流星余迹通信网络的路由算法

高 航 慕晓冬 易昭湘 仝 彤 袁覃恩
(火箭军工程大学信息工程系 西安 710025)

摘 要 流星余迹通信是一种重要的应急通信方式。其通信网络具有传输延时长和链路间歇中断的特点,适用于这种特殊网络的路由算法具有明显的针对性,有待深入研究。文中在研究流星余迹网络拓扑结构的基础上,基于OPNET仿真软件构建流星余迹组网模型,结合适用于DTN(Delay Tolerant Network)网络的ED(Earliest Delivery)算法和EDLQ(Earliest Delivery with Local Queue)算法的特点,分析通信时延模型,提出一种改进的OED(Optimistic Earliest Delivery)算法。基于已建立的模型对数据传输成功率和网络吞吐量进行仿真。仿真结果表明,OED算法在组网网络的数据吞吐量和数据传输成功率方面优于ED算法和EDLQ算法,能够避免因队列溢出导致的数据包丢失;通过增大节点容量,OED算法的数据通过率相对于ED算法和EDLQ算法分别提升了20%和8%;路由算法的选择不影响流星余迹节点间链路的平均持续时间和平均中断等待时间。OED算法在流星余迹网络中具有较强的适应性,能够为流星余迹组网的建设提供参考。

关键词 流星余迹通信,延时容忍网络,时延模型,OED算法

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.07.013

Routing Algorithm Based on Meteor-burst Communication

GAO Hang MU Xiao-dong YI Zhao-xiang TONG Tong YUAN Tan-en
(Department of Information Engineering, Rocket Force University of Engineering, Xi'an 710025, China)

Abstract Meteor-burst communication is an important emergency communication mode, and the communication network has the characteristics of long delay transmission and intermittent interruption of link. The routing algorithm applicable to this special network has obvious pertinence and needs further study. Based on the study of meteor topology network topology, this paper built a meteor network model based on OPNET simulation software, and proposed an improved algorithm OED (Optimistic Earliest Delivery) by analyzing the communication delay model combined with ED (Earliest Delivery) algorithm and EDLQ(Earliest Delivery with Local Queue) algorithm for DTN (Delay Tolerant Network). Based on the established model, the data transmission success rate and network throughput were simulated. The simulation results show that the OED algorithm is superior to the ED algorithm and EDLQ algorithm in terms of data throughput and data transmission success rate of network, and it can overcome the packet loss caused by the queue overflow. By increasing the node capacity, the data pass rate of OED algorithm is increased by 20% compared with ED algorithm and is increased by 8% compared with EDLQ algorithm. The choice of routing algorithm does not affect the average duration and the average interruption latency time of link between the meteor nodes. OED algorithm has strong adaptability in meteor trail network, which can provide reference for meteor burst network construction.

Keywords Meteor burst communication, Delay tolerant network, Delay model, Optimistic earliest delivery algorithm

流星余迹通信是利用流星进入大气层时对VHF无线电波的反射作用而实现超视距通信的无线通信方式,具有距离远、抗干扰、机动性强的特点。流星余迹通信介质受到破坏、攻击或者干扰的概率很小;建立成本较低,约为卫星通信的1/10;尤其在核爆条件下,流星余迹通信的接收信号高于平时的32倍,具有很强的顽存性,也被称为"世界末日的通信手段"。随着流星余迹技术的发展和网络规模的扩大,网络拓扑的动态变化程度越来越大,这对通信网络提出了更高的要求。近年来,适用于流星余迹网络的路由算法都有待深入研究。文献[1]建立了流星余迹通信体系结构,并根据通信协议的特点,基于OPNET仿真软件构建了流星余迹通信组网模型来仿真分析通信网络的性能指标;但对具体路由算法的研究相对缺失。文献[2]针对流星余迹路由提出了一种EDMB算法,并在环状和星状两种网络拓扑中分别进行仿真,所提算法

到稿日期:2017-05-13 返修日期:2017-08-09

高 航(1993-),硕士生,主要研究方向为计算机仿真,E-mail:2796314705@qq.com;**慕晓冬**(1965-),博士生导师,主要研究方向为计算机仿真,E-mail:wascom4@sina.com(通信作者);**易昭湘**(1980-),副教授,主要研究方向为通信仿真;**仝 彤**(1992-),硕士生,主要研究方向为计算机仿真;**袁覃恩**(1994-),硕士生,主要研究方向为计算机仿真。

