1396.8.23

به نام خدا

جابر بابکی ۹۶۱۳۱۰۲۰

روتر و سوییچ HW1

1-**یکی از روش‌های کاهش هزینه‌ی جستجو در جدول مسیریابی این است که با استفاده از روشی جدول را به بلوک‌هایی تقسیم کنیم و تنها جستجو را در میان یکی از بلوک‌ها انجام دهیم. به طور مثال در دانشگاه، دو رقم اول شماره دانشجویی هر فرد با سال ورود آن فرد شروع می‌شود (۸۹..... برای دانشجویان ورودی ۸۹، ۹۱..... برای دانشجویان ورودی ۹۱). مشخصات دانشجویان بر مبنای سال ورود آن‌ها در جداول جداگانه‌ای نگه‌داری می‌شود. جستجو برای پیدا کردن مشخصات هر فرد با توجه به سال ورود آن فرد تنها در یکی از جداول انجام می‌شود.**

· اگر فرض کنیم یک جدول مسیریابی شامل ۵۰۰۰ عنصر باشد که این عناصر در ۱۰۰۰۰ بلوک قرار گرفته‌اند.

a) **احتمال اینکه یک بلوک خالی باشد؟ متوسط تعداد بلوک‌های خالی را بدست آورید.**

b) **احتمال اینکه یک بلوک شامل ۲ عنصر باشد؟ متوسط تعداد بلوک‌های ۲ عضوی را بدست آورید.**

الف احتمال یک بلوک خالی..)

P احتمال قرار گرفتن یک عنصر در یکی از ۱۰۰۰۰ هزار بلوک برابر است با $\frac{1}{10000}$

1-p احتمال قرار نگرفتن یک عنضر در یکی از ۱۰۰۰۰ هزار بلوک

جواب مسئله توزیع 1-p با ۵۰۰۰ عنصر است

$$(\mathbf{1}-\frac{\mathbf{1}}{\mathbf{10000}})^{\mathbf{5000}}=\mathbf{۰,۶۰۶۵۲}$$

الف متوسط بلوک خالی..)

برای محاسبه میانگین باید مجموع بلوک های خالی را در حالت حساب کرد:

$$E(n)\sum_{n=0}^{10000} n\times[(1-\frac{1}{10000})^{5000}]^{n}\times[1-(\frac{9999}{10000})^{5000}]^{10000-1}$$

الف احتما یک بلوک خالی... با روش دیگر)

$$p = \frac{\binom{10000}{5000}\times 5000! + 10000}{\binom{9999}{5000}\times 5000! + 9999} = 0.606$$

اثبات روش:

برای حل این سوال ابتدا با یک مثال کوچکتر مسئله را بررسی می کنیم، با فرض ۲ عنصر و ۴ بلوک میتوان مسئله را به شکل زیر حل کرد:

ابتدا باید کل حالات ممکن را بدست بیاوریم در اینجا ۲ تا از ۴ تا مهره را انتخاب می کنیم $\binom{4}{2}$ این دو مهره انتخابی نیز دو جایگشت دارند !۲ و به طور کلی ۴ حالت قابل تصور هست این می شود کل فضای حالت پس داریم:

$$\binom{4}{2} \times 2! + 4$$

حالا با حذف یک بلوک حالت مطلوب را محاسبه می کنیم :

$$\binom{3}{2} \times 2! + 3$$

جواب مسئله حالت مطلوب تقسیم بر حالت کل است:

$$p = \frac{\binom{3}{2} \times 2! + 3}{\binom{4}{2} \times 2! + 4} = \frac{9}{16}$$

با حل این مسئله کوچک می توان به شکل زیر مسئله را فرموله کرد و برای مسائل با ابعاد بزرگتر حل کرد:

N(s) کل حالات مسئله

N(m) حالات مطلوب مسئله

N تعداد کل بلوک ها

n تعداد عناصر

$$N(s) = N + \binom{N}{n} \times n!$$

$$N(m) = (N-1) + \binom{N-1}{n} \times n!$$

$$p = \frac{n(s)}{n(m)}$$

ب احتمال یک بلوک شامل ۲ عنصر...)

P احتمال قرار گرفتن یک عنصر در یکی از ۱۰۰۰۰ هزار بلوک برابر است با $\frac{1}{10000}$

$$\binom{5000}{2} \times p^2 \times (1-p)^{4998} = \binom{5000}{2} \times \left(\frac{1}{10000}\right)^2 \times \left(1-\frac{1}{10000}\right)^{4998} = 0.0752$$

ب احتمال یک بلوک شامل ۲ عنصر ... به روش دیگر)

$$p = \frac{\left(9999 + \binom{9999}{4998} \times 4998!\right) \times \binom{5000}{2}}{\binom{10000}{5000} \times 5000! + 10000} = 0.075$$

ب متوسط...)

$$E(n) = \sum_{n=0}^{10000} n \times \left[\binom{5000}{2} \times p^2 \times (1-p)^{4998}\right]^n \times \left[\binom{5000}{2} \times p^2 \times (1-p)^{4998}\right]^{9998}$$

۲– هزینه (جست و جو، حافظه و بروزرسانی) تکنیک اشاره شده در سوال قبل را بر حسب پارامترهای زیر محاسبه کنید.

- **n = تعداد عناصر موجود در جدول مسیریابی**
- **m = تعداد بلوک‌های موجود**

فرض کنید به طور متوسط تعداد عناصر قرارگرفته در تمام بلوک‌ها با هم برابر باشد.

هزینه جستجو:

برای پیدا کردن یک عنصر ابتدا باید بلوک مورد نظر را پیدا کرد از آنجا که تعداد بلوک ها ثابت است و محدود پس هزینه O(1) دارد، اما در داخل هر بلوک با توجه به اینکه در هر بلوک i تا عنصر وجود دارد اگر این عناصر درون هر بلوک مرتب باشد هزینه O(log i) دارد اگر نامرتب باشد هزینه O(i) دارد.

