doi:10.11959/j.issn.1000-436x.2017228

基于时间聚合图的 DTN 网络最短时延路由算法

王鹏，李红艳，张焘，李朋云
(西安电子科技大学综合业务网理论及关键技术国家重点实验室，陕西 西安 710071)

摘 要：DTN(delay tolerant network)网络时变特征导致静态网络的路由算法无法求解该网络中给定业务的快速传输问题。时变路由算法 CGR (contact graph routing)利用链路最早连通时段获得最短路径，由于连通时段先后顺序影响导致其算法链路利用率低下。针对该问题，对端到端最短路径的路由方法进行了研究，在时间聚合图中增加节点缓存时间序列表征同一链路不同时间段之间的联系，采用深度优先搜索从目的点向源节点反向找路，求解出已知业务需求端到端最短时延算法。用样例证明了算法的可行性。

关键词：DTN 网络；时间聚合图；最短路径；CGR 算法
中图分类号：TN915.03 **文献标识码**：A

Minimum delay algorithm based on time aggregated graph in DTN network

WANG Peng, LI Hong-yan, ZHANG Tao, LI Peng-yun
(State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China)

Abstract: The DTN network has the characteristics of dynamic topology change,thus the routing algorithm of static network can not solve the routing problem of time-varying network, and can not support fast transmission of given tasks. The existing time-variant routing algorithm CGR (contact graph routing) uses the earliest contact to obtain the shortest path, but because of the influence of the order of the connectivity period, the CGR algorithm has low link utilization. To solve this problem, the method of finding the shortest path was proposed. Besides, in order to characterize the relation between the different connect periods of the same link, the time series of node cache was added to the time aggregated graph. Based on this, the end-to-end multi-path shortest delay routing algorithm for the known task requirements was proposed. Finally, the feasibility of the algorithm was proved by an example.

Key words: DTN network, time aggregated graph, optimal path，CGR algorithm

1 引言

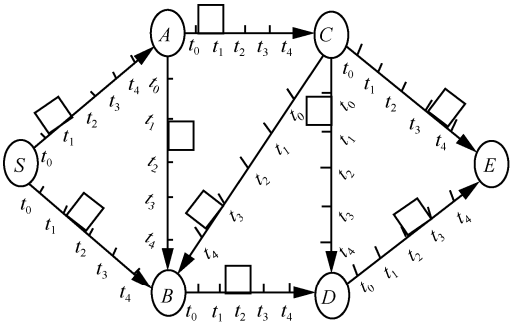
当前，DTN 网络被广泛应用于互联网[1]、传感器网络[2]、车联网[3]以及近地卫星网络[4]等具有时变特性的网络，该网络具有链路断续连通、长时延和节点缓存资源有限的特点[5]。由于网络的时变性、动态性，导致传统的静态网络路由算法无法直接应用于 DTN 网络。DTN 网络中业务 QoS 保障一直是研究的热点，但由于缺乏精确的图模型表征，导致最短时延求解困难，因此，迫切需要研究适应于时变环境的最短时延路由算法，以保障 DTN 网络中通信业务高效、快速传输。

收稿日期：2017-09-12
基金项目：国家重点研究发展计划基金资助项目（No.2016YFB0501004）；国家自然科学基金资助项目（No.91638202, No.91338115, No.61231008, No.61401326, No.61571351）；国家科技重大专项基金资助项目（No.2015ZX03002006）；中央高校基础研究经费基金资助项目（No.WRYB142208, No.JB140117）；111 计划基金资助项目（No.B08038）；陕西省自然科学基础研究计划基金资助项目（No.2016JQ6054）
Foundation Items: The National Key Research and Development Program of China (No.2016YFB0501004),The National Natural Science Foundation of China (No.91638202, No.91338115, No.61231008, No.61401326, No.61571351), National S&T Major Project (No.2015ZX03002006), The Fundamental Research Funds for the Central Universities (No.WRYB142208, No.JB140117), 111 Project (No.B08038), Natural Science Basic Research Plan in Shaanxi Province (No.2016JQ6054)

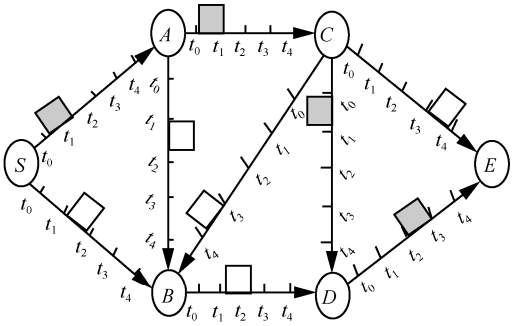
为了深入地研究 DTN 网络的时变特点，首先需构建精确的网络模型。快照模型将 DTN 网络每个时段的拓扑表征成一个静态子图[6]，但由于快照割裂了链路不同时间段之间的关系，导致运用快照模型不能够高效地利用网络中的链路资源。为了克服快照的缺陷，时间扩展图模型[7]在不同快照的同一节点处增加了存储链路，有效地表征出 DTN 网络时变拓扑的联系，图模型直观但是存储量大，计算复杂。时间聚合图[7]将同一链路不同时间段的特性聚合成链路时间序列，大大降低了网络的复杂度。但是，由于现有的时间聚合图缺乏同一链路不同时间段相互关系的表征，导致选路的先后顺序影响链路的占有情况，造成网络资源浪费。

