

الگوریتمهای مسیریابی

اصول عملکرد

روترها از الگوریتمهای مسیریابی، برای یافتن بهترین مسیر تا مقصد استفاده مینمایند هنگامی که ما در مورد بهترین مسیر صحبت میکنیم، پارامترهایی همانند تعداد hopها (مسیری که يك بسته از يك روتر دیگر در شبکه منتقل میشود)، زمان تغییر و هزینه ارتباطی ارسال بسته را در نظر میگیریم.

مبتنی بر اینکه روترها چگونه اطلاعاتی در مورد ساختار يك شبکه جمع آوری مینمایند و نیز تحلیل آنها از اطلاعات برای تعیین بهترین مسیر، ما دو الگوریتم مسیریابی اصلی را در اختیار داریم: الگوریتم مسیریابی عمومی و الگوریتمهای مسیریابی غیر متمرکز.

در الگوریتمهای مسیریابی غیر متمرکز، هر روتر اطلاعاتی در مورد روترهایی که مستقیماً به آنها متصل میباشند در اختیار دارد. در این روش هر روتر در مورد همه روترهای موجود در شبکه، اطلاعات در اختیار ندارد. این الگوریتمها تحت نام الگوریتمهای (DV distance vector) معروف هستند. در الگوریتمهای مسیریابی عمومی، هر روتر اطلاعات کاملی در مورد همه روترهای دیگر شبکه و نیز وضعیت ترافیک شبکه در اختیار دارد. این الگوریتمها تحت نام الگوریتمهای LS(Link state) معروف هستند. ما در ادامه مقاله به بررسی الگوریتمهای LS میپردازیم.

الگوریتمهای LS

در الگوریتمهای LS، هر روتر میبایست مراحل ذیل را به انجام رساند:

روترهای را که به لحاظ فیزیکی به آنها متصل میباشد را شناسایی نموده و هنگامی که شروع به کار میکند آدرسهای IP آنها بدست آورد. این روتر ابتدا يك بسته HELLO را روی شبکه ارسال میکند. هر روتری که این بسته را دریافت میکند از طریق يك پیام که دارای آدرس IP خود این روتر میباشد به پیام HELLO پاسخ میدهد.

زمان تاخیر مربوط به روترهای مجاور را اندازه گیری نماید(یا هر پارامتر مهم دیگری از شبکه همانند ترافیک متوسط)

برای انجام این کار، روترها بسته های echo را روی شبکه ارسال میکنند. هر روتری که این بسته ها را دریافت میکند با یک بسته echo reply به آن پاسخ میدهد. با تقسیم زمان مسیر رفت و برگشت به دو، روترها میتوانند زمان تاخیر را محاسبه کنند.(زمان مسیر رفت و برگشت، سنجشی از تاخیر فعلی روی یک شبکه میباشد) توجه داشته باشید که این زمان شامل زمانهای ارسال و پردازش میباشد.

اطلاعات خود را در مورد شبکه، برای استفاده سایر روترها منتشر نموده و اطلاعات روترهای دیگر را دریافت کند.

در این مرحله همه روترها دانش خود را با روترهای دیگر به اشتراک گذاشته و اطلاعات مربوط به شبکه را با یکدیگر مبادله میکنند. با این روش هر روتر میتواند در مورد ساختار و وضعیت شبکه اطلاعات کافی بدست آورد.

با استفاده از این الگوریتم مناسب، بهترین مسیر بین هر دو گره از شبکه را شناسایی کند.

در این مرحله، روترها بهترین مسیر تا هر گره را انتخاب میکنند. آنها این کار را با استفاده از یک الگوریتم همانند الگوریتم کوتاهترین مسیر Dijkstra انجام میدهند. در این الگوریتم، یک روتر مبتنی بر اطلاعاتی که از سایر روترها جمع آوری نموده است، گرافی از شبکه را ایجاد مینماید. این گراف مکان روترهای موجود در شبکه و نقاط پیوند آنها را به یکدیگر نشان میدهد. هر پیوند با یک شماره به نام Cost یا weight مشخص میشود. این شماره تابعی از زمان تاخیر، متوسط ترافیک و گاهی اوقات تعداد hopهای بین گره ها میباشد. برای مثال اگر دو پیوند بین یک گره و مقصد وجود داشته باشد، روتر پیوندی با کمترین Weight را انتخاب میکند.