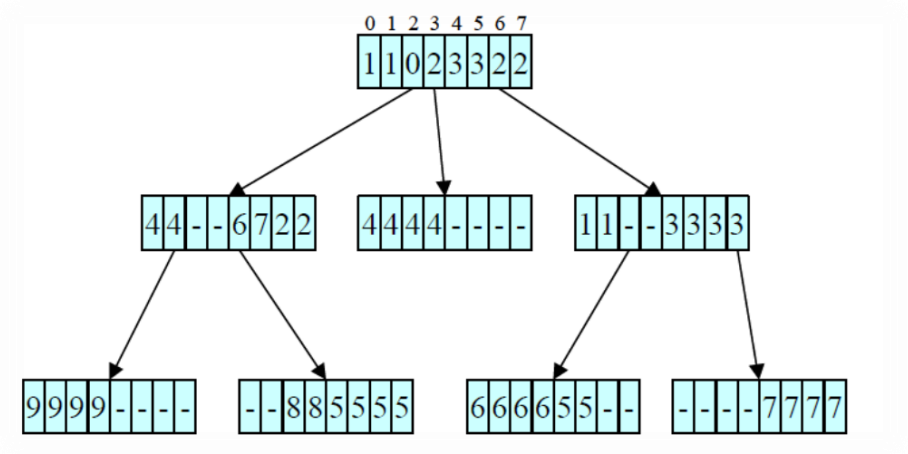
هزینه حافظه:

در بدترین حالت O(m) است

هزینه بروز رسانی:

برای بروزرسانی در بدتری حالت O(1) است

۳– درخت binary **معادل بادرخت** MultiBit **زیر را پیاده سازی کنید.**



برای بدست آوردن هر next hop با توجه به سطحی قرار دارد و آدرس دودویی هر خونه بدست می آید به طور مثال :Next hop =5

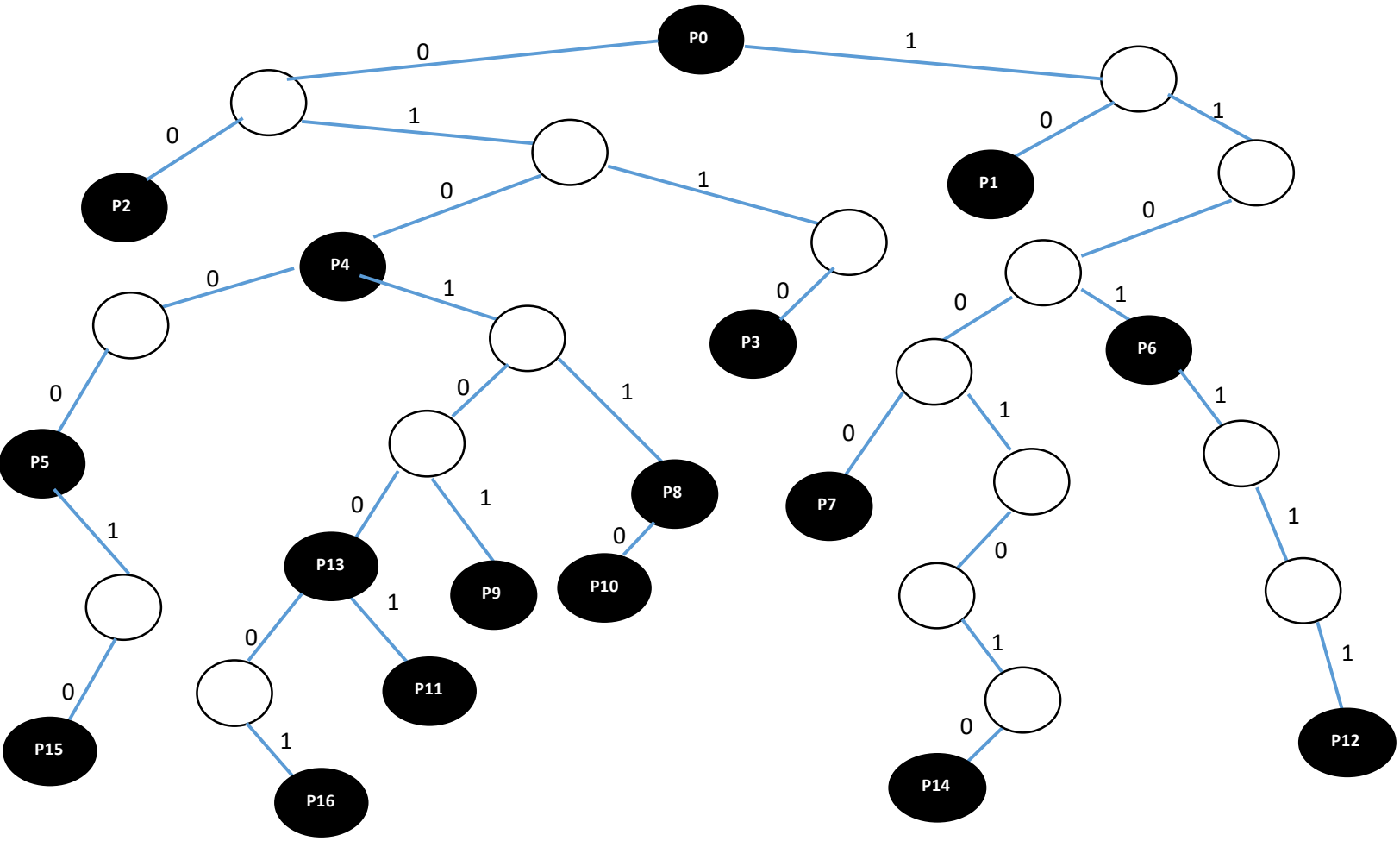
در سطح اول اشاره گر از خانه ۶ خارج شده پس داریم ۱۱۰

در سطح دوم اشاره گر از خانه ۲ خارج شده است پس می شود ۰۱۰

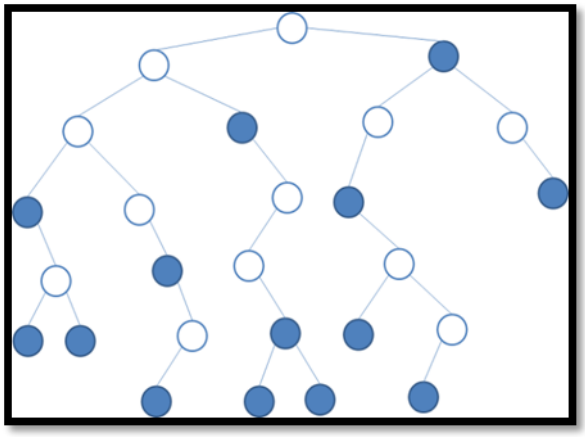
در سط سوم باید دو ۴ و ۵ را با هم aggriate کنیم که می شود ۱۰۰ و ۱۰۱ ==> ۱۰*

به همین ترتیب برای بقیه next hop ها بدست می آوریم و بعد جدول ترای را می کشیم.