在路由算法方面，由于 DTN 网络具有拓扑时变、链路断续连通、长时延的特征，导致静态网络中经典最短路径算法（如 Dijkstra 算法等）无法应用于 DTN 网络。现有的时变网络最短路径的算法如 SP-TAG 等算法[7]，借鉴了 Dijkstra 算法中松弛的思想[8]，能够求解出最短路径。但是，由于该算法采用的时间聚合图模型并没有考虑链路的不同时间段的业务量，当给定业务量求解最短路径时，SP-TAG 算法无法计算路由。此外，CGR 算法基于 DTN 网络中最早的连通链路，给出了针对业务需求的最短时延路由方法[9]。但是，当单一路径无法满足业务传输需求时，CGR 算法存在明显的链路资源浪费的缺陷。如图 1 所示，图 1(a)描述了一个由 6 个节点组成的 DTN 网络，其中，链路上的每个块表示该链路能传输 1 个单位数据。假定源节点 *S* 需向目的节点 *E* 传输 2 个单位数据，根据 CGR 的路由策略，CGR 选择的最短路径为(*S*-*A*-*C*-*D*-*E*)，如图 1(b)所示，其中，所有灰色方块表示沿该路径传输的数据（1 个单位数据）。除了路径(*S*-*A*-*C*-*D*-*E*)外，图 1(b)中没有其他可选路径，CGR 不能完成传输任务，同时剩余的链路资源无法被利用。然而，如图 1(c)所示，利用路径 1(*S*-*A*-*C*-*E*)和路径 2(*S*-*B*-*D*-*E*)可以完成 2 个单位数据的传输。

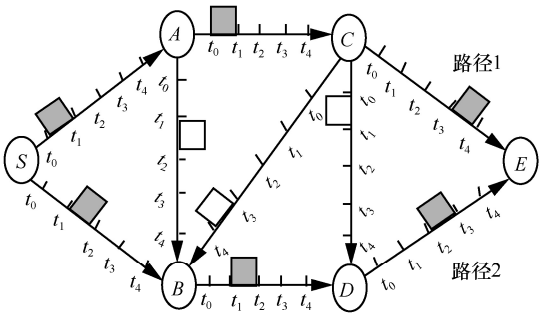
针对图 1 模型表征不精确和 CGR 算法中链路资源浪费的问题，本文利用时间聚合图中链路容量序列的容量与时间信息，提出最短路径算法，利用逆序找路的方法（从目的节点到源节点找路）将松弛找路方式转换为路径深度优先搜索，能够快速地找到最短路径；此外，在现有的时间聚合图中增加节点缓存时间序列，有效地表征同一链路不同时间段之间的联系，结合最短路径算法进一步提出多路径最短时延算法，该算法支持对已选择路径的修



(a)网络模型



(b)CGR选路结果



(c)理想的选路结果

图 1　CGR 路由结果

正，提高了网络资源的利用率，克服了 CGR 路由算法中的不足。

2　系统模型

本文主要针对拓扑、链路等网络信息可预测的 DTN 网络（如卫星网络）。假定预测的时间范围为 $[0,T]$，将 $[0,T]$ 时间段分割为 n 个连续的小的时间段 $[\delta_1,\cdots,\delta_m,\cdots,\delta_n]$，其中，时间段 $\delta_m=[t_m^{\text{start}},t_m^{\text{end}}]$，且 $t_m^{\text{start}}<t_m^{\text{end}}$。网络的节点集合记为 V，边集合记为 E，对于 $\forall i,j\in V$，若存在 $ij\in E$，则链路 ij 为一条有向链路，其中，节点 i 为前向节点，节点 j 为后向节点。

