网络流算法的分析及研究

胡金初
(上海师范大学计算机系 上海 200234)

摘 要 为了改善网络的性能,提出了一种网络流的分析方法,流量控制与某个发送方和接收方之间的点到点的通信量控制有关,即解决网络中出现快发慢收的问题,还受到通信线路容量的限制。网络的流量问题是一个网络传输中的重要问题,这里给出的算法可以解决实际问题。

关键词 网络流,算法,标记,容许流

Analysis and Study of Network Flow Algorithms

HU Jin-chu
(Department of Computer Science, Shanghai Normal University, Shanghai 200234, China)

Abstract In order to improve the performance of network, this paper proposed a method for analyzing network flow. Flow control is only referring to point to point data flow control, which ensures that the sender do not send data packets in a speed faster than the receiver. One of limit is that not enough capacity of line is available to transfer the data packets. The network flow is an important problem in netwoks transmission. The example algorithms given is useful in analyzing actual network systems.

Keywords Network flow, Algorithms, Mark, Allow flow

1 引言

近年来,随着信息技术的迅猛发展,网络应用大量增加,使得原来已经存在的庞大数据传输量成倍地增长。在Internet上大约95%左右的数据流使用的是TCP/IP协议。为了确保Internet稳定和通信流畅,必须对网络的流量进行分析和控制。

在协议中,数据链路层、网络层、传输层和流量控制的策略都有关,采用重发策略时,定义多长的时间为超时,在帧超时后如何进行重发。在处理乱序策略时,如果简单地将所有乱序的帧丢弃,那么就意味着这些帧必须全部重传,又会造成系统沉重的负担。确认策略的不同也会影响到拥塞问题,如果每个帧都要立即确认,确认帧将会造成额外的通信量。但如果采取确认保存方法,以便让别的数据帧捎带返回;这虽然可以减少确认帧的数量,但又会萌发超时帧的增加,从而又需要重传。而好的流量控制方案能降低传输速率,减少拥塞的发生。子网内的虚电路和数据报的选择会对网络产生影响。分组丢弃策略说明当没有剩余空间时,哪些分组应该被丢弃。路由选择策略能通过将通信量分散到所有线路上,而一个不好的路由选择算法会将通信量发送到本来已经发生拥挤的线路上。分组生命期管理决定了一个分组能够在网络中存活多长的时间,定得太长,会使分组长时间滞留在网络中;反之定得过短,大量分组又可能在到达目的地之前就已超时,导致不必要的重传。要确定数据在网络上传输的时间比较困难。如果定义的时间间隔太短,会产生不必要的额外分组;如果过长,拥塞会减少,但分组一旦丢失后,系统的反应时间就会很长。

2 网络流问题

网络的流量问题是一个网络传输中的重要问题,图1表示了一个典型的计算机网络,信号源S以固定的速度产生数据,然后通过图中的传输网将数据送至目的地T。为了获得更好的性能,信号源希望尽可能多地把数据包传输到T,但是每条通信线路的负载能力是有限的,并且也不是发送者能够改变的,因此用户只能在现有的通信线路网中找出可以传输最多数据的运送方案,下面分析和解决这一网络流问题。

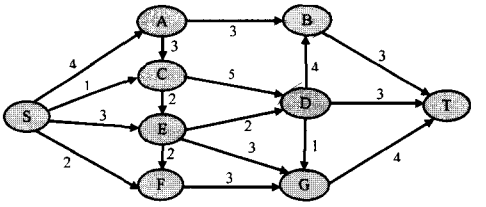

图1 一个典型的通信网

网络$G=(V,E,C)$是由顶点集V、边集E和每条边对应的容量集C组成的一个有向连通图,且满足:

- 只有一个顶点源,其入度为0,记为Vs;
- 只有一个顶点宿,其出度为0,记为Vt;
- $\forall(V_i,V_j)\in E,\exists c_{ij}\geqslant 0$称之为边$c_{ij}$的容量。

网络流F是指定义在边集E上的一个函数$F=\{f_{ij}\}$,为边(V_i,V_j)的流量,网络的容许流$V(F)$,指在网络$G=(V,E,C)$中,$\forall(V_i,V_j)\in E,\exists V(F)=\{f_{ij}\}$,且$V(F)$应满足:

- 容量限制条件 $0\leqslant f_{ij}\leqslant c_{ij}$。
- 顶点平衡条件 流入某一顶点的流量等于流出该顶点的流量。
- 源、宿平衡条件 $\sum_j f_{sj}=\sum_j f_{jt}$。

若令$f_{ij}=0$,则称其为零流。

在网络$G=(V,E,C)$中,对边有下列分类:

- 饱和边 $f_{ij}=c_{ij}$,非饱和边 $f_{ij}<c_{ij}$。

到稿日期:2008-04-29

·零流边 $f_{ij}=0$，非零流边 $f_{ij}>0$。

令 $S,i_1,i_2,\cdots,i_k,T$ 是一条从 V_s 到 V_t 的路径 P_s，则有：

·前向边：(V_i,V_{i+1})，与路径方向一致，记作 $P+$。

·后向边：(V_i,V_{i-1})，与路径方向相反，记作 $P-$。

设 $V(F)$ 是网络中的容许流，P_s 是一条从 V_s 到 V_t 的路径，若 P_s 满足下列条件：

$\forall(V_i,V_j)\in P_s^+$ 有 $0\leqslant f_{ij}<c_{ij}$，即每条前向边都是非饱和边。

$\forall(V_i,V_j)\in P_s^-$ 有 $0<f_{ij}\leqslant c_{ij}$，即每条后向边都是非零流边。

则称 P_s 存在着一条增流路径，而且增流路径有且仅有以下两种情况：

·路径 P_s 中的每条边都是前向边。

·路径 P_s 中既有前向边也有后向边。

由于存在着增流路径 P_s，因此可调整路径 P_s 上的容许流 $V(F)$，其方法是：

对于前一种情况：引入增量 Δ，并且使 $\Delta=\min\limits_{(V_i,V_j)\in P_s}(c_{ij}-f_{ij})$，则新的容许 $V(F)'=V(F)+\Delta$。

对于后一种情况：令 $(V_i,V_j)\in P_s^+$ 的增量为 $\Delta_1=\min\limits_{(V_i,V_j)\in P_s^+}(c_{ij}-f_{ij})$，令 $(V_i,V_j)\in P_s^-$ 的可改进量为 $\Delta_2=\min\limits_{(V_i,V_j)\in P_s^-}(f_{ij})$，于是有下列增流规则：

当 $(V_i,V_j)\in P_s^+$ 时，有 $f_{ij}'=f_{ij}+\Delta_1$，当 $(V_i,V_j)\in P_s^-$ 时，有 $f_{ij}'=f_{ij}-\Delta_2$。

通过对上面网络中最大流的分析，当 $V(F)$ 是网络中的容许流，若不存在从 V_s 到 V_t 的关于 $V(F)$ 的增流路径，那么 $V(F)$ 就是网络的最大流。

设 S 是网络 $G=(V,E,C)$ 中的一个顶点集，且 S 满足下列条件：

·$V_s\in S$。

·$V_t\in\overline{S}$，其中 $\overline{S}=V-S$，记 $\overline{S}$ 为 T。

若 $\forall(V_i,V_j)\in E$，有 $V_i\in S$ 且 $V_j\in T$，则由此组成的边集为网络 G 的一个分离 V_s 和 V_t 的截集，记作 (S,T)；并将 $C(S,T)=\sum\limits_{(V_i,V_j)\in(S,T)}c_{ij}$ 称为截集的容量，即截量。

分离 V_s 和 V_t 的截集实质上是由 (S,T) 所构成的一个边集。任取截集 $(S,T)=(\{S,D,G\},\{A,B,C,E,F,T\})=\{(S,A),(S,B),(S,C),(G,T)\}$，可以看出若在网络中丢弃此截集，则 V_s 到 V_t 之间就不可能存在通路。因此，截集 (S,T) 是从 V_s 到 V_t 的必经之路。

3 网络流的算法介绍

网络流的计算机算法是：从网络的一个容许流 $V(F)$ 开始，初始时 $V(F)$ 设置为零，经过下列两种过程的交替循环来计算网络最大流。

① 标记过程，寻找网络中是否存在关于 $V(F)$ 的增流路径。

对网络中的顶点进行标记，其标记的格式为（±前驱指针，增量 Δ），± 表示前/后向边，前驱指针表示当前顶点是从哪个顶点得到的标记，可增量 $\Delta=L(V_j)$。

在标记过程中顶点分为二类：已标记点（包括已检查的点和未检查的点）、未标记点。

初始时，对网络中的源顶点 V_s 标记为 $(-1,+\infty)$，其中 -1 表示该顶点没有前驱指针。V_s 就成为标记而未检查的顶点，网络中的其余顶点都是未标记顶点。然后从这个标记未检查顶点出发，深度优先搜索网络中所有未标记顶点 V_j，并执行以下操作：

1）若在边 $(V_i,V_j)\in P_s^+$ 上有 $f_{ij}<c_{ij}$，则对 V_j 标记 $(V_i,L(V_j))$，其中 $L(V_j)=\min\{L(V_j),c_{ij}-f_{ij}\}$；