相比于最短路径算法在平均流量和时延上体现出了优势；该文献同时指出流星余迹通信网络是一种典型的 DTN，依据 DTN 的性质，可以从时空和社会性质方面设计概率路由算法来提升网络的整体效果[3]。文献[4]提出了一种优化邻居节点选择的路由算法，其运用异或比较各个节点以确定最优下一跳，对数据包投递方式有一定见解。流星余迹网络与时延容忍网络具有一定的相似性，在考虑网络拓扑变化的同时，应加强针对网络节点拥塞的控制[5-6]，根据网络传输条件可以设置容量阈值，以避免网络拥塞[7]。文中结合流星余迹通信的特点，基于 OPNET 仿真平台建立流星余迹组网模型，综合考虑网络开销和数据包的传输能力，基于 ED 算法和 EDLQ 算法提出改进的 OED 算法；并针对数据通过率和网络吞吐量以及主站节点在仿真过程中的数据包溢出个数等参数进行仿真，以体现 OED 算法在流星余迹组网网络中的优势。

1　相关路由算法

1.1　ED 算法

最早的传递算法(Earliest Delivery，ED)是一种基于时变开销的算法，该算法对节点间链路连通的时间进行预测以确定数据包转发路径，但是对数据包的队列信息考虑不周全[8]。在某些情况下，当一个队列中需要发送的节点排在其他节点之后时，根据队列的先进先出原则，该消息将不能被正常发送，会遭遇很长的发送延迟；同时，在消息所存储的队列中，一旦发生队列溢出的情况，消息便会被强制丢弃，导致数据包丢失率较高，影响消息传输的可靠性，因此该方法在具体运用中存在一定的局限性。ED 算法的路由开销的表达式为：

$$w(e,t)=w'(e,t,m,s)$$

上式表示计算 t 时刻边 e 的开销，其中 m 是队列中消息的长度。w' 函数被定义为：

$$t'(e,t,m,s)=\min\{t''/\int_{x=t}^{t''}c(e,x)\mathrm{d}x\geqslant m\}$$

$$w'(e,t,m,s)=t'(e,t,m,s)-t+d(e,t'(e,t,m,s))$$

其中，t' 表示最早传输时间，$c(e,s)$ 为节点容量函数，t 表示目前时刻，w' 函数表示 t 时刻进入节点的消息在边 e 上的传输时间。

1.2　EDLQ 算法

EDLQ 算法是对 ED 算法的扩展，在其上增加了队列知识库的内容[9]。这里引入 Q 函数来表示时刻 t 与边 e 相关的节点 s 的队列长度，则：

$$t'(e,t,m,s)=\begin{cases}\min\{t_1/\int_t^{t_1}c(e,x)\mathrm{d}x\geqslant(m+Q(e,t,s))\}, & \text{if } e=(s,*)\\ \min\{t_1/\int_t^{t_1}c(e,x)\mathrm{d}x\geqslant m\}, & \text{otherwise}\end{cases}$$

EDLQ 的路由开销表达式与 ED 算法相同，但是 t' 需要修正为：

$$t'(e,t,m,s)=\min\{t''/\int_{x=t}^{t''}c(e,x)\mathrm{d}x\geqslant(m+Q(e,s,t))\}$$

2　基于流星余迹网络的路由算法

DTN 时延结构如图 1 所示。消息在节点之间的传输主要包括：传输时延 t_m，它的大小与数据包大小相关；队列时延 t_q，即数据包在节点中的队列开始排队直到前面所有数据包发送出去所花费的时间，根据节点数据包队列长度而定；传播时延，即数据包在链路中的传输耗费时间[10]。

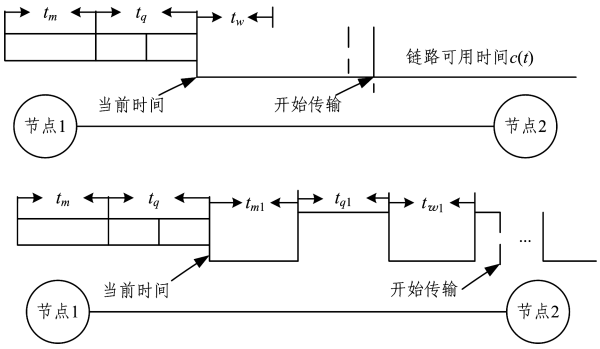


图 1　DTN 时延的组成

Fig. 1　Composition of DTN delay

ED 算法存在以下不足：

1)在选择数据包的传递路径时忽视了节点当前存储的情况，在节点存储空间有限的情况下易发生队列溢出的情况，造成数据包丢失概率增大。

2)ED 算法作为一种源路由选择算法，根据源节点的情况立即确定并固定传输路径的方式不适应流星余迹网络拓扑快速变化的情形。如果在传输过程中遇到链路变化的情况，导致路径已经不是最优路径甚至直接断开，数据包传输延迟将会增加，传输成功率也会下降。