存储时间聚合图在静态有向图的基础上，通过将不同时间段的链路带宽和节点存储状况用序列的方式标注在图上，刻画了网络在一个时间段的时变状态。

基于此，存储时间聚合图(STAG, storage time aggregated graph)定义为 $STAG=\{V,E,W_{ij}^T,S_i^T,T\mid ij\in E,i\in V,j\in V\}$，其中，$W_{ij}^T=\{w_{ij}^{\delta_1},\cdots,w_{ij}^{\delta_m},\cdots,w_{ij}^{\delta_n}\}$，$S_i^T=\{s_i^{\delta_1},\cdots,s_i^{\delta_m},\cdots,s_i^{\delta_n}\}$，$w_{ij}^{\delta_m}=\int_{t_m^{\text{start}}}^{t_m^{\text{end}}}B_{ij}^{\delta_m}\mathrm{d}t$ 表示链路 ij 在 $[t_m^{\text{start}},t_m^{\text{end}}]$ 内可以传输的数据量；$B_{ij}^{\delta_m}$ 表示有向边 ij 在时间段 $[t_m^{\text{start}},t_m^{\text{end}}]$ 内传输数据的带宽，$s_i^{\delta_m}$ 表示节点 i 在时段 $\delta_m=[t_m^{\text{start}},t_m^{\text{end}}]$ 内所缓存的数据量。此外，残余网络定义为 $rSTAG=\{V,E,rE,W_{ij}^T,rW_{pq}^T,S_i^T,T,\mid ij\in E,pq\in rE,i\in V,j\in V\}$，$rE$ 记录的是找路过程中与 $STAG$ 的 E 中不同方向的链路，$rW_{pq}^T=\{rw_{pq}^{\delta_1},rw_{pq}^{\delta_2},\cdots,rw_{pq}^{\delta_n}\mid pq\in rE\}$，记录 rE 中的链路的权重值。基于预测的信息，本文给出如图 2 所示的 $STAG$ 图模型实例。

给定的时间范围 T 被分成了 4 个连续的时段，链路上的容量时间序列对应每个时段所能传输的数据量，如链路 sa 的容量时间序列为 $(6,0,0,0)$，表明节点 s 只能在第一个时段向节点 a 发送 6 个数据。各节点的缓存时间序列初始为 $\{0,0,0,0\}$，对应 $[\delta_1,\delta_2,\delta_3,\delta_4]$ 这 4 个时间段内节点的缓存。其中，节点缓存的计算方式为

$$s_i^{\delta_m}=\begin{cases}f_i^+(\delta_m)-f_i^-(\delta_m),m=1\\ f_i^+(\delta_m)-f_i^-(\delta_m)+s_i^{\delta_{m-1}},1<m\leqslant 4\end{cases}\tag{1}$$

其中，$f_i^+(\delta_m)$ 和 $f_i^-(\delta_m)$ 分别表示在时段 δ_m 流入和流出节点 i 的数据。

3　多路径最短时延算法

3.1　问题描述

最短时延问题指给定源节点的业务需求，在从源节点到目的节点所有可行路径中，选取几条路径能够满足业务量传输且总时延最短。在求解多路径最短时延的过程中，路径选择需要满足以下约束条件。

流守恒。出入节点的数据量需要满足流守恒特性。

$$\sum_{m=1}^{n}\sum_{j\in V}f_{ji}^{\delta_m}-\sum_{m=1}^{n}\sum_{j\in V}f_{ji}^{\delta_m}=0,\ \forall i\in\{V-\{s,e\}\}\tag{2}$$

链路容量限制。每条链路的流大小不能超过链路的容量。

$$0\leqslant f_{ij}^{\delta_m}\leqslant w_{ij}^{\delta_m},\forall ij\in E\tag{3}$$

节点缓存限制。每个节点的缓存时间序列需要满足节点缓存的规律。

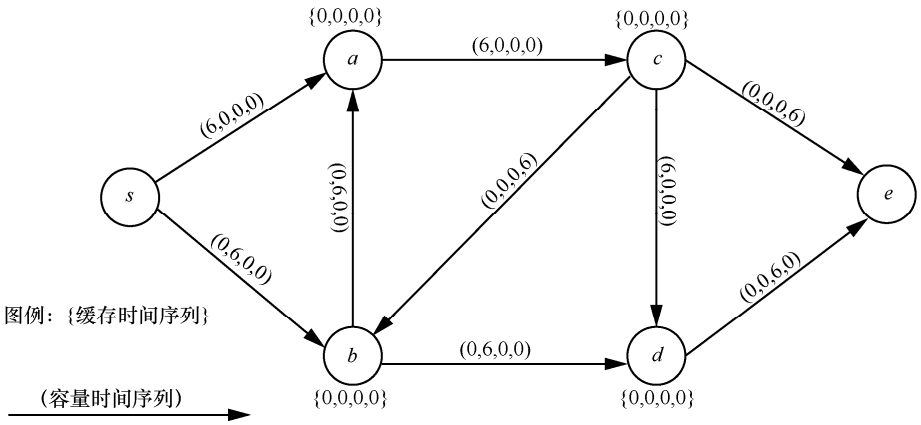


图 2　STAG 图模型

$$s_i^{\delta_m}=\begin{cases} f_i^+(\delta_m)-f_i^-(\delta_m), & m=1 \\ f_i^+(\delta_m)-f_i^-(\delta_m)+s_i^{\delta_{m-1}}, & 1<m\leqslant n \end{cases} \quad \forall i\in V \tag{4}$$