2）若在边 $(V_i,V_j)\in P_s^-$ 上有 $f_{ij}>0$，则对 V_j 标记 $(-V_i,L(V_j))$，其中 $L(V_j)=\min\{L(V_j),f_{ji}\}$。

此时，对顶点 V_j 标记为未检查顶点，实现时采用压栈操作，当 V_i 的所有相邻顶点都已标记之后，V_j 标记就已检查完毕。若无法对顶点进行标记，则出栈。

重复上述步骤，直至顶点宿 V_t 也被标记，说明已找到一条增流路径 P_s，接着执行下一个步骤增流过程。如果标记过程无法继续（栈空），这说明操作已完成，并且已经找到了最大流。则将最后一次标记过程中得到标记的顶点归入 S，而其余顶点则归入 T，这样就能构成网络 G 的最小截集 (S,T) 从而可以验证所得的网络容许流 $V(F)$ 就是所求的网络最大流，即 $V(F)=\min(S,T)=\max\{V(F)\}$。

② 增流过程，这一步是通过回溯来进一步调整 P_s，并得到网络新的容许流 $V(F')$。

根据标记中的第一个元素（前驱指针）可以回溯 P_s，根据标记的第二个元素（增量）修改 P_s 上每条边的流量，规则如下：

$$V(F')=f_{ij}'=\begin{cases}f_{ij}-\Delta & (V_i,V_j)\in P_s^+\ (V_{i-1},\Delta)\\ f_{ij}-\Delta & (V_i,V_j)\in P_s^-\ (-V_{i+1},\Delta)\\ f_{ij} & (V_i,V_j)\in P_s\end{cases}$$

然后，清除所有顶点上的标记，转入上面的标记过程再处理。算法流程描述如下：

(1)在给定的网络中任选一个容许流 $V(F)$，在初始状况下，可从零流开始。

(2)开始标记过程，给 S 标记(−1,maxflow)。

(3)如果存在一个未标记顶点 V_i，它可以通过正向（或反向）对它进行标记，并转步骤 4 执行，否则转步骤 7。

(4)如果出现 $V_i=T$，就转步骤 5，否则转步骤 3。

(5)开始增流过程，从 T 出发，根据各相关顶点的前驱指针回溯整条增流路径，并按前（后）向边分别修改路径所覆盖的边的流量。若回溯到源 S，转步骤 6。

(6)清除所有顶点的标记，然后转步骤 2 执行。

(7)程序执行到这一步时，已找到网络的最大流，程序结束。

网络流量的分析，采用的方法是对有向连通图的遍历，常用的两种方法是 DFS 和 BFS，这两种方法各有所长。在求解网络最大流时，采用不同的方法遍历网络会直接影响到算法的性能。采用 DFS 来遍历网络，可以使得从源出发的网络流总是试图以最快的速度到达宿，但在某些情况下，采用 DFS 的算法会使效率下降，例如出现图 2 的情况。

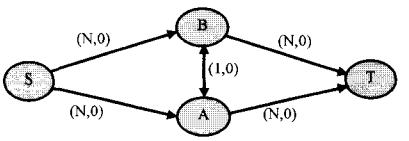

图 2 网络中的小增量回路

开始选中了图中的增流路径 $P_1=S\to A\to B\to T(\Delta_{P_1}=1)$ 和 $P_2=S\to B\to A\to T(\Delta_{P_2}=1)$，此时程序将不断在这两条路径上交替循环，这就使得要找增流路径的总数达到了 $2*N$ 条，是实际情况的 N 倍。在这种情况下，该算法的复杂度

已经与网络中顶点集 V 和边集 E 没有关系了，却受到了网络参数(图中的 N)的直接影响。

改进的算法是每次都沿一条最短的增流路径进行增流，最短是指路径所包含的边数最少。若采用先标记先检查，对网络进行广度优先搜索，就能达到这个目的。具体方法如下：

对上面的第 3 步改进为按 BFS 次序先标记先检查，选择最先标记但尚未检查的顶点，若该顶点的所有邻接顶点均已标记，则转第 7 步，否则对所有未标记的邻接顶点进行标记，然后转第 4 步。