EDLQ 算法是对 ED 算法的改进，其对节点的具体队列信息进行分析，每一个节点都会对路由的下一跳节点进行计算，加快了数据包的传输速度，但也有一定的概率形成环状路由，因此仍然无法适应间歇性网络拓扑；同时，其在转发数据包的过程中没有考虑后续节点的容量，可能造成节点存储空间已满而导致数据包被强制丢失。

结合流星余迹通信组网的特点，从以下方面对 ED 算法进行改进。

1)队列时延

ED 算法在计算 DTN 中的最短路径时，通常只设定消息的传输时延和传播时延而忽略了队列时延。在网络数据包的实际传播过程中，可能会出现多个消息积压在同一个节点中的情况，使排队时延会大大增加，因此队列时延成为计算总时延时必须考虑的一个要素[11]。若不考虑队列时延，数据包转发路径的选择将不准确，甚至出现丢失包现象。

2)动态路由中最短路径的选择

当一个节点接收到数据包而未将其发出之前，链路会发生变化，从而组成一个新的间歇性通信链路全网拓扑，此时需要立即计算当前网络拓扑的最短时延路径，如果下一个时间段拓扑再次发生变化，则需要再次构造路径，以防止数据包在某个节点长期滞留。这里需要利用一种基于时变的路由算法来计算下一跳路由。

3)存储空间

当链路连通时，所有与源节点存在链路连通的节点直接将自身的剩余缓存容量信息沿链路发送至源节点，并在满足

发送条件的基础上寻找时延最小的路径来转发数据包。重复上述步骤，避免因为中间节点的容量不足而导致出现丢包现象。如图 2 所示，当 1 号源节点与 2，3，5 号节点建立连接后，这 3 个节点迅速将自身的剩余容量信息发送至 1 号节点，然后由 1 号节点来判断哪一个节点具有接收数据包的能力，并在这些节点范围内寻找时延最短路径进行转发。

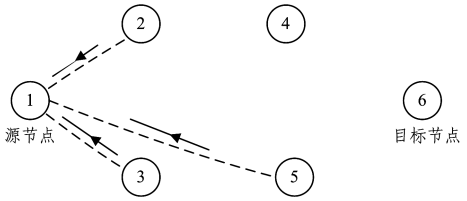


图 2 节点剩余容量的探测

Fig. 2 Detection of node remaining capacity

OED 算法的具体步骤如下：

1)标记初始节点 S 为源节点，此时没有流星余迹到达，节点之间的链路处于断开状态，即 S 节点与其他节点的时延为无穷大。这里在数据包格式中设置节点标识模块，用于记录数据包传递过程中经过的节点，以防止形成包震荡。

2)当流星余迹到达时，链路连通，形成了一个间歇性的全网拓扑网络。与节点 S 连通的节点迅速将本节点的剩余缓存容量信息发送至 S 节点，当各个节点剩余容量足够时，在这个间歇性网络拓扑中采用时变 Dijikstra 算法来计算最短时延路径并按照该路径进行传输。这里需要计算的时延为，数据包在源节点到目的节点之间的链路上传播的时延与目的节点在链路连通情况下将目前所积压数据包发送至信道上的队列时延之和。传输后标记本节点为新的 S 节点以作为下次转发的初始节点。在节点剩余容量不满足条件的情况下，选择满足节点剩余容量的条件下队列时延和传输时延之和次小的路径进行传输，以确保数据包不在任何一个节点出现长期滞留的情况。

3)当链路通断导致全网拓扑发生变化时，会构成一个新的网络拓扑，原网络拓扑结构不再存在，所计算的最短路径也失去了意义。根据新链路的链接方式，按照步骤 2)所述的剩余容量规则改变原有的传输路径，然后再次计算新网络拓扑下当前持有数据包的节点到目的节点的最短路径，转发数据包，直到数据包到达目的节点，完成传输。