找路的时间限制。若找到 k 个数据的路径共需找 m 次，第 m 次所找路径最后一跳链路传输完数据的时间为 t_m，传完所有业务量需要的时间为 τ，$\tau=t_m-0$，算法完成时需要有 $\tau=\min(t_m-0)$。

业务量约束。假定给定业务量大小为 k，第 p 次找到的路径能传输的流的大小为 f^p，传输满足业务量的数据至少需要找 m 次路，则有

$$\sum_{p=1}^{m} f^p=f_{\text{sum}}\geqslant k \tag{5}$$

3.2 多路径最短时延算法

本文提出的多路径最短时延算法，利用时间聚合图从源节点到目的节点的路径中最后一跳链路能够存储路径时延信息的特点，先找到最后一跳链路，然后从这条链路出发，由目的节点向源节点找路，把复杂的松弛操作转换成简单的深度优先搜索找路，保障正确性和高效性。此外，把时间聚合图中的节点加上缓存时间序列，建立同一链路不同时间段之间的联系，在节点处选路时，会将不同时间段的链路都加入选择范围而不用受时间的限制，从而达到可以在选路过程中对已选路径进行修正的效果。所提出的多路径最短时延算法主要步骤包括：1) 求最短路径；2) 求解最短路径可行流；3) 更新残余网络。

算法 1 多路径最短时延算法

输入

1) $STAG=\{V,E,W_{ij}^T,S_i^T,T\,|\,ij\in E,i\in V,j\in V\}$；

2) $rSTAG=\{V,E,rE,W_{ij}^T,rW_{pq}^T,S_i^T,T,|\,ij\in E,pq\in rE,\ i\in V,j\in V\}$；

3) 业务量大小 k；

4) 源节点 s，目的节点 e，开始时间 $\delta_{\text{start}}=\delta_1$。

输出

1) $rSTAG$ 中 rE 中所有链路的反向链路；

2) $rSTAG$ 中 rW_{pq}^T 中的所有 rE 的边的权重序列。

步骤 1 记已找到所有最短路径的可行流为 f_{sum}，f_{sum} 初始时为 0；

步骤 2 将 $rSTAG$ 作为网络拓扑执行最短路径算法进行找路；若找不到路径，且此时 $f_{\text{sum}}<k$，则算法在网络中找不到可以传输 k 个数据的路由方案，算法结束，否则执行步骤 3；

步骤 3 计算已找路径 l 的可行流；

步骤 4 进行残余网络更新，更新 f_{sum}，若 $f_{\text{sum}}<k$，继续执行步骤 2，否则结束算法。

3.2.1 最短路径算法

算法 2 最短路径算法

输入

1) $rSTAG$，源节点 s，目的节点 e；

2) 对任意属于点集 V 的节点 u，节点 u 有 2 个属性，$u.p$ 表示节点 u 的父节点，即为路径 l 中指向 u 的节点；$Adj[u]$ 则表示在 $rSTAG$ 中 u 的邻节点中的前向节点。

输出 最短路径 l。

开始 find_optical_path($rSTAG,s,e$)

1) (v,e)=find_earliest_links($\{Adj[e]\},e$)

//在边集 E 中的目的节点 e 与其所有的前向节点组成的所有有向链路中，找到具有最早连通时间段的链路，$\{Adj[e]\}$ 表示 e 的所有前向节点构成的集合，存储在点集 E 和 rE 中

2) $e.p=v$；

3) δ_{find}=find_earliest_period(W_{ve}^T)； //找到链路 ve 中最早连通的时间段

4) loop:

5) if($s_v^{\delta_{\text{find}}}>0$ 且 $s_v^{\delta_{find}+1}>0$)

6) δ_{temp}=find_latest_peroid($S_v^T,\delta_{\text{find}}$)；

//找到节点 v 缓存时间序列中从 δ_{find} 开始非零缓存值最晚出现的时间

7) $\delta_{\text{find}}=\delta_{\text{temp}}$；

8) end if

9) 对任意 $u\in\{Adj[v]\}$

10) if $uv\in E$

11) δ_{temp}=find_earliest_period(W_{uv}^T)；

12) else

13) δ_{temp}=find_earliest_period(rW_{uv}^T)；

14) end if

15) if($\delta_{\text{temp}}<\delta_{\text{find}}$)

16) $v.p=u$；

17) $\delta_{\text{find}}=\delta_{\text{temp}}$；

18) if($u=s$)

19) 算法结束，输出路径 l；

20) else

21) $v=u$；

22)　　goto(loop)

