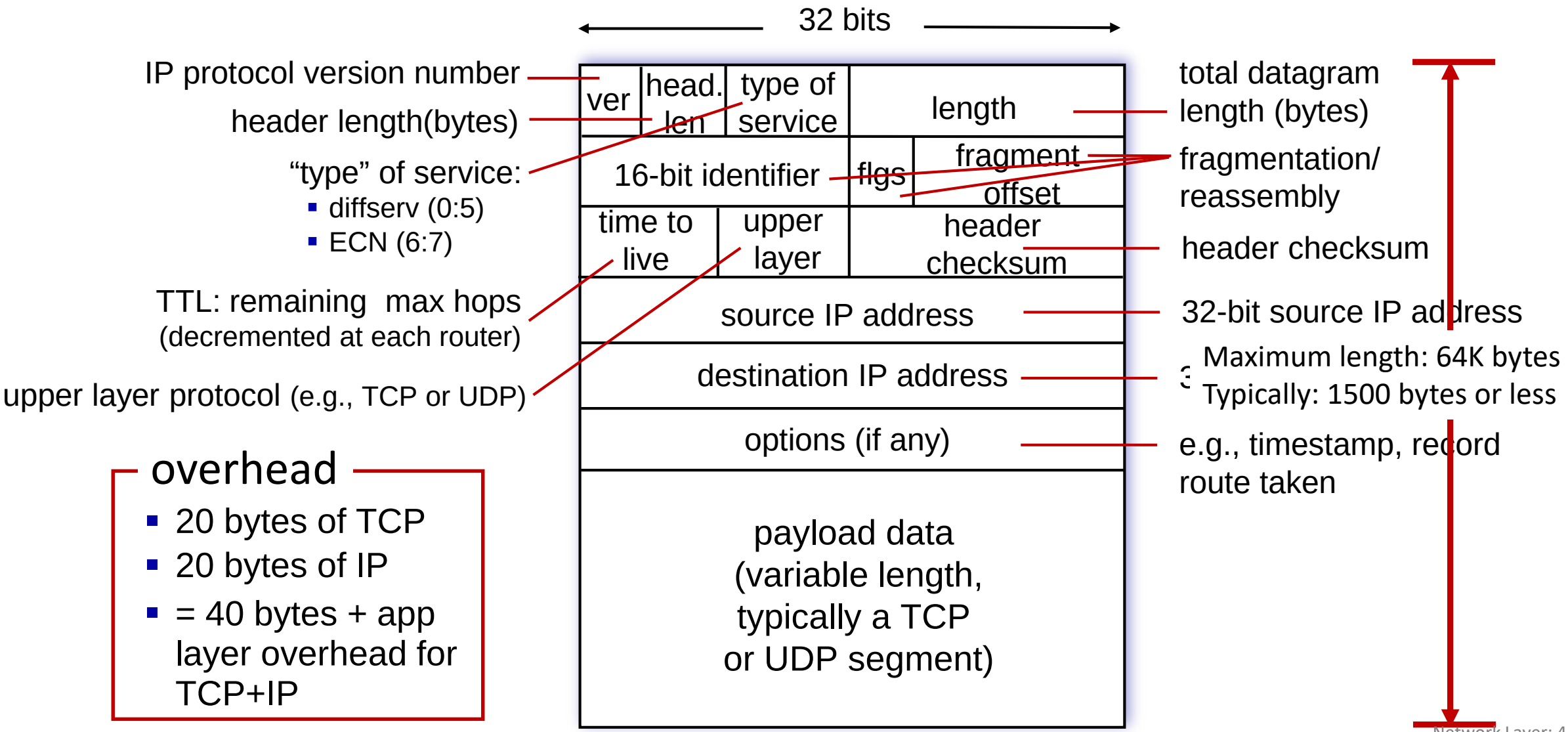
link layer:
e.g., Ethernet
(chapter 6)

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- destination-based forwarding: forward based only on destination IP address (traditional)