为更清晰地说明上述情况，按照图 3 的拓扑结构进行分析。如图 3(a)所示，当流星到达时，源节点至目的节点之间出现两条链路，分别为 1→2→3→8 与 1→2→5→8，链路上方标记传播时延的大小，每个节点标记队列时延的大小，可知两条链路的时延总和分别为 10 和 13。在这个拓扑网络中，假设各个节点的容量满足传输条件，选取最短的时延路径 1→2→3→8。如图 3(b)所示，当数据包传输至 2 号节点时，网络拓扑发生变化，重新计算最短时延路径，并标记 2 号节点为源节点，将数据包发送至距离 8 号目的节点更近的 5 号节点。图 3(c)给出另一种情况，即源节点为 2 号，假设此时按照时延计算出路径 2→3→8 更加合理，但此时 3 号节点剩余容量不足，因此放弃该路径，选择路径 2→5→8。

4)若网络拓扑在数据包传输过程中发生了变化，则重复步骤 2)和步骤 3)，直至数据包到达目的节点。

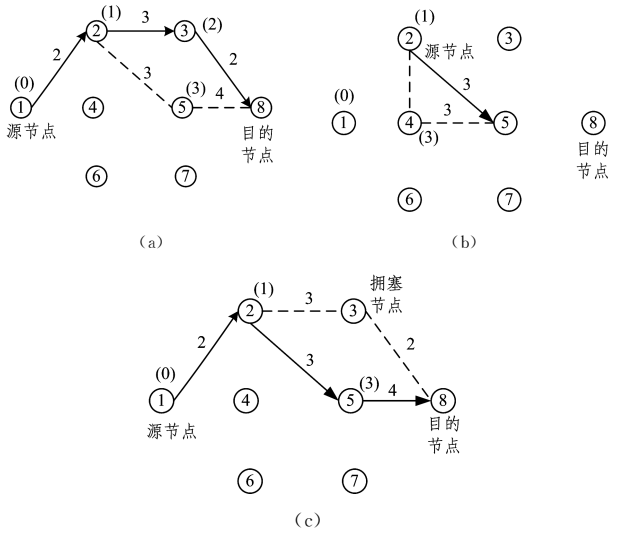


图 3 OED 路由算法状态图

Fig. 3 State diagram of OED algorithm

3 流星余迹组网模型的建立

OPNET 是一种功能强大的网络仿真软件，包含多种封装的模块和函数，可以有效针对各类无线通信进行仿真分析。文中针对流星余迹组网设置的特点，基于 OPNET 仿真软件构建流星余迹通信组网，并根据各路由算法的特点对节点模型进行针对性设置，以达到预期的仿真效果[12]。

3.1 组网设计

选取 5 个主站的环状网络作为组网方案，每个主站连接 4 个从站，每个主站与所属从站通信前应发送探测信号来探测此刻流星的到达情况，如果满足信道最低要求，从站立即取出排队的数据包并发送至主站。

流星余迹组网仿真参数的设置如下：

1)流星余迹的网络场景范围为 1800 km * 1800 km；

2)网络节点数为 25 个；

3)数据分组产生的间隔符合参数为 13.2 的泊松分布；

4)数据分组大小为 120 bit；

5)链路等待连通的时间符合参数为 10 的泊松分布；

6)链路连通保持的时间符合参数为 1 的泊松分布；

7)设置每个节点缓存容量为 3000 bit。

流量余迹网络的结构如图 4 所示，其中包含 25 个节点模拟站点，红色为 5 个主站，蓝色为红色节点辖下的 4 个从站。

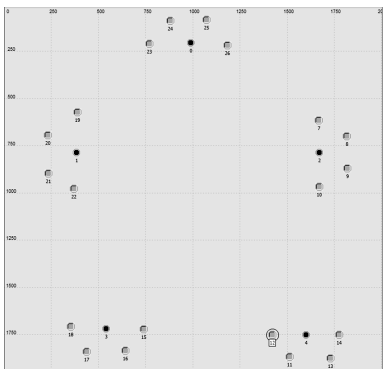


图 4 流星余迹网络结构图

Fig. 4 Structure of meteor burst network

3.2　节点模型的设置

图 5 和图 6 分别给出主站和从站的节点模型。从图中可以看出，一个主站节点有 4 个发射机，它们对应所属的 4 个从站节点；而一个从站对应一个接收机，只能与所属主站进行通信，如果需要与其他主站所属的从站进行通信，则需要通过主站进行中转。