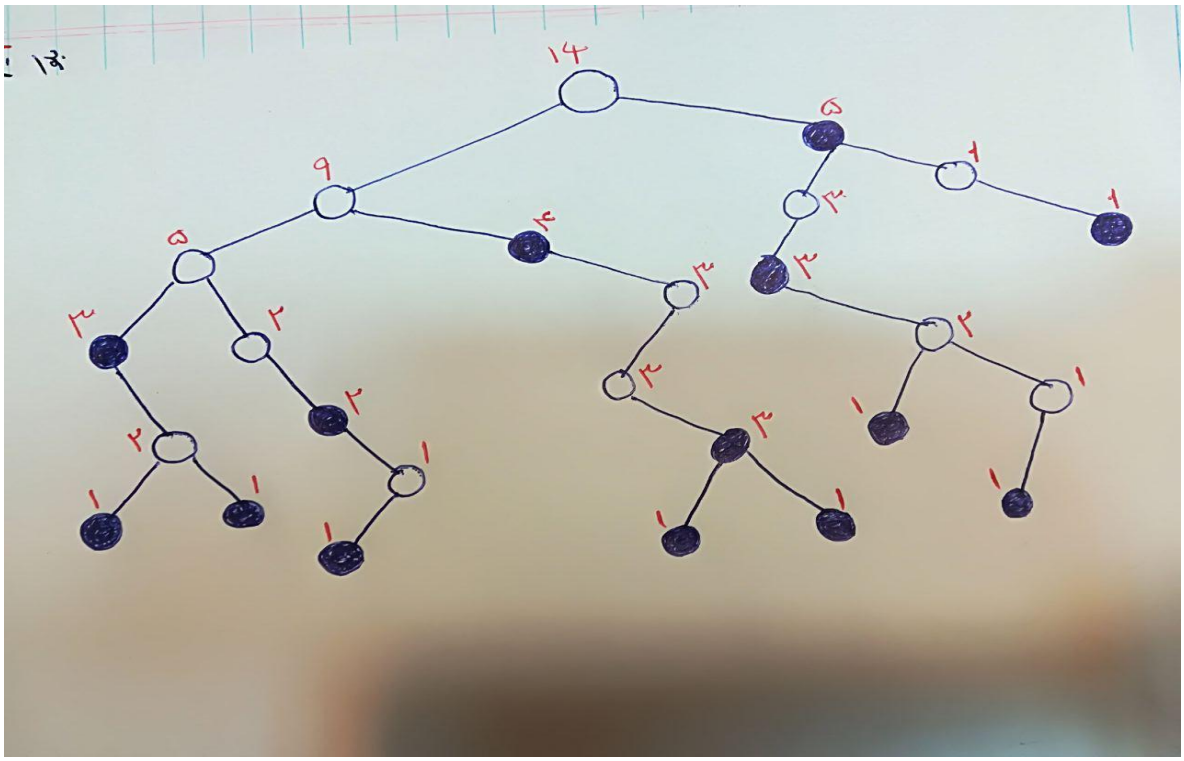
name	prefix	Next hop
P_0	*	2
P_1	10*	3
P_2	00*	1
P_3	0110*	4
P_4	010*	0
P_5	01000*	4
P_6	1101*	3
P_7	11000*	1
P_8	01011*	2
P_9	010101*	7


P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16

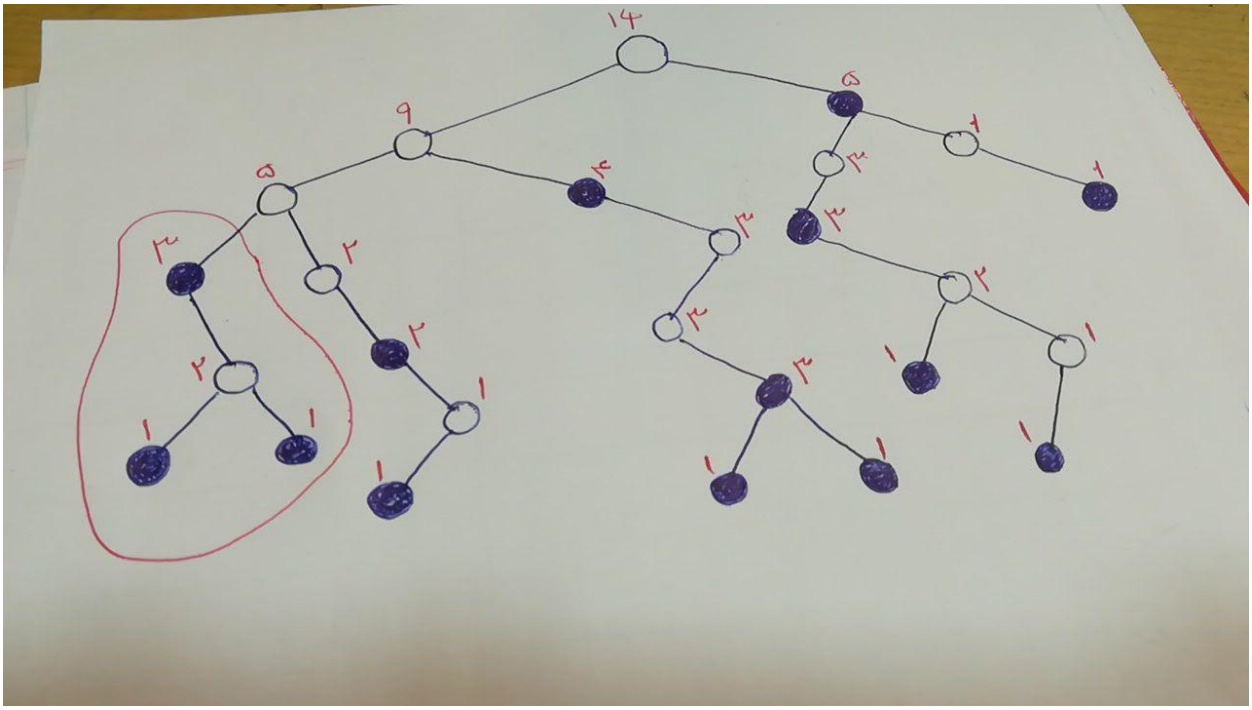
۴– از جمله روش‌های کاهش توان مصرفی توسط TCAM **روش** Trie-Based Table Partitioning **است. روش‌های** Subtree Splitting **و** Post-order Splitting **در این دسته قرار می‌گیرند. با اعمال این دو روش بر روی درخت زیر جدولی مشابه با جدول رسم شده در صفحه ۶۴ کتاب رسم کنید. گره های آبی رنگ، شامل آدرس‌های** prefix **معتبر هستند. فرض کنید** b=4 **است.**