-
توی مستطیل قرمز باید lookup انجام بشه و این lookup براساس ادرس مقصد پکت است و ادرس مقصد پکت هم قبلا گفتیم توی هدر پکت لایه شبکه یک فیلدی هست به اسم دستینیشن ip ادرس که اونجا ادرس 32 بیتی برای مقصد گذاشتیم و براساس اون lookup انجام میگیره
دیدگاه سنتی: ینی اصولا شبکه ها بر این اساس طراحی شدند - توی این دیدگاه محدود به این هستیم که ادرس مقصد چی هست مثلا دوست داریم براساس اینکه اپلیکیشن + ادرس مقصد چی هست بیایم تصمیم بگیریم که میشه دیدگاه جدید که علاوه بر ادرس مقصد هم می تونیم چیزهای دیگه هم داشته باشیم - حالا این کجا به درد میخوره؟ که مثلا تشخیص بدیم اپلیکیشن چی هست؟ ما اگر از این نود به مقصد مورد نظر که براساس ip مقصد مشخص میشه و دوتا مسیر داشته باشیم که یکی مسیر هزینه اش بیشتره و سریعتر میرسه ولی اون یکی برعکس تاخیرش زیاده ولی هزینه اش کمتر و پکت لاسش بالا است --> حالا اپلیکیشنی که براش پکت لاس مهمه از اون مسیر پرهزینه می فرستیم و هزینه براش مهم نیست ولی اپلیکیشنی که براش تاخیر و پکت لاس مهم نیست از اون مسیر کم هزینه می فرستیم حالا اگر اینجا فقط به مقصد نگاه میکردیم برای یک مقصد مشخص فقط یک پورت خروجی مشخص میشه مثلا با الگوریتم دایکسترا و فقط به همون می تونیم بفرستیم و دیگه نمی تونیم فرق بذاریم بین اپلیکیشن های مختلف
پس توی generalized forwarding فرق می ذاریم بین پکت های کانکشن های مختلف از مبداهای مختلف به مقصد یکسان و... اینطوری خیلی دقیق تر می تونیم روتینگ رو انجام بدیم پس فورواردینگ تیبل باید با قبلی متفاوت باشه اینجا
دیدگاه جدید: پس توی این دیدگاه بیایم همه فیلدهارو نگاه بکنیم و براساس اون فیلدها ببینیم کدوم مقصد خوبه

IP Datagram format



overhead

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

فرمت IP Datagram

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010000 00000100	0
11001000 00010111 00010000 00000101 through 11001000 00010111 00010000 00000111	1
11001000 00010111 00011000 11111111	0
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	********	0
11001000	00010111	00011000	********	1
11001000	00010111	00011***	********	2
otherwise				3

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001	which interface?	
11001000 00010111 00011000 10101010	which interface?	

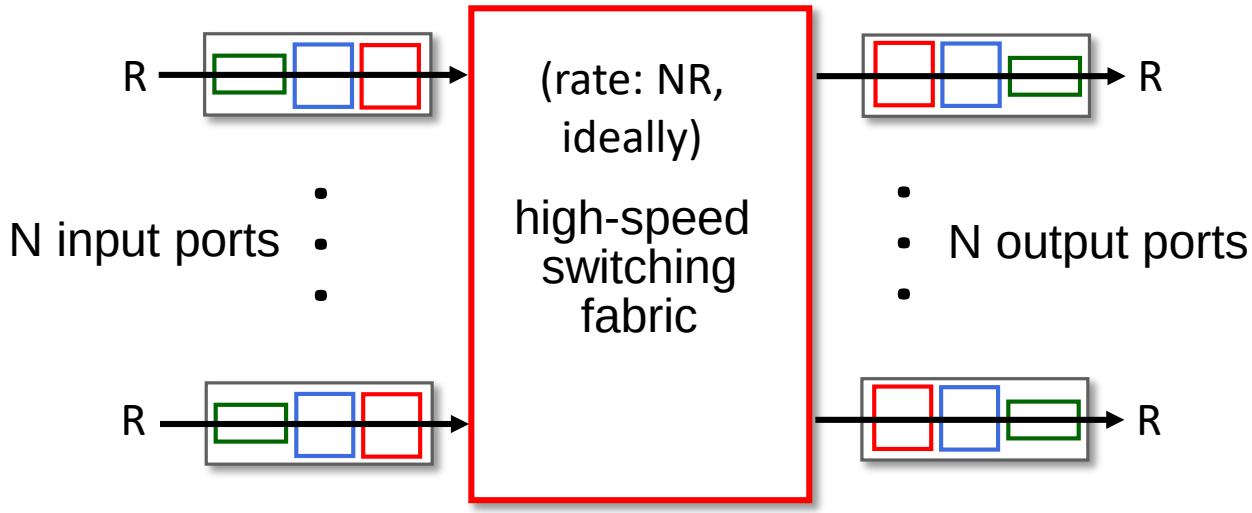
Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: ~1M routing table entries in TCAM