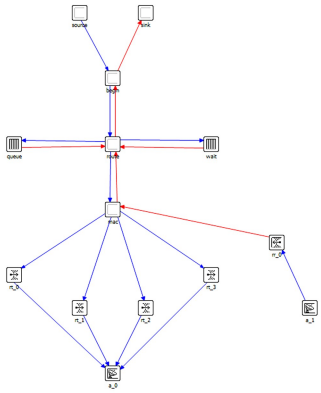


图 5　主站节点模型

Fig. 5　Master node model

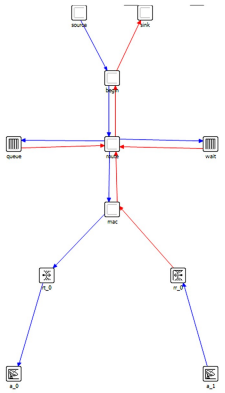


图 6　从站节点模型

Fig. 6　Slave node model

source 模块是整个组网数据包的产生源头，数据产生模型符合参数(时间间隔)为 13.2 的泊松分布。

begin 模型的作用是对产生的数据包进行初始化，利用产生随机数的方法随机产生 0～24 之间的 10 个数字代表发送到包括当前节点在内的 25 个节点中，当本节点作为目的节点时，将数据包送至 sink 模块进行销毁。

route 模块是路由算法的核心体现，当数据包到达该模块时，该节点判断数据包是否到达目的节点，若没有到达目的节点则分析并确定下一跳节点，反之接收该数据包，结束整个传输过程。

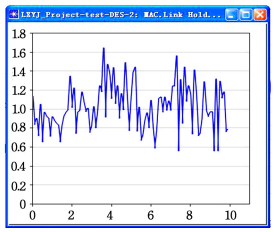
sink 模块用于销毁初始化过程中目的节点为本节点的错误数据包，确保整个传输过程正常完成。

wait 模块用于收容无法立即传输的数据包，同时负责确保排队等待的模块在满足条件后返回 route 模块。

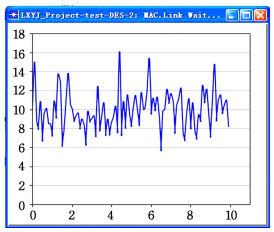
queue 模块的功能与 route 模块相似。数据包滞留于该节点，主要是因为链路的间歇性通断和预计的下一跳节点因排队过长导致节点容量不足而无法传输。当链路连通时，采用 OPNET 中的 OPC_QSTAT_BITSIZE 函数获取队列中积压的比特数，并采用 OPC_QSTAT_FREE_BITSIZE 函数获取当前队列还可以容纳的比特数；然后直接向 queue 模块发送剩余容量信息，接收到信息的模块决定是否发送数据包，如果无法发送则需要等待，当滞留时间过长时，则选择次优下一跳节点为目的节点。

4　路由算法的仿真与分析

图 7 给出了网络拓扑中节点之间所有链路保持连通的时间和链路的中断等待时间的仿真结果。其中，图 7(a)的连通保持时间在 1 s 上下浮动，浮动范围为 0.6～1.6 s；图 7(b)的中断等待时间在 10 s 上下浮动，浮动范围为 6～16 s。仿真结果符合上文中设置的仿真参数。



(a)链路保持连通的时间



(b)链路的中断等待时间

图 7　链路仿真结果

Fig. 7　Link simulation results

4.1　数据传输成功率

图 8 显示了数据传输成功率的变化趋势。由仿真结果可以看出，在 0～1 min 时间段内，OED 算法的传输成功率略高于 ED 算法和 EDLQ 算法，处于上升阶段；在仿真时间超过 1 min 后，各算法的传输成功率的差距逐渐拉大；仿真时间在 4～10 min 期间，3 种算法的性能差距极为明显，OED 算法的数据成功到达率远高于 ED 算法，最后稳定在 75%左右，而 ED 算法在 50%左右，EDLQ 算法略优于 ED 算法，约为 62%。以上结果说明，OED 算法对数据包排队溢出有一定的规避作用，考虑时延更加全面，数据包的传输路径能适应网络拓扑的动态变化，数据包传输路径的选择更合理，在数据传输成功率方面显示出一定的优势。

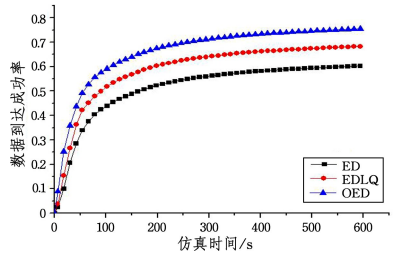


图 8　数据传输成功率的比较

Fig. 8　Comparison of data transmission success rate

4.2　网络吞吐量

图 9 对比了 3 种路由算法作用下的网络吞吐量。同样地在 0～60 s 内，3 条曲线快速上升，几乎重合，因为在链路建立初期，链路通断对网络性能的影响不明显，所以 3 种算法的差距不大。随着仿真时间的增加，链路通断对网络拓扑数据包传输的影响增强，OED 算法作用下的吞吐量高于 ED 算法和 EDLQ 算法，最后稳定在 1200 bps；ED 算法的吞吐量稳定在 1000 bps；EDLQ 算法虽然数据包传输速率更快，但由于出现路径环路的概率大，吞吐量受到影响，数值略高于 ED 算法，稳定在 1090 bps，同时图像在 250 s 处出现突降，这是因为形成了路径环路而影响到了吞吐量。

