航 空 学 报　　Jul. 25 2020 Vol. 41 No. 7
Acta Aeronautica et Astronautica Sinica　　ISSN 1000-6893 CN 11-1929/V
http://hkxb.buaa.edu.cn hkxb@buaa.edu.cn

引用格式：吕娜，陈坤，陈柯帆，等．适应拓扑变化的拥塞最小化网络更新策略[J]．航空学报，2020，41(7)：323661．LYU N，CHEN K，CHEN K F，et al．Congestion-minimization network update strategy for topology changes[J]．Acta Aeronautica et Astronautica Sinica，2020，41(7)：323661 (in Chinese)．doi：10.7527/S1000-6893.2019.23661

适应拓扑变化的拥塞最小化网络更新策略

吕娜[1]，陈坤[1,*]，陈柯帆[2]，朱海峰[1]，潘武[1]

1. 空军工程大学 信息与导航学院，西安 710077
2. 中国人民解放军 94860 部队，南京 210000

摘 要：软件定义网络的出现推动了航空集群机载网络的发展，同时也为机载网络的一致性更新带来了挑战。针对拓扑变化导致的网络更新期间可用路径失效问题，提出适应拓扑变化的拥塞最小化网络更新策略。首先，为减少可用路径失效带来的数据包丢失，在更新之前采用重路由机制进行处理；然后，针对重路由机制可能引起的网络拥塞，设计了贪婪流迁移算法缓解拥塞；最后，通过瞬时拥塞最小化更新算法完成整个网络更新。仿真结果表明：与传统的拥塞一致性更新算法相比，所提更新策略用规则开销、瞬时网络拥塞方面的少量提高换取了网络更新期间数据包丢失的显著降低。

关键词：航空集群；机载网络；软件定义网络；网络更新；拓扑变化；容量一致性

中图分类号：V249.1　　**文献标识码**：A　　**文章编号**：1000-6893(2020)07-323661-10

伴随着战争形态向信息化、网络化和智能化的演变，为提升航空作战平台效能发挥，航空集群作战的概念被提出。由一定数量、能力各异的有人/无人航空平台组成航空集群，以网络为中心，根据作战需求进行灵活高效的组织协同，从而实现平台间优势互补，最大限度提升作战效能[1]。

机载网络作为各航空集群成员信息传输、共享的承载，是进行高效协同作战的核心前提。而当下以各类数据链为代表的机载网络依然采用传统的分布式网络体系架构，网络的维护和管理复杂僵化[2]。与内部协议紧耦合的网络设备，只能根据所需网络服务添加越来越多的协议，难以升级和拓展功能，严重制约了机载网络的作战效能发挥。

软件定义网络(Software-Defined Networking，SDN)为航空集群机载网络的发展打开了新局面[3]。SDN 将控制功能从网络设备中独立出来，采用逻辑集中的控制器进行统一监视与管控，控制平面与数据平面的解耦使得对网络的集中管控和资源的灵活调度变得更加简单[4-6]。文献[7]将 SDN 范式与机载网络相融合，提出软件定义航空集群机载网络(Software-Defined Airborne Network of Aviation Swarm，SDAN-AS)，但机载网络引入 SDN 带来诸多好处的同时也为网络的一致性更新带来了一系列新的挑战。

在 SDN 中，为了响应流量需求变化、链路或节点故障、安全策略变化等事件，需要频繁地更新网络状态[8]。而网络状态的更新依赖于对交换机转发规则的更改，需要在一系列交换机上添加、删除或修改流表规则。尽管 SDN 提供了逻辑集中的控制，但本质上依然是复杂的异步分布式系统[9]。由于控制器与交换机之间存在固有的不可预测的延迟，且交换机反应和处理时间存在差异，即使控制器同时下发所有的更改指令，更改指令到达各交换机的时间不同，各交换机成功更改转