-
در روش سنتی که براساس ادرس مقصد فورواردینگ انجام میگیره ما از Longest prefix matching استفاده میکنیم و برای پیاده سازی Longest prefix matching از TCAM استفاده میشه
حالا برای generalized forwarding باید اینو تعمیم بدیم به فیلدهای دیگر منتها فقط ادرس مقصد است که Longest prefix matching براش معنا داره و فیلدهای دیگر الزاما اینطوری نیستن

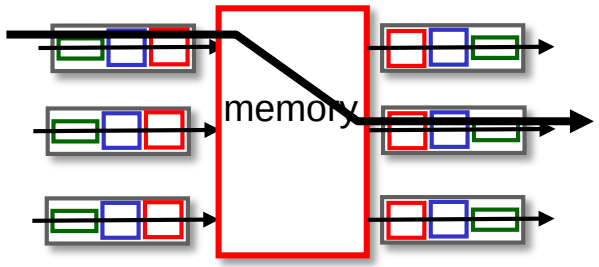
Switching fabrics

- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfered from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable

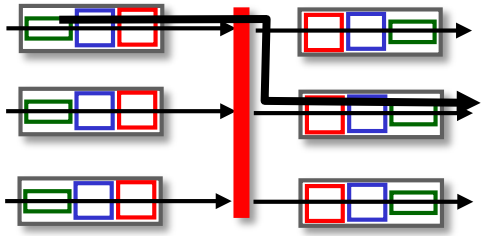


Switching fabrics

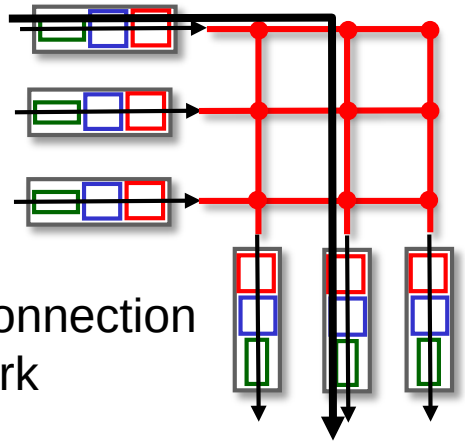
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfered from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