23)　　end if

24)　end if

如算法 2 所示，最短路径算法旨在找到一条从目的节点通往源节点的路径，首先找到目的节点与前向节点构成的链路中最早连通的链路径，然后从该链路开始使用深度优先搜索算法找路。若存在一条以该链路为最后一跳链路的从源节点到目的节点的路径，则该路径必为最短路径。利用这种特点，将常见的迭代松弛找最短路径的策略转化为先找最短路径必经的最后一跳链路，然后用深度优先搜索就可以找到最短时延的路径。

此外，在找路过程中，利用节点的缓存时间序列，扩宽了找路时间。通过这种做法，可以更改已找路径，可以避免出现某条路径占用了网络资源不能修改的情况。

3.2.2　求解路径 l 的可行流

求解可行流即为求解最短时延内能够传输的数据量，不是求解最短路径的最大流，这种数据量应该是满足最短时延能够最短传输的数据，所以求解最短路径的可行流是求解在某个时刻之前最短路径上的所有链路中能够传输的最大的数据量。具体的求解路径 l 的可行流方法，如算法 3 所示。

算法 3　求解路径的可行流

输入　残余网络 $rSTAG$,最短路径 l ,源节点 s ，目的节点 e 。

输出　最短路径 l 的可行流 f^l 。

find_available_flow($rSTAG,l$)

1) $v=e.p$;

2) δ_{find} =find_earliest_period(W_{ve}^T);

3) $f^{\text{current}}=w_{ve}^{\delta_{\text{find}}}$;

4) $f^{\text{temp}}=\infty$;

5) $f^l=\infty$;

6) $u=v$;

7)while($u\neq s$)

8)　　$u=v.p$;

9)　　if($uv\in E$)

10)　　δ_{temp} = find_earliest_period(W_{uv}^T);

11)　　$f^{\text{next}}=w_{uv}^{\delta_{\text{temp}}}$;

12)　　else

13)　　δ_{temp} = find_earliest_period(rW_{uv}^T);

14)　　$f^{\text{next}}=rw_{uv}^{\delta_{\text{temp}}}$;

15)　　end if

16)　　if($\delta_{\text{temp}}>\delta_{\text{find}}$)

17)　　δ_{temp_1} =find_latest_period($S_v^T,\delta_{\text{temp}}$);

//寻找节点 v 中从 δ_{temp} 开始连续的非零序列持续的最晚时间

18)　　$f^{\text{temp}}=\min\{s_v^{\delta_{\text{period}}}\mid\delta_{\text{period}}\in[\delta_{\text{temp}},\delta_{\text{temp1}}]\}$;

19)　　end if

20)　　$f^l=\min\{f^{\text{current}},f^{\text{temp}},f^{\text{next}},f^l\}$;

21)　　$v=u$;

22)　　$\delta_{\text{find}}=\delta_{\text{temp}}$;

23)　　$f^{\text{current}}=f^{\text{next}}$;

24)　　$f^{\text{temp}}=\infty$;

25)　　$f_{\text{sum}}=f_{\text{sum}}+f^l$;

26)　　f^l =min{ $f^l,k-f_{\text{sum}}$ };

27) end while

如算法 3 所示，求解最短路径的可行流是求解在某个时刻之前最短路径上的所有链路中能够传输的最大的数据量，算法从路径 l 的最后一跳链路开始，结合连续两跳链路中间节点的缓存时间序列求解出下一跳链路的寻路时间，然后根据寻路时间求解该链路的可行流，路径 l 所有链路的最小可行流即为路径 l 的可行流 f^l 。

3.2.3　更新残余网络

如算法 4 所示，在求出最短路径和最短路径的可行流之后，需要对残余网络进行更新，从而在下次选路时获得最新的网络拓扑。对残余网络的更新，就是按照路径 l 记录的路径，逐跳对每一跳链路上记录的权重时间序列进行更新，并且结合两跳链路上的流对两跳链路中间节点的缓存时间序列按照节点缓存限制式(4)进行了更新。

算法 4　更新残余网路

输入　$rSTAG$,源节点为 s ,目的节点为 e ,最短路径为 l ，可行流为 f^l 。

输出　更新后的 $rSTAG$ 。

update_residual_network ($rSTAG,l$, f^l)

1) $v=e.p$;

2) δ_{find} =find_earliest_period(W_{ve}^T);

3) $w_{ve}^{\delta_{\text{find}}}=w_{ve}^{\delta_{\text{find}}}-f^l$;