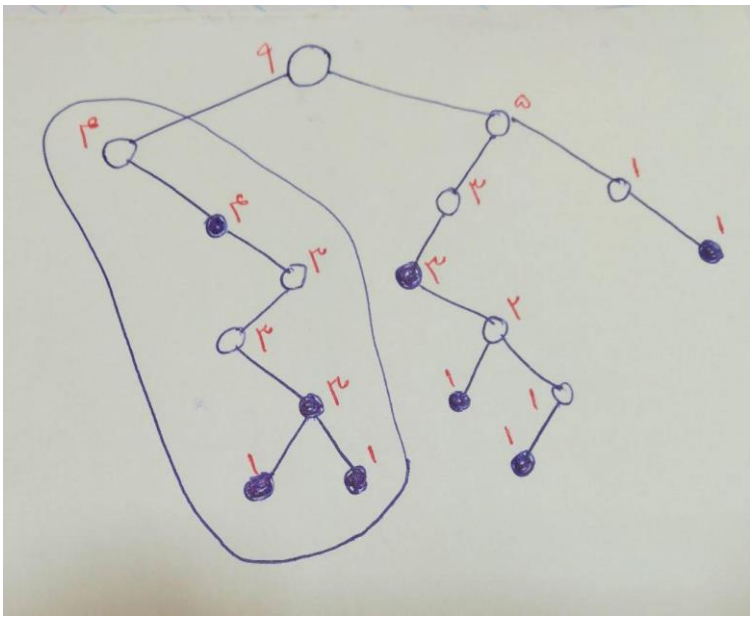
در بتدا باید مقدار هر گره را که به این صورت که prefix های موجود در زیر درخت با خود گره را شامل می شود مشخص کنیم که شکل زیر نشان داده شده است :



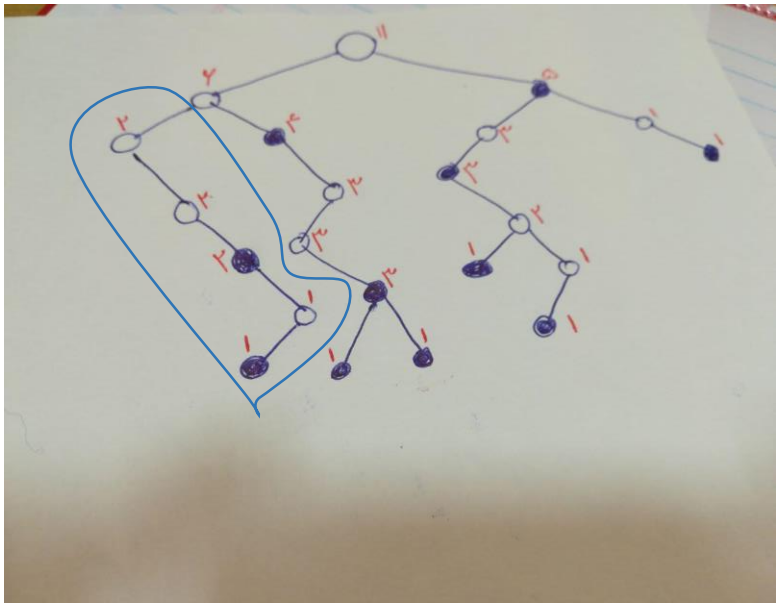
در روش subtree به صورت preorder پیمایش می کنیم و مقدار پیشوندی کوچکتر یا مساوی ۴ بود تمامی زیر درخت ها انتخاب می شود در شکل زیر نشان داده شده است:



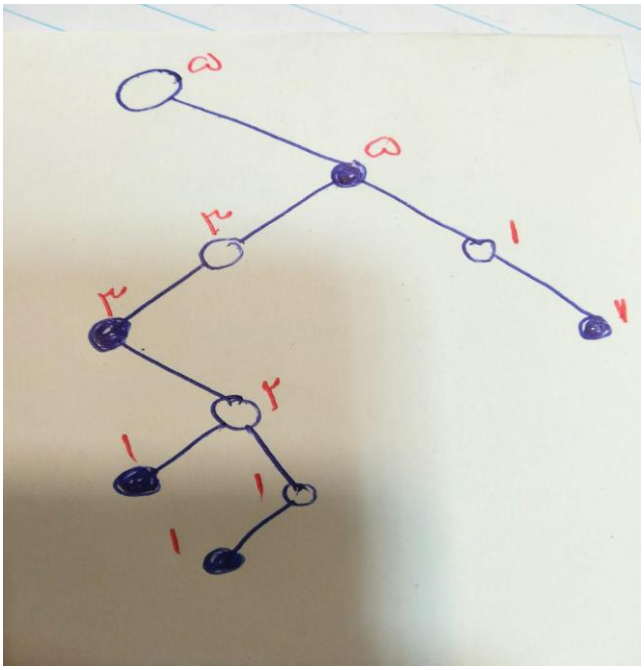
اکنون از مسیر پیشوند ۰۰۰* برای ایندکس این ۳ prefix (۰۰۰* و ۰۰۰۱۰* و ۰۰۰۱۱*) استفاده می شود و به همین ترتیب برای بقیه درخت ادامه می دهیم :