memory



bus



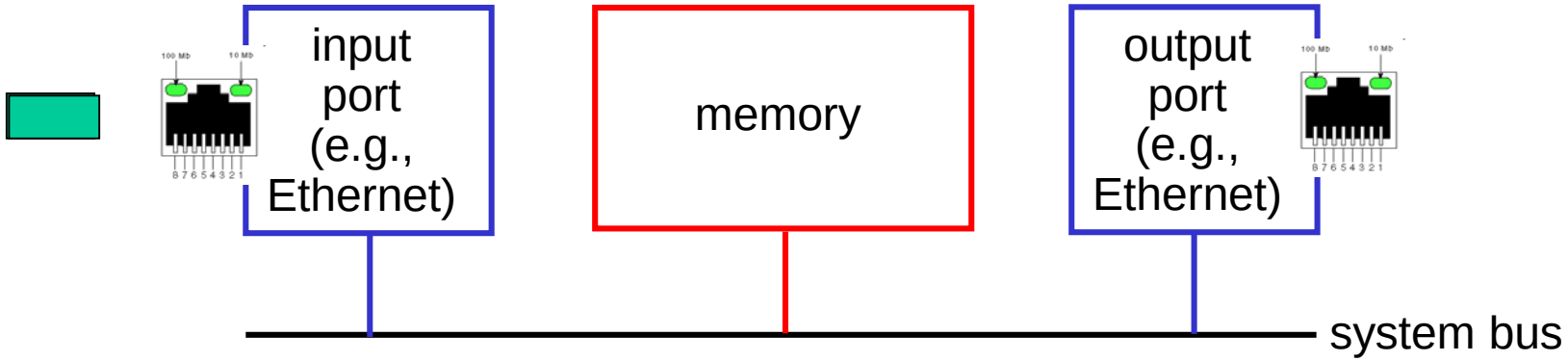
interconnection network

-
یک روش پیاده سازی در روترهای با سرعت پایین تر معمولا اینه که از حافظه استفاده میشه ینی همه پورت های ورودی پکت رو توی حافظه می نویسن و بعد پورت های خروجی از حافظه می خونن و ارسال میکنن - پورت ورودی وقتی میخواد پکت رو توی حافظه بنویسه توی قسمتی می نویسه که برای اون پورت خروجی مورد نظر اختصاص داده شده و پورت خروجی همیشه می ره از قسمت خودش پکت ها رو برمیداره و ارسال میکنه - یک لینکدلیست هم شکل میگیره که ترتیب این ها هم رعایت و مشخص بشه که کی زودتر بره و کی دیرتر برای یک پورت خروجی یکسان
bus: این شیرد است البته مموری هم شیرد بود - در یک پکت تایم نوبت میده به همه پورت ها که پکت هاشون رو بفرستن منتها مستقیما از طریق باس می فرستن به اون پورت مورد نظر
corssbar یا interconnection network هر پورت ورودی از طریق crossbar با ست شدن سوییچ محل تلاقی سطر و ستون می تونه به محل خروجی بفرسته و مزیتی که داره این است که همزمان است ینی سه تا پکت می تونه همزمان از ورودی ها به خروجی ها ارسال بشه ولی توی مموری و باس باید به نوبت انجام میشد پس توی این روش سرعت فبریک می تونه کمتر باشه منتها یک شرط داره که این ها مشترک نباشن ینی پورت های ورودی متفاوت به پورت خروجی یکسان نباید در یک لحظه پکت بفرستن حتما باید مقصد هاشون متفاوت باشه - تعداد سوییچ هایی که توی این فبریک است زیادتر است و تنظیم این نقطه ای که گفتیم چک بشه سخته و پیچیدگی بالاتر است - توی این روش به تعداد پورت ها میتونه پکت ارسال بشه ینی n پکت ولی توی مموری و باس اگر بخواهد همون پکت ارسال بشه n برابر باید کلاک فبریک بالاتر باشه
نکته: اگر سرعت مموری n برابر باشه هر پورت ورودی توی هر پکت تایم پکتشو می فرسته به پورت خروجی دلخواه و اگر نباشه ینی بخوان به یک پورت خروجی بره یکیش ارسال میشه و یکیش می مونه و صف تشکیل میشه توی ورودی همونی که توی اسلایدهای قبلی گفتیم حالا وقتی که پکت ها ارسال میشن به خروجی صف توی پورت خروجی تشکیل میشه ینی از دوتا پورت ورودی متفاوت می خواد بره به یک پورت خروجی و بعد این دوتا که نمی تونن به لینک خروجی ارسال بشن به نوبت باید ارسال بشن پس صف داریم اینجا ... ادامه صفحه 24

Switching via memory

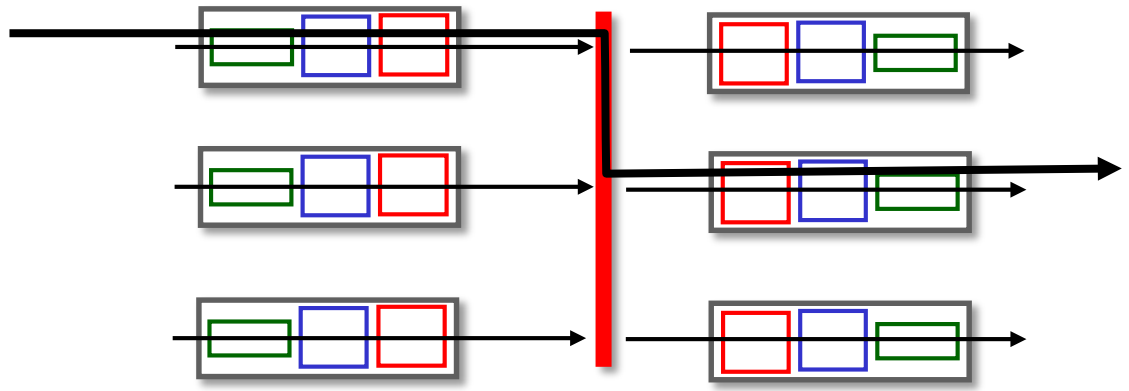
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via a bus