4) if $ev\notin rE$

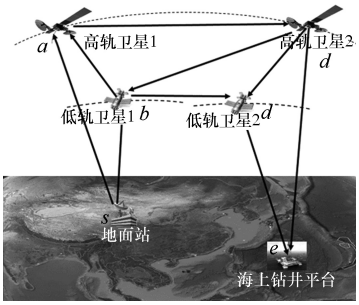
5)　rE .add(ev)；//将 ev 加入边集 rE 中

6) end if
7) $rw_{ev}^{\delta_{\text{find}}} = f^l$;
8) while($u \neq s$)
9) $u = v.p$;
10) if $uv \in E$
11) δ_{next} =find_earliest_period(W_{uv}^T);
12) $w_{uv}^{\delta_{\text{next}}} = w_{uv}^{\delta_{\text{next}}} - f^l$;
13) if $vu \notin rE$
14) rE .add(vu);
15) end if
16) $rw_{vu}^{\delta_{\text{next}}} = f^l$;
17) else
18) δ_{next} =find_earliest_period(rW_{uv}^T);
19) $w_{vu}^{\delta_{\text{next}}} = w_{vu}^{\delta_{\text{next}}} + f^l$;
20) $rw_{vu}^{\delta_{\text{next}}} = rw_{vu}^{\delta_{\text{next}}} - f^l$;
21) end if
22) if $\delta_{\text{next}} < \delta_{\text{find}}$
23) $s_v^{\delta_{\text{peroid}}} = s_v^{\delta_{\text{peroid}}} + f^l$, $\delta_{\text{peroid}} \in [\delta_{\text{next}}, \delta_{\text{find}-1}]$;
24) else
25) $s_v^{\delta_{\text{peroid}}} = s_v^{\delta_{\text{peroid}}} - f^l$, $\delta_{\text{peroid}} \in [\delta_{\text{find}}, \delta_{\text{next}}]$;
26) end if
27) $v = u$;
28) $\delta_{\text{find}} = \delta_{\text{next}}$;
29) end while

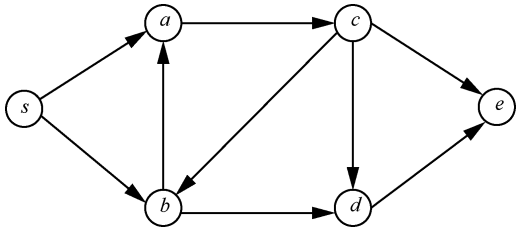
4　实例验证

在进行海上油气开发时，由于有些石油开采的地方不能被地面基站覆盖，需依靠卫星进行通信。本文对一个地面站向海上钻井平台发送信息的场景进行分析，如图 3(a)所示。地面站发送的信息可经过低轨和高轨卫星的存储转发，最后下传到海上钻井平台。该场景的网络拓扑，如图 3(b)所示，其中，s 节点和 e 节点分别对应地面站和海上钻井平台，a 节点和 c 节点分别对应高轨卫星 1 和高轨卫星 2，b 节点和 d 节点对应于地轨卫星 1 和地轨卫星 2。

本文以图 3(a)中的通信场景进行建模，对各节点之间的链路的连通时间段和链路容量做了合理的假设，构建了如图 3(a)所示的存储时间聚合图。同时，将多路径最短时延求解算法应用于此模型中，具体的求解过程如图 4 所示。



(a) DTN 网络中卫星通信场景



(b) 网络拓扑

图 3　仿真场景

在 STAG 中(如图 4(a)所示)，时间 T 被分成 4 个时间段，源节点为 s，目的节点为 e,假设从源节点向目的节点需要传输的数据为 12 个单位量。除源节点和目的节点外，每个节点的缓存时间序列，均初始为{0,0,0,0}。执行最短路径算法，首先找到与目的节点 e 相邻的所有邻接节点，获得其中具有最早连通时间段的链路 *d-e*, 而后从链路 *d-e* 出发继续找路，最后找最短路径可以找到路径 *s-a-c-d-e*(如图 4(b)所示)。通过路径可行流计算方法，获得该最短路径的可行流为 6，更新残余网络（如图 4(c)所示）。重复最短路径找路算法，首先找到具有最早连通时间段的链路为 *c-e*,然后从 *c-e* 出发寻找最短路径，可以找到 *s-b-d-c-e*,计算可行流为 6，更新残余网络(如图 4(d)所示)。此时，已找的所有路径可行流之和大于待传输的业务，结束找路算法，最终获得最短时延的路由方案(如图 4(e)所示)。

在相同的网络拓扑中(如图 5(a)所示)，CGR 路由算法依据最早连通选路规则，首先找到路径 *s-a-c-d-e*（如图 5(b)所示），由于 CGR 路由算法无法对已选路径进行修改，最终只能找到一条路径(如图 5(c)所示)，只能传输 6 个数据，不能满足业务 12 个数据的传输需求。

从该算例可以看出，所提算法最开始找到的路径与 CGR 算法找到的路径相同，但所提算法在找路过程允许对之前的路径进行了修正，从而增加了网络中的路由机会，大大地提高了网络中链路的利