收稿日期：2019-11-18；退修日期：2019-12-16；录用日期：2020-02-27；网络出版时间：2020-04-07 11:08
网络出版地址：http://hkxb.buaa.edu.cn/CN/html/20200720.html
基金项目：国家自然科学基金(61703427)
* 通信作者．E-mail：648315977@qq.com

发规则的时间也不同[10]。这将导致网络更新期间可能出现交换机转发规则不一致，继而产生如转发循环、路由黑洞、链路拥塞等问题，从而影响业务传输的有效性和可靠性，严重降低网络性能[10-12]。因此，网络更新成为亟待解决的挑战性问题，即如何在满足网络一致性的情况下，将初始网络状态转变为最终网络状态。

近年来，研究人员针对各种不同的一致性要求对网络更新问题开展了大量研究。针对无环一致性，文献[13]提出依赖林算法，文献[14]提出反向更新算法，保证了更新期间数据包传输无环路。文献[9]针对数据包一致性提出两阶段更新机制，保证在网络更新期间每个数据包只单一按照旧的路径或者按照新的路径转发，而不是两者的混合。而后，文献[15]提出增量两阶段更新，将更新过程划分为多个轮次，牺牲更新时间来换取流表规则开销的减少。上述2种更新策略只能保证网络更新期间数据包的一致性，无法满足拥塞一致性，即在网络更新期间不产生拥塞。

为实现无拥塞网络更新，文献[16]提出了SWAN(Software-driven WAN)更新系统，为网络中每条链路预留空闲容量 s，通过线性规划求解一系列中间状态，并证明了至多需要 $1/s-1$ 次更新即可实现无拥塞网络更新。但SWAN必须始终预留一部分链路容量，牺牲了网络带宽资源利用率。文献[17]拓展了SWAN中的线性规划公式，提出拥塞最小化(Congestion-Minimizing，CM)更新算法，采用随机舍入法近似计算出给定最大瞬时拥塞情况下的最小化中间阶段数目的更新，用一定的拥塞换取更新速度的提高。文献[10]提出Dionysus更新系统，构建全局依赖关系图并根据交换机间的依赖关系进行动态更新，从而提高网络更新速度。文献[18]提出Cupid更新系统，通过关键节点将全局依赖图转化为具有潜在拥塞链路的局部依赖图，有效加快更新速度。

上述网络更新策略[10,16-18]能够较好地保证网络更新期间的拥塞一致性，但这些策略主要关注于地面有线网络，网络拓扑是固定不变的，未针对无线网络的特点进行考虑。然而航空集群作战场景下的网络拓扑随着作战任务和需求的变化而变化。拓扑变化必然导致部分旧转发路径失效，继续在其上传输业务时将产生严重的数据包丢失现象[19]，现有的网络更新策略不能适应SDAN-AS的需求。本文针对上述问题，提出适应拓扑变化的拥塞最小化更新策略(Congestion-Minimizing for Topological Changes，CM-TC)，考虑拓扑变化的影响，对受影响的业务流采用重路由机制进行处理，并设计贪婪流迁移算法降低产生的网络拥塞，最后通过拥塞最小化更新算法完成整个网络更新。

1 问题描述

1.1 网络模型

在问题描述之前，首先引入本文的网络模型。本文将软件定义航空集群机载网络抽象为无向图 $G=(V,E)$。其中 $V=\{v_1,v_2,\cdots,v_m\}$ 表示网络中交换机的集合，$E=\{e_1,e_2,\cdots,e_n\}$ 表示链路集合。对于 $\forall e\in E$，C_e 表示该链路的容量。$\mu=\{\mu_{e_1},\mu_{e_2},\cdots,\mu_{e_n}\}$ 表示网络链路利用率的集合。G_o 表示初始网络状态，G_n 表示最终网络状态。$F=\{f_1,f_2,\cdots,f_k\}$ 表示网络中业务流的集合，对于 $\forall f\in F$，d_f 表示该流的需求，将每条流分别打上不同的VLAN(Virtual Local Area Network)标签将流拆分到多条路径进行传输。$P_\mathrm{o}(f)$ 表示 G_o 中的流 f 的路径集合，$P_\mathrm{n}(f)$ 表示 G_n 中流 f 的路径集合，$P_\mathrm{a}(f)\subset P_\mathrm{o}(f)$ 表示 G_o 中流 f 已失效路径的集合，F_a 表示受失效路径影响的流的集合。