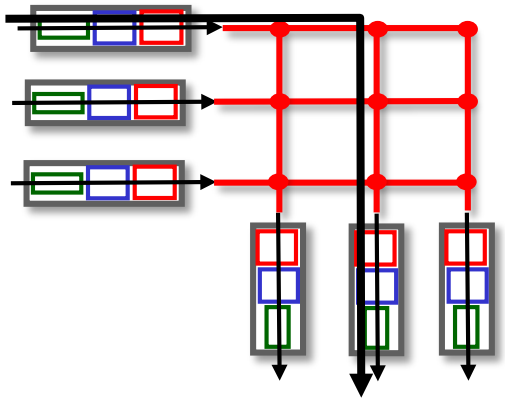
- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers



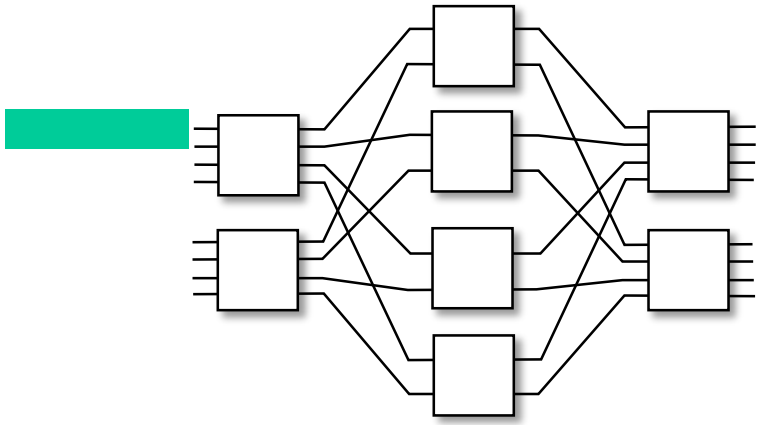
--

Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- multistage switch: *nxn* switch from multiple stages of smaller switches
- exploiting parallelism:
  - fragment datagram into fixed length cells on entry
  - switch cells through the fabric, reassemble datagram at exit