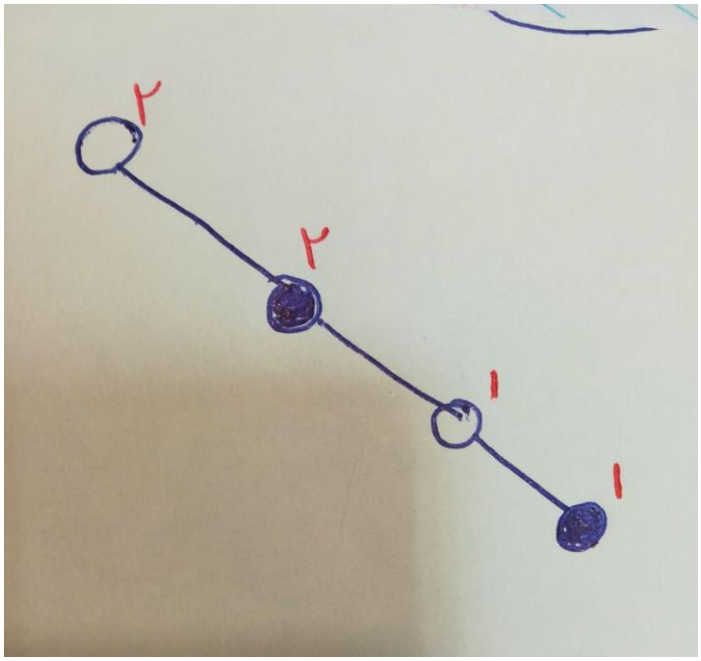
۳



۲



۴



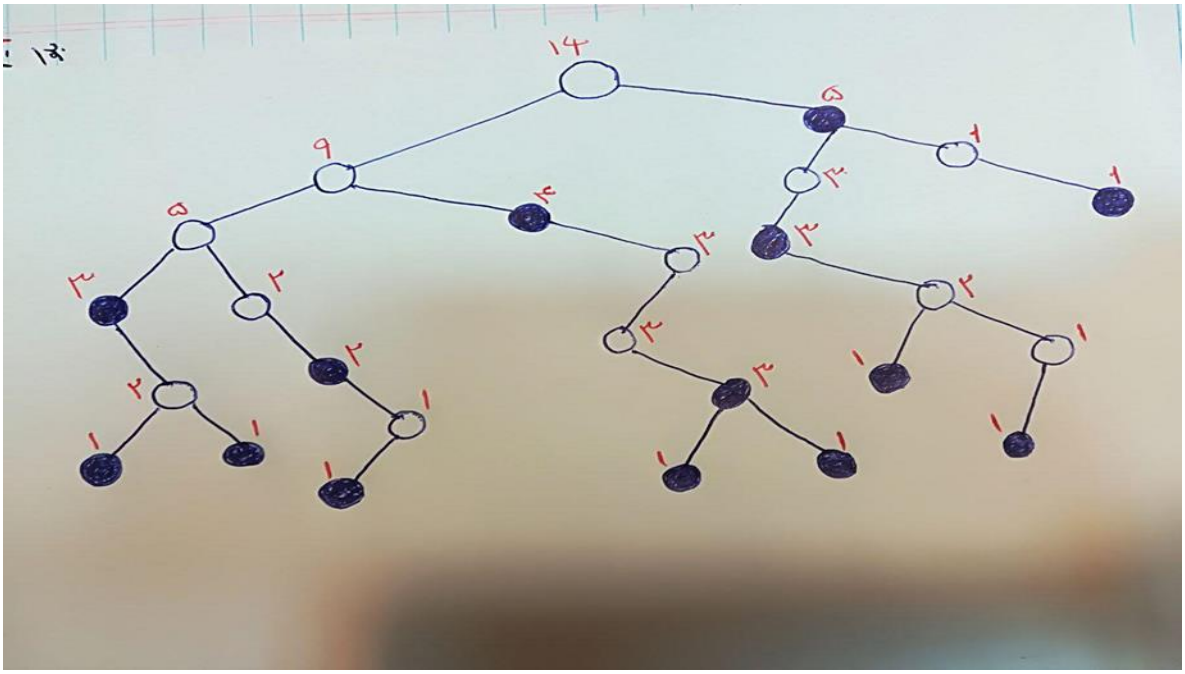
۵

در نهایت جدول زیر در روش subtree بدست می آید

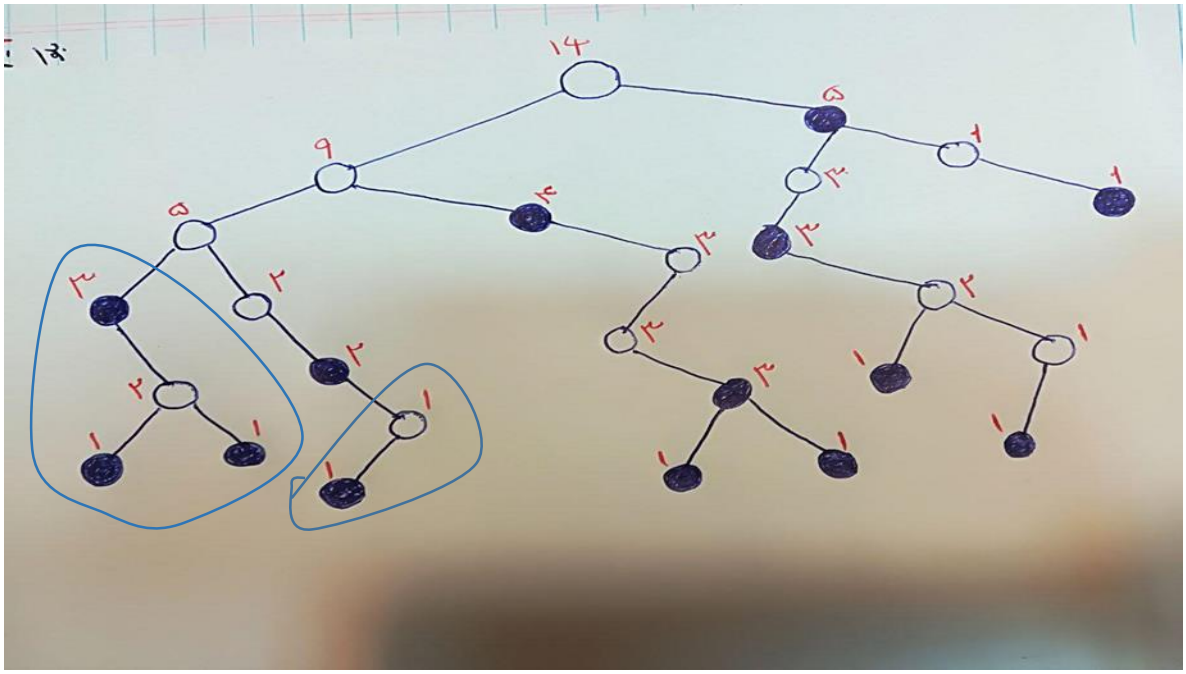
index	Bucket-prefix	Bucket-size	Convering prifix
000*	000* 00010* 00011*	3	000*
00*	0011* 001110*	2	*
0*	01* 01101* 011010* 011011*	4	*
10*	100* 10010* 100110*	3	1*
*	1* 111*	2	-

روش post order :