1.2 网络更新问题描述

在SDAN-AS场景中，由于作战任务、作战需求的变化，网络拓扑在较长的时间跨度上呈现出较大的变化，但在较短的时间跨度上，网络拓扑的变化可以视为局部的、小范围的变化。网络拓扑的变化可以是由于节点/链路故障或节点间相对运动所导致的。如图1(a)所示，对于源节点为1，目的节点为4的流 f，通过打上不同的VLAN标签将流 f 拆分到多条路径进行传输，流 f 的路径集合为 $P_\mathrm{o}(f)=\{p_1,p_2,p_3\}$，$p_1$、$p_2$、$p_3$ 为图1(a)中实线箭头所指路径。每条路径上的节点根据标签进行相应的转发，在源节点1处为每条路径分配权重实现流量分配。

当网络拓扑由图1(a)转变为图1(b)时，节点

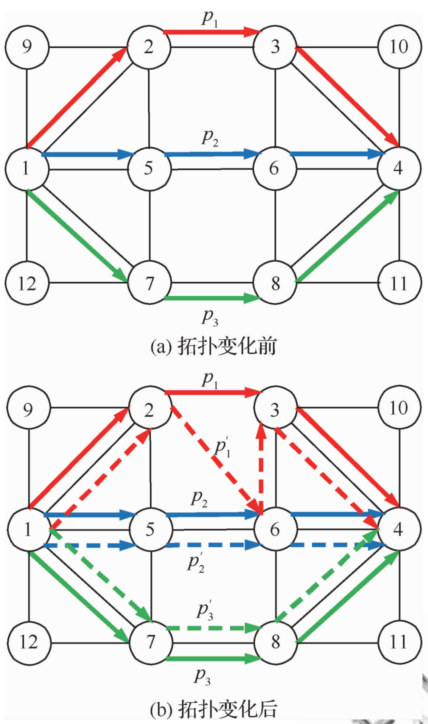

图 1 拓扑变化前后的网络状态

Fig. 1 Network state before and after topology change

2 和节点 3 之间的链路中断,路径集合中的 p_1 已经失效,然而节点 1～节点 4 交换机中的转发规则没有更改,流 f 依旧按照 p_1 路径进行传输,节点 2 在接收到来自节点 1 的数据包后,依旧会按照旧的转发规则将数据包向节点 3 转发,链路 $e(2,3)$ 的中断将导致数据包无法到达节点 3,从而导致 p_1 路径上的数据包无法到达目的节点 4,造成严重的数据包丢失。此时,在流量工程周期内,控制器将会根据当前网络拓扑及流量需求计算出全网新的最优转发路径,如图 1(b)中的虚线所示,计算出的新转发路径集合为 $P_n(f)=\{p_1',p_2',p_3'\}$,$p_1'$、$p_2'$、$p_3'$ 为图 1(b)中虚线箭头所指路径。图 1(b)中的实线部分构成初始网络状态 G_o,虚线部分构成最终网络状态 G_n。

网络更新就是将网络状态从 G_o 转换到 G_n 的中间过程,即将流量从 G_o 中的旧转发路径 $P_o(f)$ 迁移到 G_n 中的新转发路径 $P_n(f)$ 上的中间过程[20]。

2 更新策略

为解决由于网络拓扑变化导致的初始网络状态中部分可用路径失效的问题,本文基于 CM 算法提出 CM-TC 更新策略。本节首先描述 CM-TC 更新策略的流程,再对策略中的相关算法进行详细介绍。

2.1 CM-TC 更新策略描述

CM-TC 更新策略流程如图 2 所示。

阶段 1 控制器根据收集到的当前网络拓扑信息,检测失效路径集合 P_a,以及受失效路径影响的流集合 F_a。若失效路径存在,则执行阶段 2,若无失效路径,则执行阶段 4。