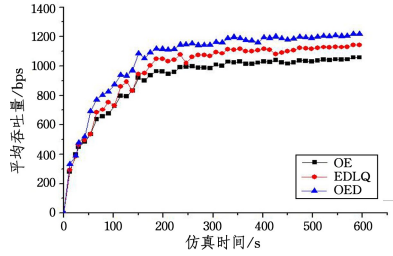
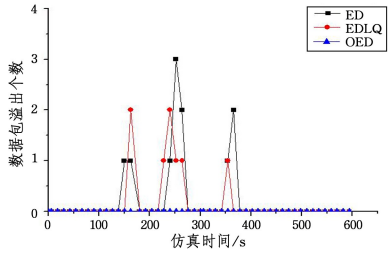


图 9　网络吞吐量的对比

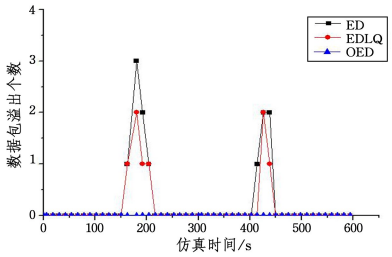
Fig. 9　Comparison of network throughput

4.3 主站节点排队数据包的个数变化

对主站节点中的队列存储容量进行仿真，以 0 号与 1 号主站节点为研究对象，调用 OPNET 中的 OPC_QSTAT_OVERFLOWS 函数来显示节点中队列溢出数据包的个数，如图 10 所示。



(a)0 号主站节点



(b)1 号主站节点

图 10 主站节点数据包的溢出结果

Fig. 10 Overflow results of master node packet

由图 10(a)可以看出，在 162 s，180 s，192 s 和 204 s 时刻，矩形标记曲线突变；在 414 s，426 s 和 438 s 时刻，圆形标记曲线突变；而三角标记图线数值一直保持为 0，说明此时在 ED 和 EDLQ 算法的作用下，数据队列已满，若继续发送其余数据将会造成溢出，发生数据包丢失的情况。ED 算法和 EDLQ 算法在这方面较为类似，都没有考虑队列溢出的情况；而 OED 算法对节点容量控制得更好，这是因为在链路连通时，所有可能的下一跳节点都会将本节点的剩余容量自动发送至发送节点，以判断下一跳节点是否有足够的空间来接收更多的数据包，从而防止数据包因排队原因丢失。1 号主站也有上述类似的变化趋势，如图 10(b)所示。

4.4 不同节点容量下的数据通过率仿真

图 11 反映的是 3 种算法在增加流星余迹通信节点容量的情况下的数据包平均通过率。

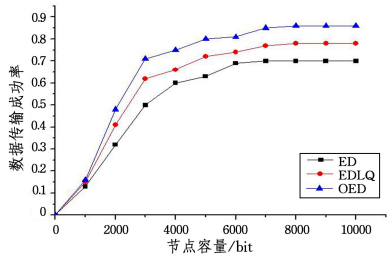


图 11 不同节点容量下的数据通过率比较

Fig. 11 Comparison of data pass rates for different node capacities

3 种算法的数据包通过率在初始阶段提升得很快，随着节点容量增加最后趋于平稳。OED 算法稳定在 0.8 左右；ED 算法稳定在 0.6 左右；EDLQ 算法略高于 ED 算法，最后稳定在 0.72。OED 算法的延时小于 ED 算法和 EDLQ 算法的延时，因为可用链路建立前需要一段很长的等待时间，或者链路突然断开造成了发送的延误，而 OED 算法考虑到路径的排队延时，可以更准确地计算出最优下一跳节点，同时考虑到了下一跳节点的剩余容量，从而提高了整个网络的延时容忍力和容错力，提升了流星余迹算法的效率。站点容量的增大有利于提高流星余迹网络的数据承载力，OED 算法在容量大的主从站节点中体现出了更大的优势。