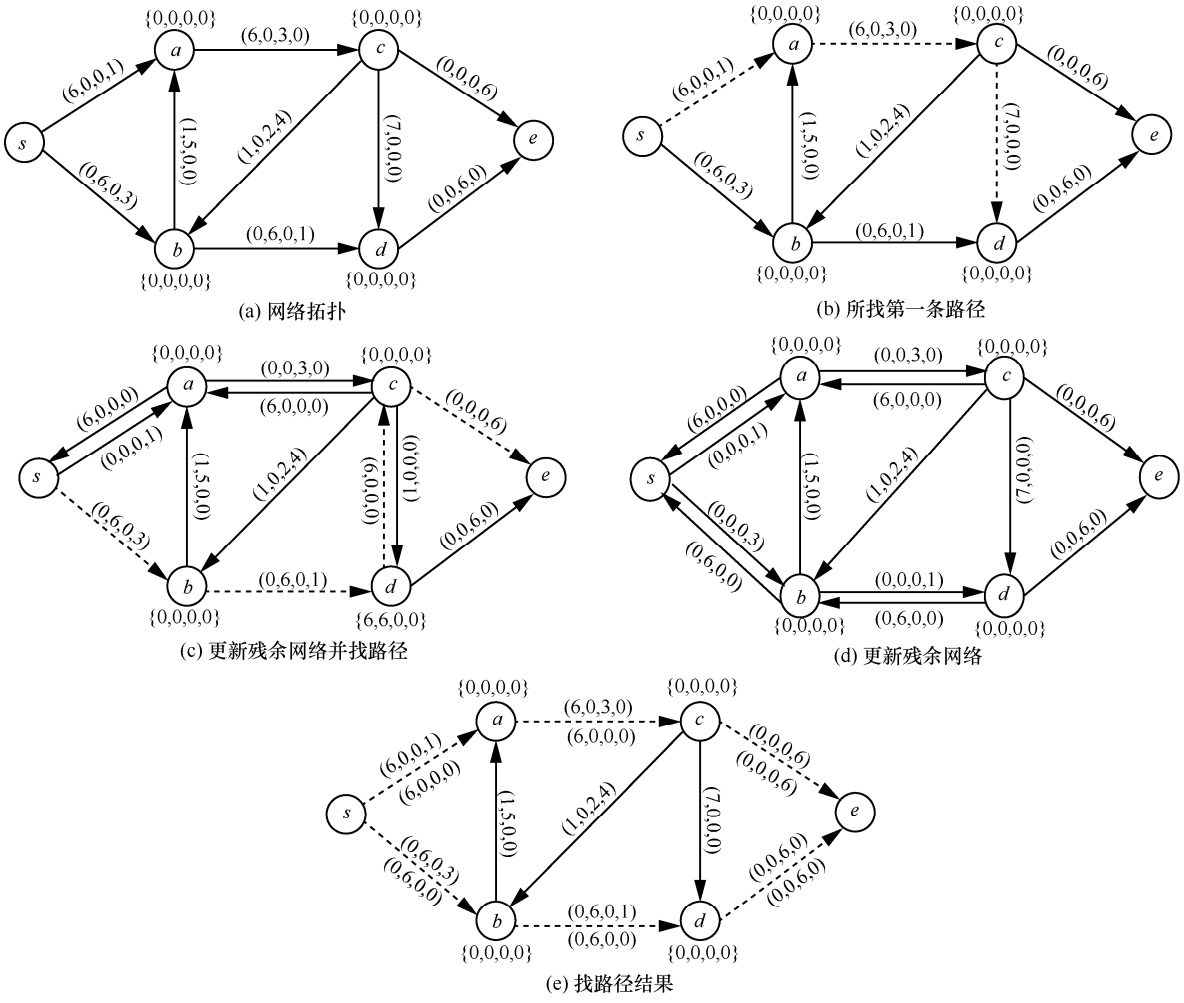


图 4　算法求解过程

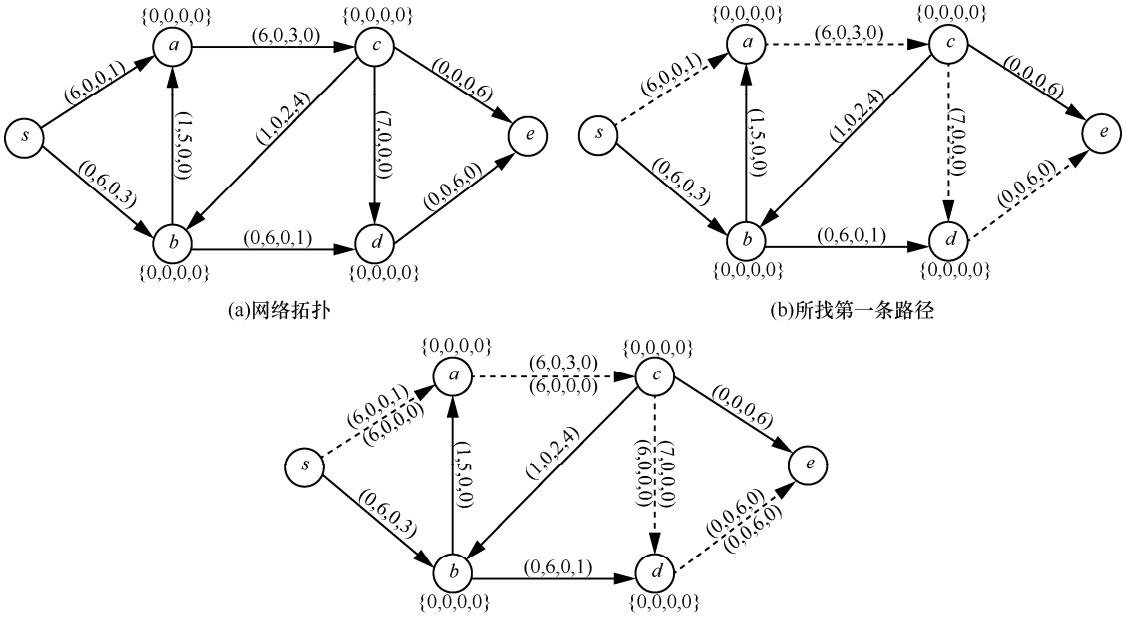


图 5　CGR 算法求解过程

用率，与 CGR 相比更加高效。

5 结束语

本文针对现有 DTN 网络路由算法 CGR 链路资源利用率低的问题，研究基于时间聚合图的高效多路径最短时延算法。利用时间聚合图中链路容量序列，同时表征容量与时段的特点从目的节点向源节点找路，结合深度优先搜索策略，快速高效地找到最短时延路径。此外，通过在时间聚合图中给节点增加缓存时间序列，增强了同一链路不同时间段之间的联系，使算法在选路过程中可以对已选路径做出修正，有效地解决了 CGR 中存在的链路资源浪费的问题。本文以吞吐量、时延为优化目标进行 DTN 网络路由的建模与求解，下一步的工作将继续挖掘时变图的优势，以能量、链路代价等费用为优化目标研究 DTN 网络的最小费用流问题。

参考文献：

[1] FALL K. A delay-tolerant network architecture for challenged internets[C]//2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM, 2003: 27-34.

[2] CHEN H, LOU W. On protecting end-to-end location privacy against local eavesdropper in wireless sensor networks[J]. Pervasive and Mobile Computing, 2015, 16: 36-50.

[3] SHAH S F A, ZAFAR M H, ANFONOVIC I, et al. Hybrid routing scheme for vehicular delay tolerant networks[C]//Computer Science and Electronic Engineering (CEEC), IEEE, 2016: 158-163.

[4] CAINI C, FIRRINCIELI R. DTN for LEO satellite communications[C]//International Conference on Personal Satellite Services. Springer Berlin Heidelberg, 2011: 186-198.