4 算法的实现和分析

以上面的网络为例，在给定的网络中从源顶点 S 出发，根据通信链路的容量，选容许流 $V(F)$，该相关顶点进行标记，得到图 3 的结果。

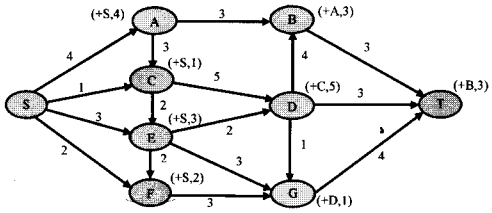

图 3 第一次标记

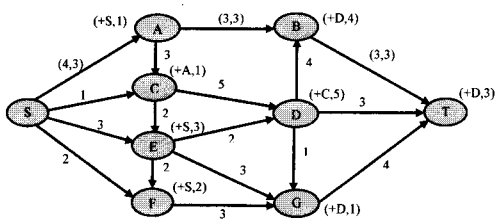

图 4 第二次标记

显然存在有一条增流路径 $P_1=S\to A\to B\to T(\Delta_{P_1}=3)$。在相应的通信链路上标记容许流 $V(F)$，在完成增流路径的操作以后，再进行第二次标记，得到图 4 的结果。

又发现有增流路径，继续操作。然后再作标记，调整相应的通信链路上标记容许流 $V(F)$，直到整个网络拓扑图搜索完成，得到图 5 的结果，解决了网络流的问题。

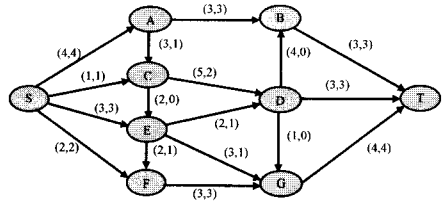

图 5 标记完成后的网络流

下面对该算法的复杂性进行分析，该算法每次都需要寻找网络中最短的增流路径，它可能只有一条边，但是最长时可能达到 $|V|-1$ 条边。若将增流路径所含边的总数称为长度，那么网络中就可能存在长度为 $1,2,\cdots,|V|-1$ 的增流路径。使用 BFS 搜索网络是，寻找一条增流路径在最坏情况下需要遍历网络中所有的边 $O(|E|)$。为了找到所有长度的边，在最坏情况下算法须执行 $O(|V||E|)$。同时，每次调整过程中修改一条增流路径时，在最坏情况下算法的复杂度为 $O(|E|)$。因此，整个算法的复杂度就是 $O(|V||E|^2)$，算法可以用 Java 实现。

结束语 网络流量问题的讨论在实际生活中也有意义，许多系统中都包含了流量问题，例如，公路系统中的车流量，控制系统中有信息流，供水系统中有水流量，金融系统中有现金流等。该方法具有一定的普遍性，也可以应用到这类系统中去，这些由“源”产生的流，经过复杂的中转网络，到达目的地“宿”，为了获得尽可能好的性能，可以转化为最大流的分析。

参 考 文 献

[1] Andrew S. Tanenbaum, Computer Network, Fourth Edition. Pearson Education North Asia Limied, 2004

[2] 胡金初. 计算机网络. 高等教育出版社, 2006

(上接第 292 页)

表 4

a2	Y	Z	P(out2≠12)	P(out2=12)
T	T	T	0	1
	T	F	$1-\Delta$	Δ
	F	T	$1-\Delta$	Δ
	F	F	$1-12\cdot\Delta^2$	$12\cdot\Delta^2$
	…	…	$1-\Delta$	Δ

至此，贝叶斯网络已经构建完成。由于该贝叶斯网络中有环存在，利用网址[4]提供的 Bayesian Network 工具箱中联合树算法引擎进行推理计算，获得各元件故障概率分别如下：

$P(m_1)=0.6806, P(m_2)=0.0492, P(m_3)=0.0079$

$P(a_1)=0.2903, P(a_2)=0.0022$

上述结果与文献[2,3]进行比较可发现元件 m_2 的故障概率下降了约 50%，元件 a_2 的故障概率下降了约 25%，而这正是因为条件概率的引入使得两个故障叠加后系统行为正常的概率下降，与我们之前的分析一致。

结束语 本文讨论了利用贝叶斯网络进行模型诊断的方法。文献[1]提出过采用贝叶斯网络进行模型故障诊断的思路，但未能结合团树算法(Junction Tree Algorithm)求出元件故障概率，本文给出了精确的概率值，这是本文的最大创新点。后续工作将考虑采用信息熵作为判据，结合贝叶斯网络进行故障诊断。

参 考 文 献

[1] Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. 2nd ed. San Mateo, CA: Morgan Kaufmann, 1991

[2] Kohlas J, Anrig R, Haenni R. Model-based diagnosis and probabilistic assumption-based reasoning[J]. Artificial Intelligence, 1998, 104(1): 71-106

[3] 邓勇, 等. 基于模型的贝叶斯诊断及应用[J]. 上海交通大学学报, 2003, 30(1): 5-8

[4] http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html

[5] Retiter R. A theory of diagnosis from first principles[J]. Artificial Intelligence, 1987, 32(1): 57-95