结束语 文中根据流星余迹网络的特点，建立了更全面的延时模型，并在分析延时模型的基础上，基于以往对延时容忍网络路由算法的研究，结合流星余迹组网信息的传输需求，提出了一种 OED 算法。在 OPNET 仿真平台上的实验表明，相比较于 ED 算法和 EDLQ 算法，OED 算法在网络吞吐量和数据成功到达率方面具有较大的优势，能通过探测目的节点的剩余容量来有效解决因队列溢出导致的数据包丢失问题。同时，通过仿真得到了流星余迹链路保持时间和中断时间的变化范围，在增大节点容量的情况下，OED 算法考虑了路径的排队延时，计算最优下一跳节点的方式更加科学和准确，体现出了明显的优势，为流星余迹组网建设路由算法的选择提供了借鉴。接下来，需要增加站点，以进一步扩大组网规模，从而获得更加具有代表性的实验结论。

参 考 文 献

[1] LI Q, ZHU L D. Modeling and Simulation of Meteor Burst Communication[J]. Communications Technology, 2010, 43(11): 78-80. (in Chinese)
黎庆，朱立东. 流星余迹通信网络建模与仿真[J]. 通信技术，2010，43(11)：78-80.

[2] WANG Y C, LI H Y. Research of Routing Algorithm Based on the Meteor-Burst Communication [D]. Xi'an: Xidian University, 2009. (in Chinese)
王永灿，李红艳. 基于流星余迹通信网络的路由算法研究[D]. 西安：西安电子科技大学，2009.

[3] JIA J X, LIU G Z, XU M. Probability Routing Algorithm in DTN Based on Time and Space and Sociality[J]. Computer Science, 2017, 43(6A): 295-301. (in Chinese)
贾建新，刘广忠，徐明. DTN 基于时空和社会性的概率路由算法[J]. 计算机科学，2017，43(6A)：295-301.

[4] ZHANG L, CHEN Z G, WU J, et al. Optimal Selection of Neighbor Node Routing Protocol[J]. Journal of Chinese Computer System, 2017, 38(1): 62-66. (in Chinese)
张霖，陈志刚，吴嘉，等. 最优化选择邻居节点路由协议[J]. 小型微型计算机系统，2017，38(1)：62-66.

[5] FRAIRE J A, MADOERY P G, FINOCHIETTO J M. On the Design and Analysis of Fair Contact Plans in Predictable Delay-Tolerant Networks[J]. IEEE Sensors Journal, 2014, 14(11): 3874-3882.

[6] ZHAO H, LIANG H. Contact Gragh Routing Algorithm Based on Capacity Constraints in Deep Space Communication[J]. Science Technology and Engineering, 2016, 16(17): 210-214. (in Chinese)
赵辉，梁花. 基于容量约束的深空接触图路由算法[J]. 科学技术与工程，2016，16(17)：210-214.

[7] ZHAO G S，CHEN M. Congestion control mechanism based on accepting threshold in delay tolerant networks[J]. Journal of Software，2013，24(1)：153-163. (in Chinese)
赵广松，陈鸣. 基于接收阈值的容延网络拥塞控制机制[J]. 软件学报，2013，24(1)：153-163.

[8] YAN H C，GUO J，ZHANG H J. Performance evaluation of routing algorithms on space delay/disruption tolerant networks[J]. Institute of Spacecraft System Engineering，2016，36(4)：38-46. (in Chinese)
燕宏成，郭坚，张红军. 空间延迟/中断容忍网络路由算法性能评估[J]. 空间科学与技术，2016，36(4)：38-46.

[9] ZHOU X B，ZHOU J，LU H C. Analysis of Delay Model in DTN[J]. Journal of Computer Research and Development，2008，45(6)：960-966. (in Chinese)
周晓波，周健，卢汉成. DTN 网络的延时模型分析[J]. 计算机研究与发展，2008，45(6)：960-966.

[10] WANG E，YANG Y J，LI L. Game of Life Based Congestion Control Strategy in Delay Tolerant Networks[J]. Journal of Computer Research and Development，2014，51(11)：2393-2407. (in Chinese)
王恩，杨永健，李莅. DTN 中基于生命游戏的拥塞控制策略[J]. 计算机研究与发展，2014，51(11)：2393-2407.

[11] QIU K. Performance of DTN Bundle Routing Protocol in Deep Space[D]. Harbin：Harbin Institute of Technology，2012. (in Chinese)
邱坤. 深空 DTN 集束层路由协议研究[D]. 哈尔滨：哈尔滨工业大学，2012.

[12] 陈敏. OPNET 网络仿真[M]. 北京：清华大学出版社，2004.