阶段 2 将每条受失效路径影响的流 $f^*\in F_a$ 重路由到未失效的路径集合 $P_o(f^*)-P_a(f^*)$ 上,控制器执行重路由机制计算未失效路径上的流量分配,下发指令修改入口交换机流表中的路径流量分配权重 $x_{f,p}$。同时在网络中添加每条流的新转发路径的流表规则,删除失效路径的流表规则。

阶段 3 在新转发路径的流表规则添加成功之后,若网络中存在拥塞,则执行贪婪流迁移算法,具体算法细节将在 2.2 节进行详细阐述。

阶段 4 执行最小化瞬时拥塞更新算法,通

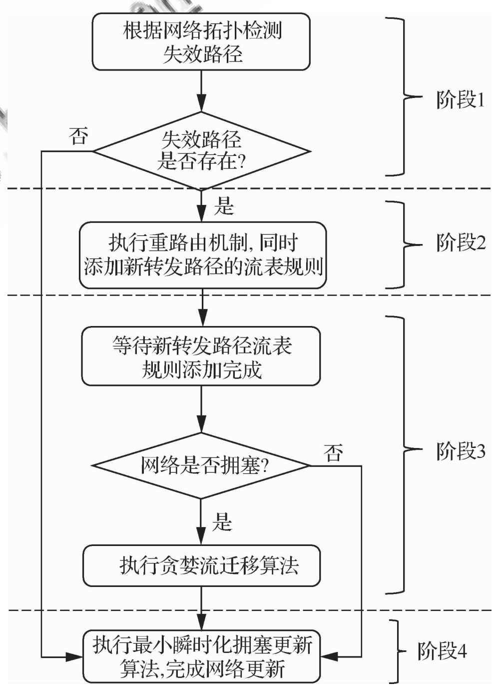

图 2 CM-TC 更新策略流程

Fig. 2 Process of CM-TC update strategy

过线性规划计算出给定数目的中间网络状态,将

网络从初始网络状态逐个转变为最终状态 G_n，完成网络更新。

控制器周期性地执行网络更新策略，首先执行阶段1进行失效路径检测，当检测到旧转发路径中存在部分路径失效时，即按照策略阶段顺序执行。若无路径失效，即无需执行阶段1和阶段2，直接执行阶段4的最小化瞬时拥塞更新算法。