الگوریتم Dijkstra دارای مراحل ذیل میباشد:

روتر گرافی از شبکه را ایجاد نموده و گره های منبع و مقصد (برای مثال V_1 و V_2) را شناسایی میکند. سپس یک ماتریس به نام ماتریس adjacency را میسازد. در این ماتریس یک مختصه مبین Weight میباشد. برای مثال $[i, j]$ ، وزن یک پیوند بین V_i و V_j میباشد. در صورتی که هیچ پیوند مستقیمی بین V_i و V_j وجود نداشته باشد این وزن (ویت) بصورت infinity در نظر گرفته میشود.

روتر يك مجموعه رکورد وضعیت را براي هر گره روي شبکه ایجاد مینماید این رکورد داراي سه فیلد میباشد:

فیلد Predecessor: اولین فیلدي که گره قبلي را نشان میدهد.

فیلد Length: فیلد دوم که جمع وزنهای از منبع تا آن گره را نشان میدهد.

فیلد Label: آخرین فیلد که وضعیت گره را نشان میدهد. هر گره میتواند داراي يك مود وضعیت باشد: tentative یا permanent

روتر، پارامترهای مجموعه رکورد وضعیت براي همه گره ها را آماده سازي اولیه نموده و طول آنها را در حالت infinity و Label آن را در وضعیت tentative قرار میدهد.

روتر، يك گره T را ایجاد میکند. براي مثال اگر V1 میبایست گره T منبع باشد، روتر برچسب V1 را در وضعیت permanent قرار میدهد. هنگامی که يك Label به حالت permanent تغییر میکند دیگر هرگز تغییر نخواهد کرد. يك گره T در واقع يك agent میباشد.

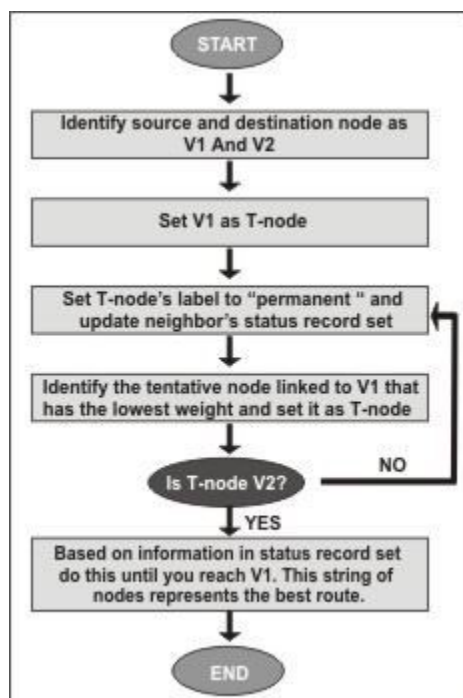
روتر، مجموع رکورد وضعیت مربوط به همه گره های Tentative را که مستقیما به گره T منبع متصل هستند، روز آمد مینماید.

روتر همه گره های Tentative را بررسی نموده و گره های را که وزن آن تا V1 کمترین مقدار را دارد انتخاب میکند. سپس این گره، گره T مقصد خواهد بود

اگر این گره، V2 نباشد (گره مقصد) روتر به مرحله 5 باز میگردد.

اگر این گره V2 باشد، روتر گره قبلي آن را از مجموع رکورد وضعیت استخراج نموده و این کار را انجام میدهد تا به V1 برسد، این فرست از گره ها، بهترین مسیر از V1 تا V2 را نشان میدهد.

این مراحل بصورت يك فلوچارت در شکل نشان داده شده است ما از این الگوریتم بعنوان يك مثال در ادامه مقاله استفاده خواهیم نمود.