[5] JAIN S, FALL K, PATRA R. Routing in a delay tolerant network[M]. ACM, 2004.

[6] WERNER M, A dynamic routing concept for ATM-based satellite personal communication networks[J]. IEEE Journal on Selected Areas in Communications, 1997, 15(8): 1636-1648.

[7] GEORGE B, SANGHO K. Spatio-temporal networks: modeling and algorithms[M]. Springer Science & Business Media, 2012.

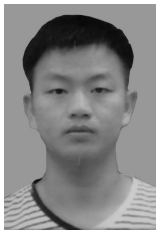
[8] SMITH D K. Network flows: theory, algorithms, and applications[J]. Journal of the Operational Research Society, 1994, 45(11): 1340.

[9] ARANITI G, BEZIRGIANNIDIS N, BIRRANE E, et al. Contact graph routing in DTN space networks: overview, enhancements and performance[J]. IEEE Communications Magazine, 2015, 53(3): 38-46.

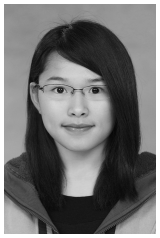
作者简介：

王鹏（1995-），男，河南济源人，西安电子科技大学硕士生，主要研究方向为空间信息网络。

李红艳（1966-），女，陕西西安人，博士，西安电子科技大学教授、博士生导师，主要研究方向为空间信息网络、无线网络、移动自组织网络。



张焘（1992-），男，福建三明人，西安电子科技大学博士生，主要研究方向为空间信息网络。



李朋云（1995-），女，河北保定人，西安电子科技大学硕士生，主要研究方向为空间信息网络。