2.2 算法描述

本节配合图3和图4对CM-TC更新策略中的相应算法进行描述。

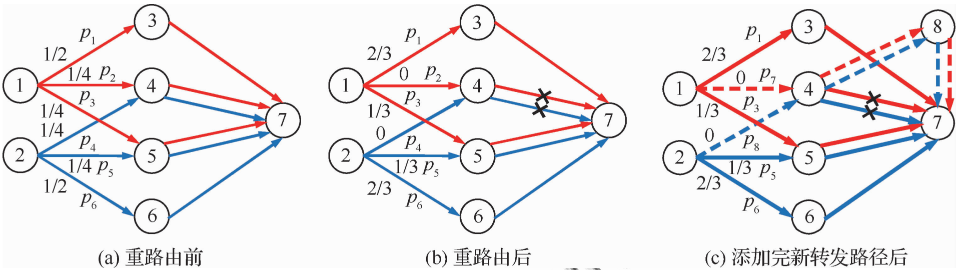

图3 阶段2中的网络状态

Fig. 3 Network state in the second stage

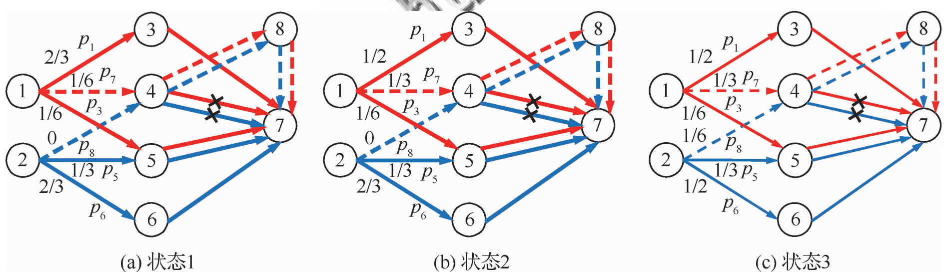

图4 阶段3中的网络状态

Fig. 4 Network state in the third stage

2.2.1 重路由机制

当旧转发路径中存在失效路径时，更新策略进入阶段2。此时新转发路径的流表规则还没有添加，由于每条业务流新转发路径的流表规则添加需要在该路径上的所有交换机上添加新的流表规则，同时受影响的流数目较多，因此流表规则的添加需要一定时间，在该段时间内网络只能按照旧转发路径传输。考虑到继续在失效路径上传输业务会导致严重的数据包丢失，首先要将受影响的流在失效路径上的流量转移到未失效路径上，即使这可能产生一定的拥塞，但仍优于在失效路径上继续传输带来的直接数据包丢失。同时为了尽可能地减少拥塞，从全局出发重新计算受影响的流在未失效路径上的流量分配权重，从而最大限度地最小化网络最大链路拥塞。因此，问题形式化为

$$
\begin{aligned}
&\text{minimize} \quad \max_{e\in E} \mu_e \\
&\text{s. t.} \\
&\begin{cases}
\sum\limits_{p\in P_o(f)} x_{f,p} = 1 & \forall f \in F \\
x_{f,p} = 0 & \forall f\in F_a,\ \forall p\in P_a(f) \\
x_{f,p} \geqslant x^o_{f,p} & \forall f\in F_a,\ \forall p\in P_o(f)-P_a(f) \\
x_{f,p} = x^o_{f,p} & \forall f\in (F-F_a),\ \forall p\in P_o(f)
\end{cases}
\end{aligned}
\tag{1}
$$

线性规划式(1)的目标函数，即最小化网络最大链路利用率。约束条件第1条表示网络流量需求守恒；第2条表示受路径失效影响的流 $f\in F_a$ 在失效路径 $p\in P_a(f)$ 上的权重为0，其中 $x_{f,p}$ 表示重路由后流 f 分配到路径 p 上的流量权重。第3条表示流的多路径路由约束，由于重路由机制只将受影响流在失效路径上的流量进行重路由，未失效路径上的并不变动，因此受影响流重路由之后的流量分配权重将大于等于其旧的网络状态中的流量分配权重，其中 $x_{f,p}$ 表示旧的网络状

态中的流 f 分配到路径 p 上的流量权重;第 4 条表示未受影响的流的流量分配权重保持不变。通过标准线性规划求解器对线性规划式(1)进行求解即可计算出重路由之后转发路径上的流量分配,即重路由之后的网络配置。

如图 3(a)中,每条链路的容量为 1,节点 1～节点 7 的流 f_1 通过红色的 3 条路径进行传输,节点 2～节点 7 的业务流 f_2 通过蓝色的 3 条路径进行传输。2 条业务流的需求都为 2,路径上的流量划分见图中标注。当节点 4～节点 7 之间的链路故障时,检测到 p_2 和 p_4 这 2 条路径失效后,执行重路由机制,将其上的流量重路由到未失效的其他路径上,通过线性规划计算出的流量划分如图 3(b)所示。在图 3(c)中,虚线表示的 p_7 和 p_8 为添加的新路径,p_2 和 p_4 这 2 条失效路径已被删除。

2.2.2 贪婪流迁移算法

在执行完重路由机制且新转发路径的流表规则已添加成功后,更新策略进入阶段 3。尽管阶段 2 中的重路由机制减少了数据包的直接丢失,但可能会使网络产生一定程度的拥塞。CM 算法利用短暂的网络瞬时拥塞换取网络更新时间的减少,在更新期间会产生瞬时拥塞,而执行完重路由机制后的网络是存在一定拥塞的,若直接执行 CM 算法将网络转变到最终网络状态,难以达到 CM 算法原有的效果,与 CM 算法在常规网络更新中达到的效果相比,将会在网络更新期间产生更大程度的拥塞。因此需要在执行 CM 算法之前尽可能地通过流量迁移降低网络拥塞。