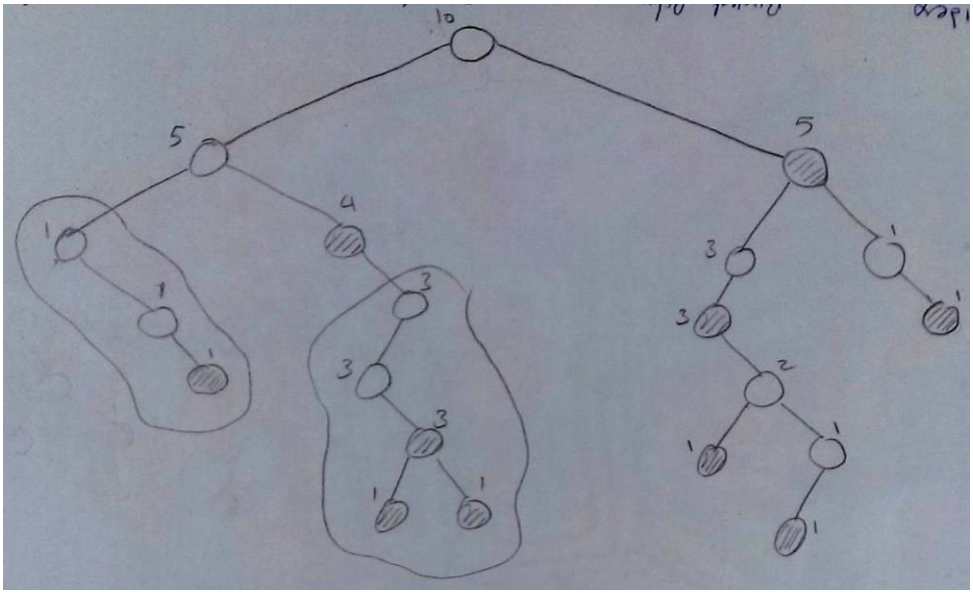
باید در ابتدا مقدار هر گره را مشخص کنیم، مقدار هر گره برابر است با تعداد prefix های موجود در زر درخت یک گره که خود آن گره را هم شامل می شود.

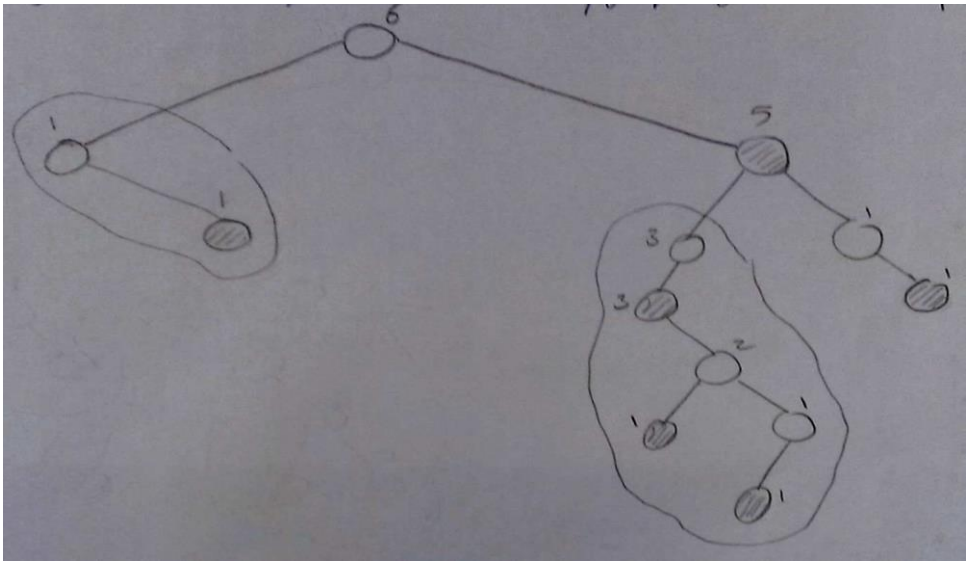


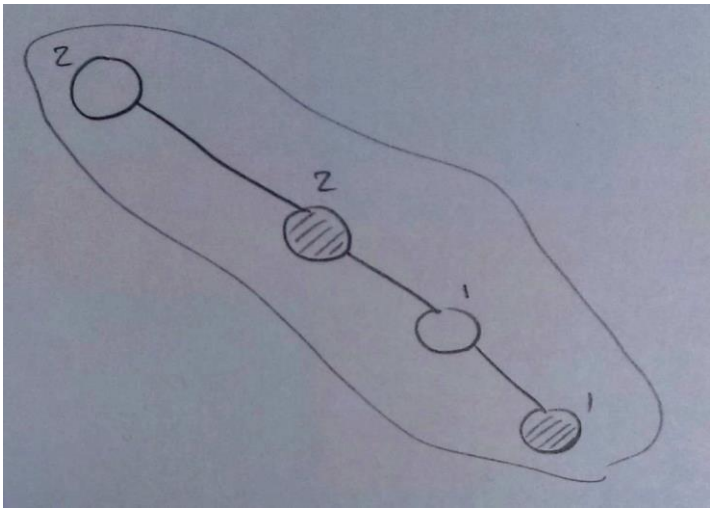
در این روش مجموع مقدار پیشوند هایی که پیمایش می کنیم دقیقا به اندازه سایز بلوک شد b=4 آن زیر شاخه را جدا می کنیم به همین دلیل اگر زیر شاخه ای با ارزش ۴ نداشته باشیم پیمایش را باید ادامه دهیم تا مجموع ارزش برابر ۴ شود.



در اینجا ما دارای ۲ ایندکس یعنی *۰۰۰ و *۰۰۱۱۱ برای ۴ پیشوند (*۰۰۰ و *۰۰۰۱۰ و *۰۰۰۱۱ و *۰۱۱۰۱۰)هستیم.







جدول بدست آمده از post order

index	Bucket prefix	Bucket size	Covering prefix
000* 00111*	000* 00010* 00011* 001110*	4	000* 0011*
00* 011*	0011* 01101* 011011* 011010*	4	* 01*
0* 10*	01* 100* 10010* 10010*	4	* 1*
*		4	*

۵– مقاله ضمیمه با عنوان "The Future of Switching in Data Centers" را مطالعه کنید و به سوالات زیر پاسخ دهید.

هدف نویسنده از مقاله فوق به همراه دست آوردهای مقاله را در دو پاراگراف بنویسید.

به طور کلی میتوان هدف مقاله را نفوذ تکنولوژی های optical-base در دیتا سنتر ها دانست همانطور که اتصال های point-to-point برای سالها به منظور اتصال مستقیم سرور ها و سوئیچ ها استفاده میشد باید برای اتصالات داخلی سوئیچ هم عملیات را به صورت optical بر مبنای یکی از معماری های active ، passiveیا hybrid انجام دهیم.