3x3 crossbar

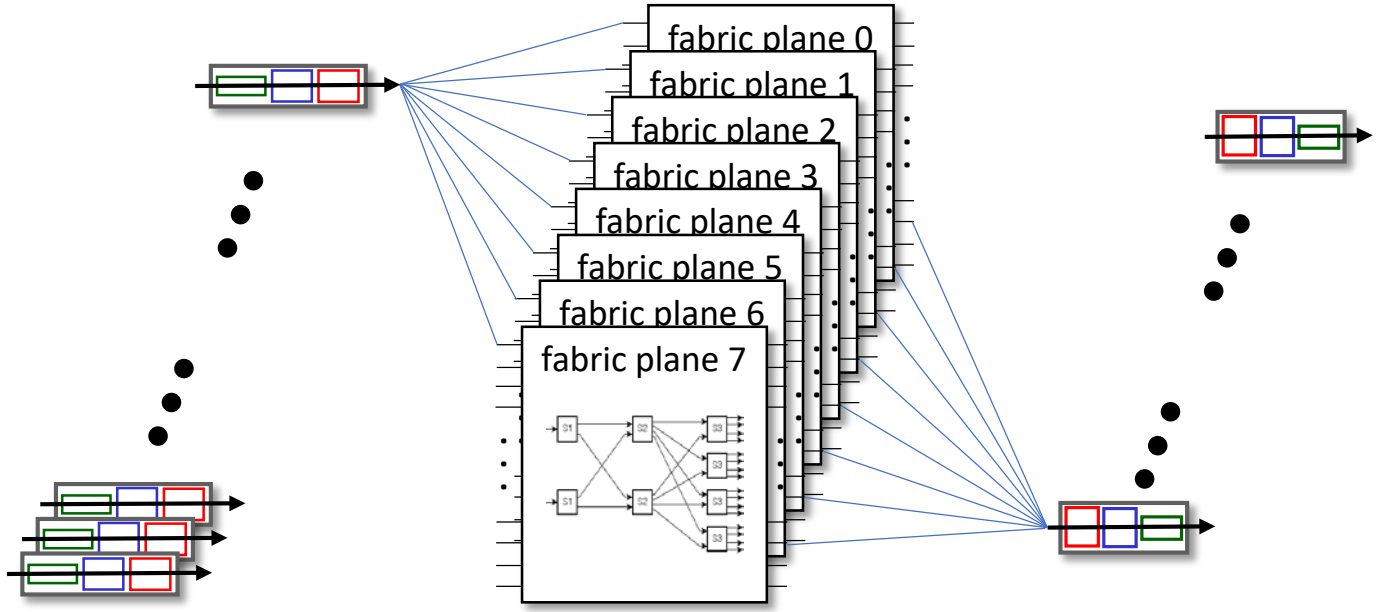


8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network

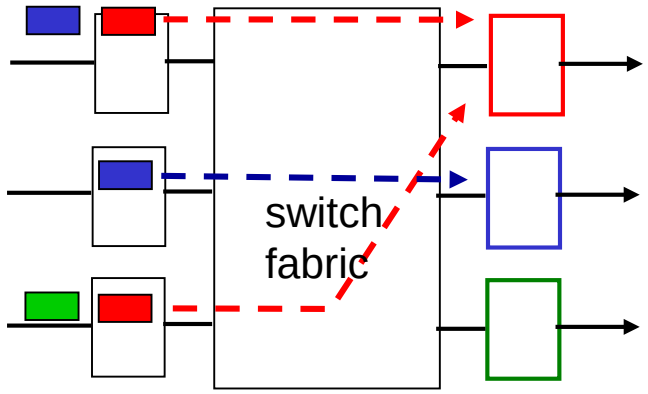
- scaling, using multiple switching “planes” in parallel:
  - speedup, scaleup via parallelism

- Cisco CRS router:
  - basic unit: 8 switching planes
  - each plane: 3-stage interconnection network
  - up to 100’s Tbps switching capacity