(上接第 65 页)

[10] Apache hbase[EB/OL]. http://hbase.apache.org.

[11] LAKSHMAN A，MALIK P. Cassandra：a decentralized structured storage system[J]. Acm Sigops Operating Systems Review，2010，44(2)：35-40.

[12] Ssdb-a fast nosql database for storing big list of data[EB/OL]. https://github.com/ideawu/ssdb.

[13] AMER A，LONG D D E，MILLER E L，et al. Design issues for a shingled write disk system[C]// IEEE，Symposium on MASS Storage Systems and Technologies. IEEE Computer Society. 2010：1-12.

[14] AMER A，HOLLIDAY J A，DE LONG D，et al. Data management and layout for shingled magnetic recording[J]. IEEE Transactions on Magnetics，2011，47(10)：3691-3697.

[15] Leveldb：a fast and lightweight key/value database library by google[EB/OL]. http://code.google.com/p/leveldb.

[16] FCOOPER B，SILBERSTEIN A，TAM E，et al. Benchmarking cloud serving systems with ycsb[C]// ACM Symposium on Cloud Computing. 2010：143-154.

[17] PITCHUMANI R，HUGHES J，MILLER E L. SMRDB：Key-Value Data Store for Shingled Magnetic Recording Disks[C]// 8th ACM International Systems and Storage Conference. 2015.

[18] YAO T，WAN J G，HUANGY P，et al. A Light-weight Compaction Tree to Reduce I/O Amplification toward Efficient Key-Value Stores[C]// 33rd International Conference on Massive Storage Systems and Technology. 2017.

[19] SAXENA M，SWIFT M M，ZHANG Y Y. Flashtier：a lightweight，consistent and durable storage cache[C]// Proceedings of the 7th ACM european conference on Computer Systems. 2012：267-280.

[20] KGIL T，ROBERTS D，MUDGE T. Improving NAND Flash Based Disk Caches[C]// International Symposium on Computer Architecture. IEEE，2008：327-338.

[21] YANG Q，REN J. I-CASH：Intelligently Coupled Array of SSD and HDD[C]// IEEE，International Symposium on High PERFORMANCE Computer Architecture. IEEE，2011：278-289.

[22] SRINIVASAN M，SAAB P. A general purpose，write-back block cache for linux[EB/OL]. https://github.com/facebookarchive/flashcache.

[23] THORNBER M S J，MAUELSHAGEN H. dm-cache[EB/OL]. https://en.wikipedia.org/wiki/Dm-cache.

[24] Emc fast cache：A detailed review[EB/OL]. http://www.emc.com/collateral/software/white-papers/h8046-clariioncelerra-unified-fast-cache-wp.pdf.

[25] Exadata smart flash cache features and the oracle exadata databas machine[EB/OL]. http://www.oracle.com/technetwork/serverstorage/engineer/d-systems/exadata/exadata-smart-flash-cache-366203.pdf.

[26] ZHOU Y Y，CHEN Z F，LI K. Second-level buffer cache management[J]. IEEE Transactions on parallel and distributed systems，2004，15(6)：505-519.

[27] JIANG S，ZHANG X. Lirs：An efficient low inter-reference recency set replacement policy to improve buffer cache performance[C]// Proceeding of 2002 ACM SIGMETRICS. 2002.

[28] NMEGIDDO，MODHA D. Arc：a self-tuning，low over-head replacement cach[C]// Proceedings of the 2nd USENIX Symposium on File and Storage Technologies. 2003.

[29] PRITCHETT T，THOTTETHODI M. SieveStore：a highly-selective，ensemble-level disk cache for cost-performance[C]// International Symposium on Computer Architecture. ACM，2010：163-174.

[30] HUANG S，WEI Q，CHEN J，et al. Improving flash-based disk cache with lazy adaptive replacement[C]// Proceedings of the 29th International Conference on Massive Storage Systems and Technology. 2013.

[31] GREGGB. L2arc[EB/OL]. https://blogs.oracle.com/brendan/entry/test.

[32] Under the hood：Building and open-sourcing rocksdb[EB/OL]. http://goo.gl/9xulVB.

[33] ZHANG Z，YUE Y，HE B，et al. Pipelined Compaction for the LSM-Tree[C]// IEEE，International Parallel and Distributed Processing Symposium. IEEE Computer Society，2014：777-786.

[34] WANG P，SUN G Y，JIANG S，et al. An efficient design and implementation of lsm-tree based key-value store on open-channel ssd[C]// Proceedings of the Ninth European Conference on Computer Systems. 2014.