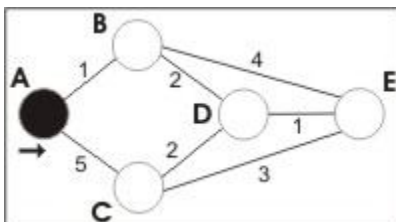


مثال

الگوریتم *Dijkstra*

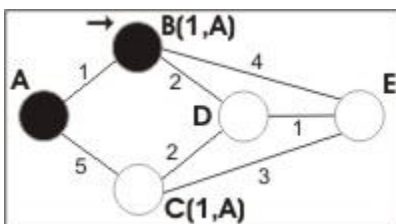
در اینجا ما می‌خواهیم بهترین مسیر بین گره‌های A و E را پیدا کنیم همانطور که می‌بینید 6 مسیر بین A و E وجود دارد. (ABE، ACE، ABDE، ACDE، ABDCE، ACDBE) واضح است که ABDE بهترین مسیر می‌باشد زیرا کمترین وزن را دارد اما همیشه به این سادگی نیست و برخی موارد پیچیده وجود دارد که در آن ما مجبوریم از الگوریتم‌هایی برای یافتن بهترین مسیر استفاده کنیم.

همانطور که در تصویر ذیل مشاهده می‌کنید، گره منبع (A) بعنوان گره T انتخاب شده و بنابراین برچسب آن، Permanent می‌باشد. (ما گره‌های Permanent را با دایره‌های تو پر و گره‌های T را با یک پیکان نشان می‌دهیم)



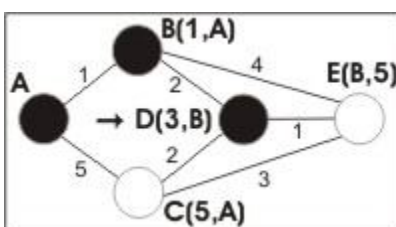
در این مرحله شما می‌بینید که مجموع رکورد وضعیت گره‌های Tentative که مستقیماً به گره (C) T، متصل شده‌اند، تغییر یافته است. همچنین از آنجایی که گره B کمترین

وزن را دارد، بعنوان گره T انتخاب شده و برچسب آن به حالت Permanent تغییر کرده است.

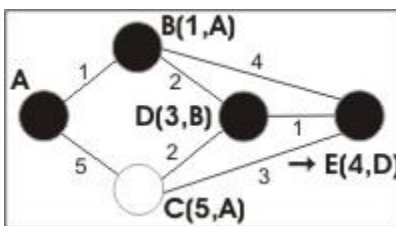


در این مرحله همانند مرحله قبل دو مجموعه رکورد وضعیت گره هایی که Tentative دارای اتصال مستقیم به گره T می باشد (D,E) تغییر کرده است. همچنین از آنجایی که گره D وزن کمتری دارد، بعنوان گره T انتخاب شده و برچسب آن به وضعیت Permanent تغییر کرده است.

در این مرحله ما هیچ گره Tentative نداریم بنابراین فقط گره T بعدی را شناسایی میکنیم. از آنجایی که E دارای کمترین وزن می باشد بعنوان گره T انتخاب میشود.



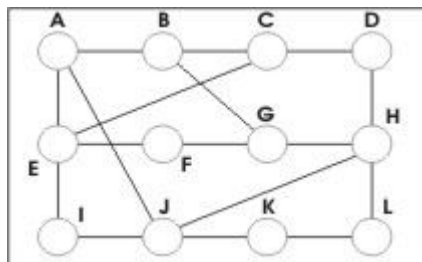
E گره مقصد بوده بنابر این کار ما در اینجا تمام میشود. اکنون ما کار شناسایی مسیر را به انتها رسانده ایم. گره قبلی E گره D، گره B می باشد و گره قبلی B، گره A می باشد. بنابر این بهترین مسیر ABDE است در این مورد وزن کل مسیر، $4(1+2+1)$ می باشد.



با وجودی که این الگوریتم بخوبی کار میکند اما آنقدر پیچیده است که زمان پردازش آن برای روتر طولانی بوده و راندمان شبکه را کاهش میدهد. همچنین اگر يك روتر اطلاعات غلطی را به روترهای دیگر بدهد، همه تصمیمات مسیر یابی نادرست خواهد بود.

الگوریتمهای DV

الگوریتمهای DV با نامهای الگوریتمهای مسیریابی Bellman-Ford و ford-fulkerson نیز یاد میشوند. در این الگوریتمها، هر روتر دارای یک جدول مسیریابی میباشد که بهترین مسیر تا هر مقصد را نشان میدهد. یک گراف معمولی و جدول مسیریابی مربوط به روتر G در شکل زیر نشان داده شده است.