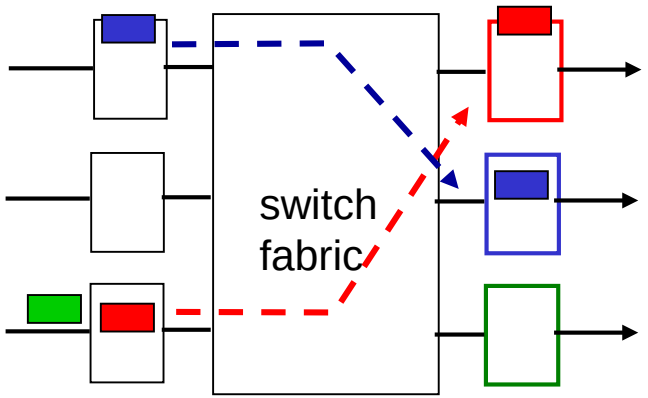


Input port queuing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

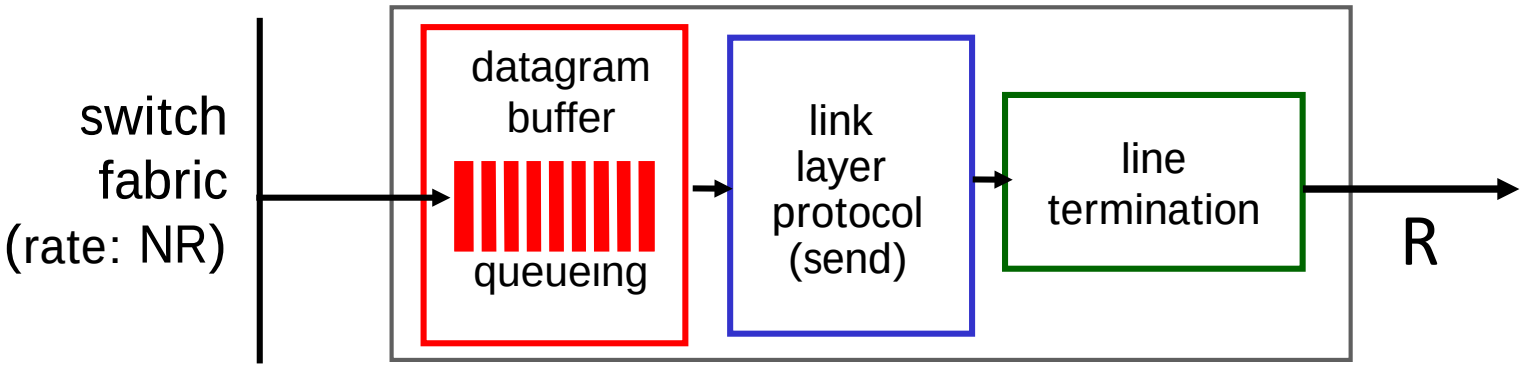


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

Output port queuing



This is a really important slide

- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

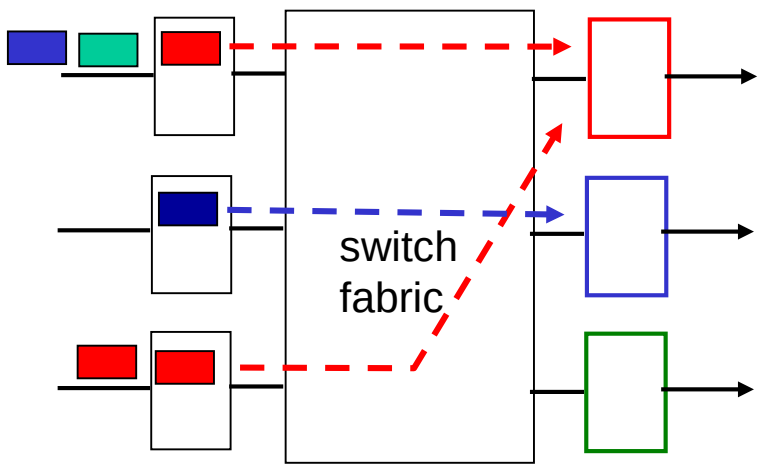
→ Datagrams can be lost due to congestion, lack of buffers

- *Scheduling discipline* chooses among queued datagrams for transmission

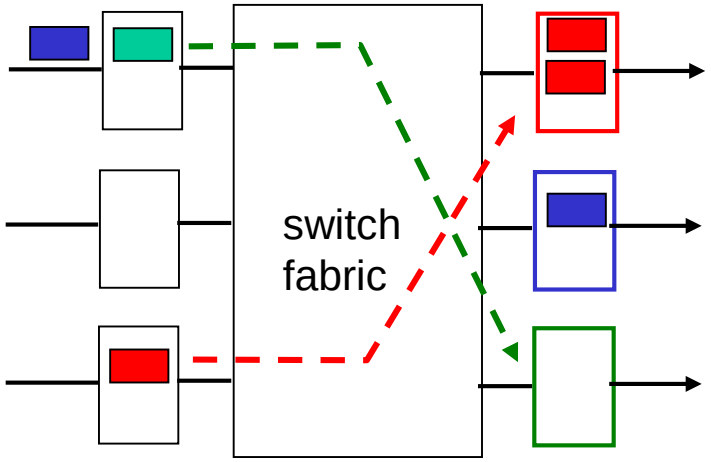
→ Priority scheduling – who gets best performance, network neutrality