- ✓ بهینه کردن معماری داخلی سوئیچ هیای نوری با اتکا کردن به معماری hybrid که انعطاف پذیر و قادر به سازگاری با موقعیت های مختلف ترافیکی میباشد .
- ✓ ترکیب کردن انتقال نوری و سوئیچینگ نوری به صورت بهینه برای پی بردن به این مسئله که اتصالات داخلی سوئیچ ها با ظرفیت بالا به صورت نوری یک رویکرد امیدوار کننده برای رسیدن به سیستم ها با کارایی بالاست.

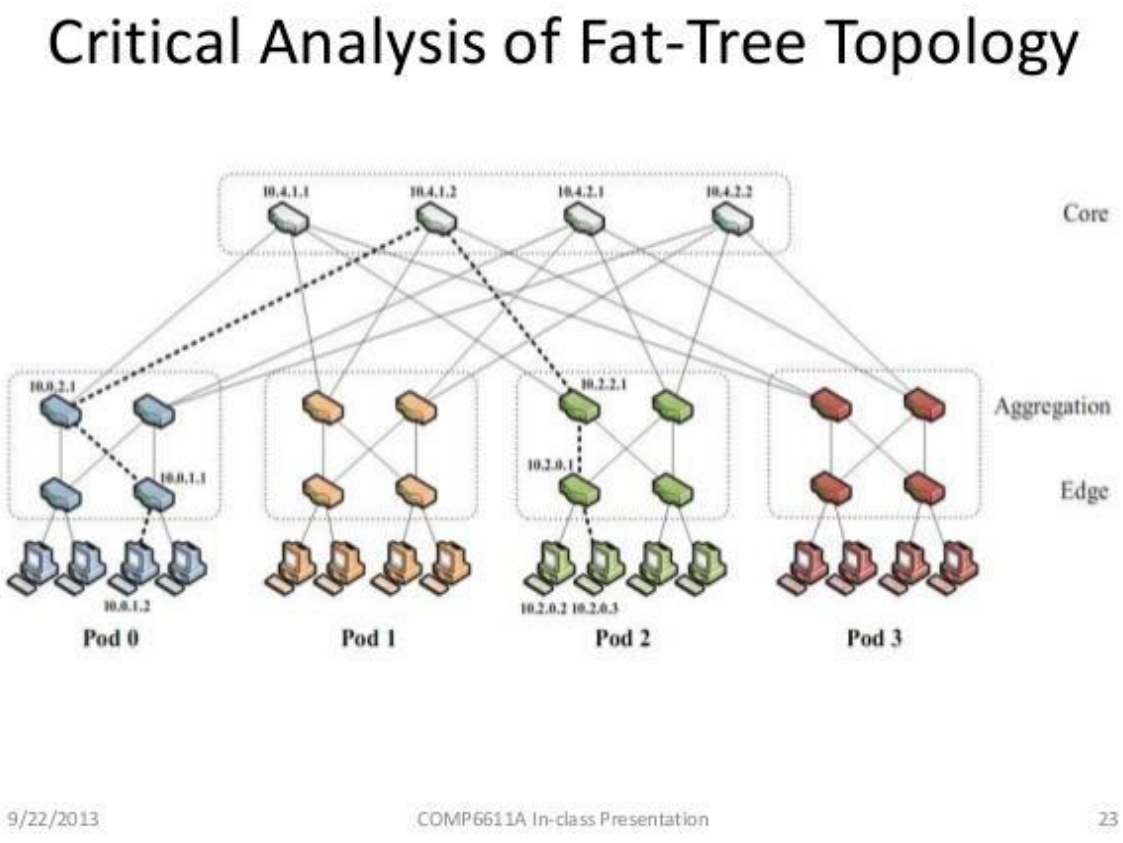
معماری شبکه مراکز داده را می توان به دو دسته Switch Centric و Server Centric دسته بندی کرد. از هر دسته یک معماری را نام برده، رسم کنید و مزایا و معایب هر یک را با یکدیگر مقایسه کنید.

در معماری switch centric ، routing intelligenceبر روی سوئیچ ها قرار داده میشود، یک سرور هم در این معماری معمولا از یک پورت NIC استفاده میکند و در کارهای مربوط به forwarding دخالتی نمیکنند.در معماری server centric سوئیچ ها فقط به عنوان crossbar استفاده میشوند و routing intelligence بر روی سرور ها قرتر داده میشوند و دارای چندین پورت NIC هستند و هم به عنوان نود های computing و هم نود های forwarding استفاده میشوند.

معماری برای: switch-centric

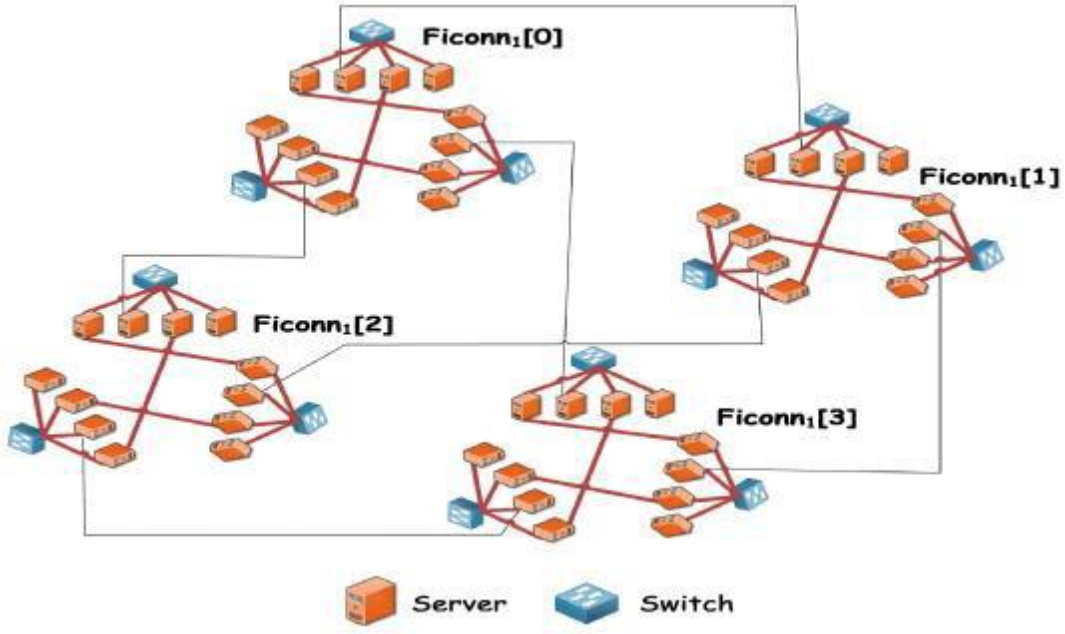
Fat-Tree , Flattened Butterflyکه معماری fat-tree را مشاهده میکنید

مزیت وعیب: مزیت آن قابلیت سریع سوئیچینگ سوئیچ ها میباشد اما سوئیچ ها قابلیت برنامه ریزی کمتری دارند.