همانطور که در جدول مشاهده میکنید، اگر روتر G بخواهد بسته هایی را به روتر D ارسال کند، میبایست آنها را به روتر H ارسال نماید. هنگامی که بسته ها به روتر H رسیدند، این روتر جدول خود را بررسی نموده و روی چگونگی ارسال بسته ها به D تصمیم گیری می کند.

Line	Weight	Destination
A	8	A
A	20	B
I	28	C
H	20	D
I	17	E
I	30	F
H	18	G
H	12	H
I	10	I
-	0	J
K	6	K
K	15	L

در الگوریتم‌های DV، هر روتر میبایست مراحل ذیل را انجام دهد:

وزن لینک‌های مستقیما متصل به آن را اندازه گرفته و این اطلاعات را در جدول خود ذخیره کند.

در يك دوره زماني خاص، روتر جدول خود را به روترهاي مجاور ارسال نموده و جدول مسيريابي هر يك از روترهاي مجاور خود را دريافت ميكند.

مبتني بر اطلاعات بدست آمده از جداول مسيريابي روتريهاي مجاور، جدول خود را روز آمدسازي مينمايد.

يکي از مهمترين مشكلات، هنگام کار با الگوريتمهاي DV، مشکل Count to infinity اجازه بدهيد اين مشکل را با ذکر يك مثال روشن کنيم.

همانطور که در قسمت ذيل نشان داده شده است يك شبکه را در ذهن خود تصور کنيد. همانطور که در اين جدول مي بينيد، فقط يك پيوند بين A و ساير بخشهاي شبکه وجود دارد. در اينجا شما ميتوانيد، اين گراف و جدول مسيريابي همه گره ها را مشاهده کنيد:

D	C	B	A	
3,D	2,B	1,A	0,-	A
3,D	2,C	0,-	1,B	B
1,C	0,-	1,C	2,B	C
0,-	1,D	2,C	3,B	D

اکنون تصور کنيد که پيوند بين A و B قطع شود. در اين هنگام، B جدول خود را تصحيح ميکند بعد از يك مدت زمان خاص، روتريها جداول خود را مبادله نموده و بنابر اين B جدول مسيريابي C را دريافت ميکند. از آنجايي که C نميداند چه اتفاقي براي پيوند بين A و B رخ داده است اين اطلاعات را حفظ ميکند. B اين جدول را دريافت نموده و فکر ميکند که يك پيوند جداگانه بين C و A وجود دارد، بنابر اين جدول خود را تصحيح نموده مقدار infinity را به 3 تغيير ميدهد. به همين شکل دوباره روتريها جداول خود را مبادله ميکنند. هنگامي که C، جدول مسيريابي B را دريافت ميکند، مشاهده ميکند که B وزن پيوند خود تا A را از 1 به 3 تغيير داده است، بنابر اين C، جدول خود را روزآمد نموده و وزن پيوند خود تا A را به 4 تغيير ميدهد. اين پروسه تکرار ميشود تا همه گره ها وزن پيوند خود را تا A در وضعيت infinity قرار دهند. اين وضعيت در جدول زير نشان داده شده است.

D	C	B	
3,C	2,B	∞ ,A	Sum of weight to A after link cut
3,C	2,B	3,C	Sum of weight to B after 1st updating
3,C	4,B	3,C	Sum of weight to A after 2nd updating
5,C	4,B	5,C	Sum of weight to A after 3rd updating
5,C	6,B	5,C	Sum of weight to A after 4th updating
7,C	6,B	7,C	Sum of weight to A after 5th updating

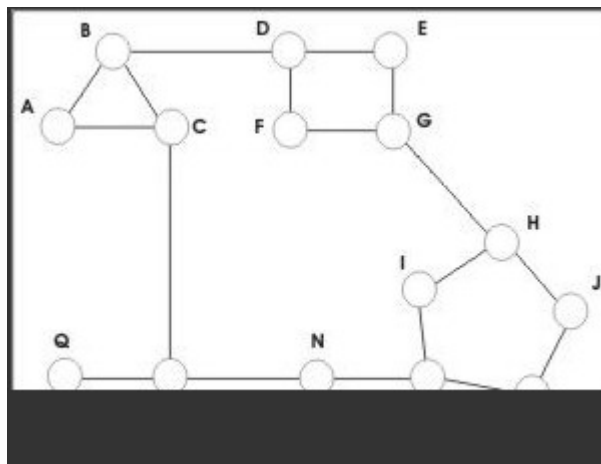
در اين روش متخصصين ميگويند، الگوريتمهاي DV داراي يك سرعت همگرابي پايين هستند. يك روش براي حل اين مشکل در مورد روتريها، ارسال اطلاعات فقط به روتريهاي

میباشد که دارای پیوند انحصاری تا مقصد نیستند. برای مثال در این مورد، C نمیباشد هیچ اطلاعاتی را به گره B در مورد A ارسال کند زیرا B فقط یک مسیر تا A را در اختیار دارد.