-
پس توی پورت خروجی هم باید بافر داشته باشیم و بتونیم ذخیره بکنیم و این ذخیره شدن باید برای تعداد کافی از پکت ها امکان پذیر باشه - اگر اینجا صف طولانی تر شد تاخیر صف ایجاد میشه - اینجا یک سوال پیش میاد --> اگر بسته ای که توی صف داریم و بسته ای داریم که مال مکالمه تلفنیه و تاخیر براش مهمه و نباید خیلی معطل بشه و یک بسته داریم مال فایل ترنسفره و دیر هم بره مشکلی نیست و ممکنه فایل ترنسفره زودتر بیاد توی صف تا مکالمه تلفنی ولی ما مکالمه تلفنی رو بفرستیم زودتر تا اون یکی رو که به این میگیم Scheduling
Scheduling : پس هر جا صف تشکیل بشه و ما پکت از نوع های مختلف داشته باشیم باید ببینیم Scheduling مون چی هست و چجوری اینارو Scheduling کنیم توی روترهای سنتی این Scheduling به صورت هرکی زودتر بیاد می ره بیرونه ولی توی روترهای جدید الزاما اینطوری نیست که جلوتر میگه بعدا اینو
مسائل عمده دیتاپلن: سرعت ارسال هست و بحث صف و Scheduling است و چون بافر محدوده ممکنه بقیه نیان توی صف اگه طولانی بشه صف و پکت لاس اتفاق بیوفته

Output port queuing



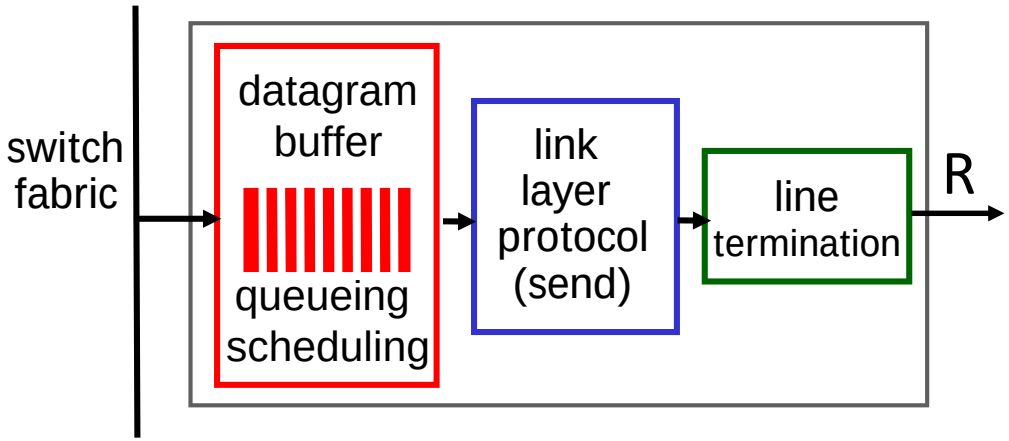
at *t*, packets more from input to output



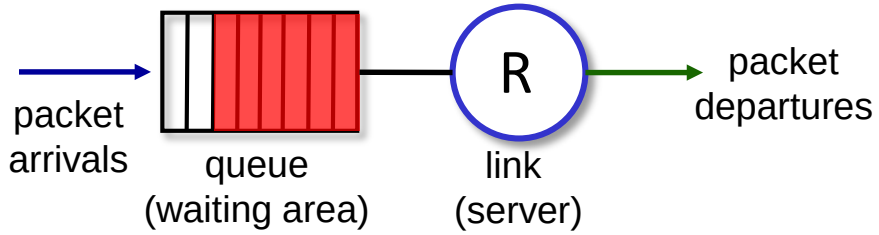
one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Buffer Management



Abstraction: queue



buffer management:

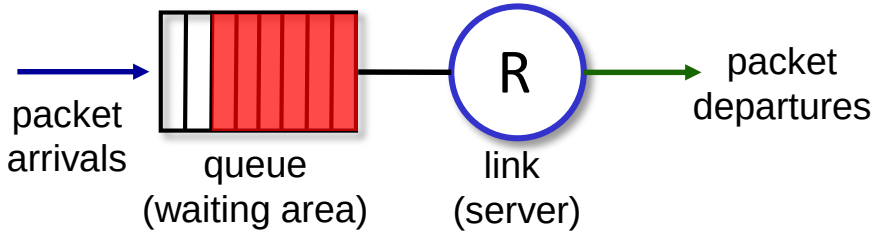
- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)

Packet Scheduling: FCFS

packet scheduling: deciding which packet to send next on link

- first come, first served

Abstraction: queue



FCFS: packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)
- real world examples?