معماری برای: server-centric

BCube, FiConn, Dcell که در زیر شکل معماری FiConn را برای نمونه اورده ام.



مزیت و عیب : قابلیت برنامه ریزی سرور ها بیشتر است اما سرور ها معمولا تاخیر processing بیشتری نسبت به سوئیچ ها دارا هستند.

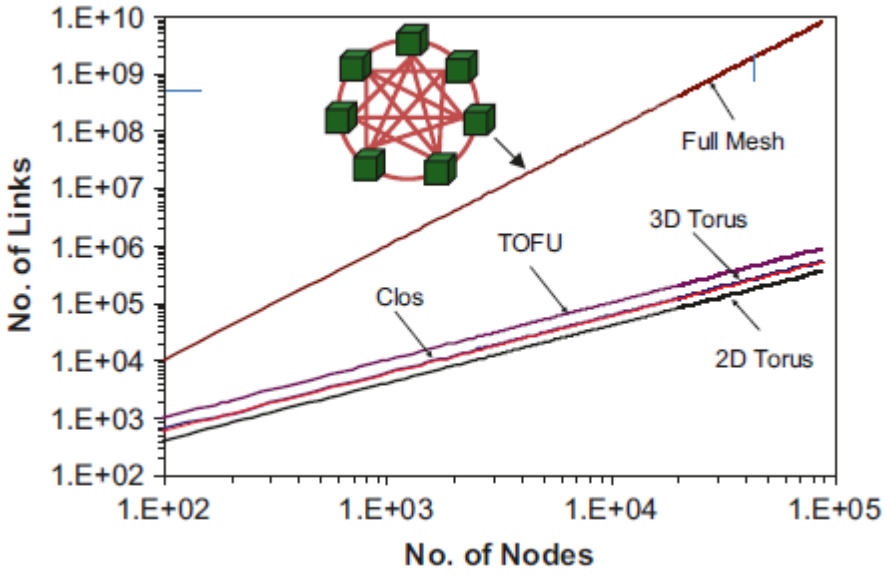
نویسنده ترافیک داخلی مراکز داده را به چهار دسته تقسیم کرده است. این چهار دسته را با یکدیگر مقایسه کنید.

اتصالات داخلی دیتا سنتر های بزرگ ، سوپر کامپیوتر ها یا روترها معمولا به ٤ گروه کلاسبندی میشوند که بتوان به عنوان سطوح مختلف سیستم سلسله مراتبی دیده شود،بالاترین سطح سلسله مراتبی اتصال rack-to-rack را بیان میکند که طول لینک های آن میتواند از چند متر تا چند صد متر شود، در داخل تجهیزات رک طول لینک intra-rackمعمولا بین ۱۵ cmتا چند متر میباشد،اتصالات chip-to-chip که بین چیپ ها در یک ماژول روی یک برد که طول این اتصال کمتر از ۱۵ cmمیباشد و در نهایت اتصالات on chip که کمتر از ۲cm میباشد.

wiring problemرا شرح دهید. توضیح دهید که کدام یک از همبندی های مطرح شده در این مقاله مقیاس پذیری بهتری

یک مسئله ی مهم که هم مقیاس پذیری و هم مدیریت سیستم های مقیاس بزرگ را محدود میکند تعداد زیاد لینک های اتصالات داخلی است که نتیجه ی آن تعداد زیاد کابل هاست ، به این موضوع Wiring Problem گویند بنابراین تعداد لیتک های مورد نیاز یک پارامتر مهم برای انتخاب توپولوژی مناسب شبکه است.

طبق نمودار نشان داده شده در مقاله که در زیر نیز ضمیمه شده است:



از بین توپولوژی های full mesh ، TOFU، Clos، 3D Torusو ۲ Torus Dکه در شکل نشان داده شده است ، توپولوژی ۲ Torus Dبازای تقریبا ۸۰۰۰۰ نود نیازمند تقریبا ٤۰۰۰۰۰ لینک میباشد که در بین توپولوژی های نشان داده شده در نمودار کمترین wiring را نیازمند است و از مابقی توپولوژی ها مقیاس پذیر تر است. اما در متن مقاله ذکر شده که TOFUدارای مقیاس پذیری بسیار خوبی است اما همچنان نیازمند نیازمند ۹٦۰۰۰۰ لینک بازای ۸۰۰۰۰ نود میباشد.

معماری های Cantor و Banyan را با یکدیگر مقایسه کنید.

طبق جدول زیر که در مقاله آورده شده است

Switch architectures	Blocking type
Classical logN	Blocking
Benes	Rearrangeably non-blocking
Crossbar	Wide-sense non-blocking
Spanke	Strict-sense non-blocking
Cantor	Strict-sense non-blocking
Banyan	Blocking

معماری Cantor دارای blocking از نوع Strict-Sense non-blocking میباشد در حالی که معماری سوئیچ Banyanاز نوع Blocking میباشد حال در ادامه مقاله جدول دیگری ارائه شده که با توجه به نوع بلاکینگ توپولوژی مورد استفاده را بیان میکند که با

استفاده از آن میتوانیم تفاوت های دیگری نیز بیان کنیم.

Network topologies	Blocking type
Clos (fat-tree, multistage)	Strict-sense, non-blocking if $p \geq 2n$-1
d-dim symmetric mesh	Rearrangeable, blocking if $p > 2$
d-dim symmetric torus	Rearrangeable, blocking if $p > 2$
d-dim hypercube	Rearrangeable

p is the number of edge switches
n is the number of ports of a single switching element

طبق این جدول بلاکینگ از نوع Strict-Sense مربوط به توپولوژی fat-tree یا Clos می شود. انتخاب نوع توپولوژی تاثیرات برجسته ای بر روی کارایی سیستم دارد بعنوان مثال fat-tree قادر است که performance بسیار خوبی با توجه به پهنای باند و تاخیر تامین کند اما به طور نسبی قیمت بالای کل شبکه به علت تعداد زیاد پورت های سرعت بالا مقیاس پذیری این توپولوژی را محدود میکند ، از طرف دیگر توپولوژی های meshو d-dim Torus که عموما دارای Blocking هستند اما دارای بهره وری هزینه در مقیاس بزرگ هستند به خصوص در کاربرد های محلی که در مورد این توپولوژی کارایی بالاتری نسبت به Clos را تامین میکند.