مسیریابی سلسله مراتبی

همانطور که شما میبینید، در هر دو الگوریتم LS و DV، هر روتر مجبور به ذخیره نمودن اطلاعات مربوط به روترهای دیگر میباشد. هنگامی که اندازه شبکه رشد میکند، تعداد روترهای شبکه افزایش می یابد در نتیجه اندازه جداول مسیریابی نیز افزایش می یابد و روترها نمیتوانند ترافیک شبکه را به طور موثر کنترل کنند. ما از مسیریابی سلسله مراتبی برای برطرف کردن این مشکل استفاده میکنیم. اجازه بدهید این موضوع با ذکر یک مثال روشن کنیم:

ما از الگوریتمهای DV برای یافتن بهترین مسیر بین گره ها استفاده میکنیم در وضعیت نشان داده شده در ذیل، هر گره از شبکه مجبور به نگهداری یک جدول مسیریابی با 17 رکورد میباشد. در اینجا یک گراف معمولی و جدول مسیریابی مربوط به A ارائه شده است.

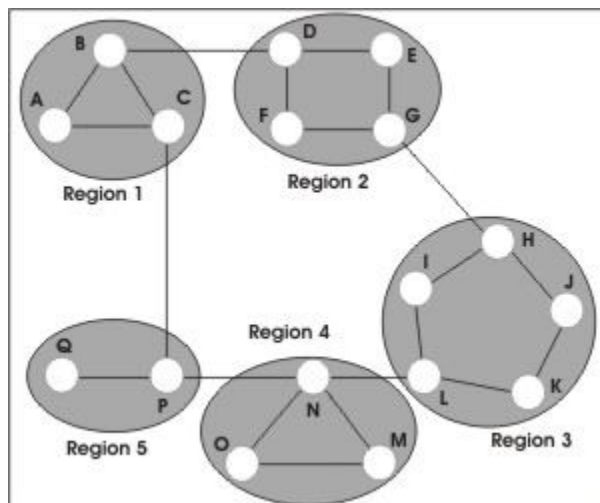


Weight	Line	Destination
-	-	A
1	B	B
1	C	C
2	B	D
3	B	E
3	B	F
4	B	G
5	B	H
5	C	I

6	C	J
5	C	K
4	C	L
4	C	M
3	C	N
4	C	O
2	C	P
3	C	Q

در مسیریابی سلسله مراتبی، روترها در گروههایی به نام regions طبقه بندی میشوند. هر روتر دارای اطلاعاتی فقط در مورد روترهایی که در region آنها قرار دارد در اختیار داشته و هیچ گونه اطلاعاتی در مورد region های دیگر ندارند.

در این مثال ما شبکه خود را به پنج region تقسیم میکنیم. اگر A بخواهد بسته ها را به هر روتر در region2 ارسال کند، آنها را به B ارسال میکند و الی آخر.



Weight	Line	Destination
-	-	A
1	B	B
1	C	C
2	B	Region 2
2	C	Region 3
3	C	Region 4
4	C	Region 5

در این نوع مسیریابی، جداول را میتوان خلاصه نمود بنابراین راندمان شبکه بهبود مییابد. مثال بالا مسیریابی سلسله مراتبی دو سطحی را نشان میدهد همچنین میتوان از مسیریابی سلسله مراتبی 3 سطحی و 4 سطحی استفاده کرد. در

مسیریابی سلسله مراتبی 3 سطحی، شبکه به تعدادی کلاستر تقسیم بندی میشود. هر کلاستر متشکل از تعدادی region و هر region دارای تعدادی روتر میباشد. مسیریابی سلسله مراتبی به طور وسیعی در مسیریابی اینترنت مورد استفاده قرار میگیرد و استفاده از چندین پروتکل مسیریابی را ممکن می سازد.

برگرفته از سایت : www.PooyeshR.com