针对重路由机制带来的网络拥塞问题,在阶段 3 中设计贪婪流迁移算法以降低网络拥塞程度,如算法 1 所示。首先找出链路利用率最大的链路 e^*,若该链路存在拥塞,则将经过该链路的所有业务流添加到集合 F^* 中;然后对于集合 F^* 中的每一条业务流 f,计算其转发路径中的剩余容量 $C_r(f,p)$;之后找出剩余容量最大的流 f^* 以及对应的剩余容量最大的路径 p^*;再将流 f^* 经过链路 e^* 的路径上的流量迁移到具有最大剩余容量的路径 p^* 上;循环上述过程,直至控制器检测到网络无拥塞或无法找到具有剩余容量的路径时终止算法。

对应图 3(c)中,首先找出拥塞最大的链路 $e(5,7)$,经过该链路的流为 f_1 和 f_2,剩余容量 $C_r(f_1,p_7)=C_r(f_2,p_8)=1$,因此任选其一进行迁移即可。选择将 f_1 在 p_3 上的流量部分迁移到 p_7 上,流量划分如图 4(a)所示。然后找出图 4(a)中拥塞最大的链路 $e(3,7)$,经过该链路的流为 f_1,剩余容量 $C_r(f_1,p_7)=2/3$,因此将 f_1 在 p_1 上的部分流量迁移到 p_7 上,流量划分如图 4(b)所示。再找出图 4(b)中拥塞最大的链路 $e(6,7)$,经过该链路的流为 f_2,剩余容量 $C_r(f_2,p_8)=1/3$,因此将 f_2 在 p_6 上的部分流量迁移到 p_8 上,流量划分如图 4(c)所示。此时网络无拥塞,算法终止。

算法 1:贪婪流迁移算法

输入:链路集合 E;流集合 F;链路容量 C;路径集合 P;流需求集合 d_f;当前网络配置 $x_{f,p}$;链路利用率集合 μ

输出:贪婪流迁移后的网络流量配置 $x_{f,p}$

While $\mu_{\max}>1$ **do**;网络存在拥塞开始循环
  $F^*=\Phi,C_r=\Phi,e^*=\operatorname{argmax}(\mu)$ 进行变量初始化
  for each $f\in F$ **do**
    if 流 f 的转发路径 p 经过链路 e^* **then**
      将流 f 添加到集合 F^* 中
      标记流 f 经过 e^* 的路径 $p_c(f)=p$
    end if
  end for
  for each $f\in F^*$ **do**
    for each $p\in P(f)$ **do**
      if $\forall e\in p$ 都满足 $\mu_e<1$ **then**
        计算流 f 在路径 p 上的剩余容量 $C_r(f,p)$
      end if
    end for
  end for
  if $C_r==\Phi$ **then**
    break
  end if
  $(f^*,p^*)=\operatorname{argmax}(C_r(f,p))$;找出空闲容量最大的流 f^* 以及对应的路径 p^*
  $T=\min(d_{f^*}\cdot(x_{f^*,p_c(f^*)}-1),C_r(f^*,p^*))$;计算待迁移的流量值 T
  $x_{f^*,p^*}=x_{f^*,p^*}+T/d_{f^*}$;增加流 f^* 在路径 p^* 上的流量权重
  $x_{f^*,p_c(f^*)}=x_{f^*,p_c(f^*)}-T/d_{f^*}$;减少流 f^* 在路径 $p_c(f^*)$ 上的流量权重
  执行算法 1 更新集合 μ
end while

贪婪流迁移算法每次对拥塞最大的链路上的流量进行迁移,贪婪地占用具有最大剩余容量的

路径，从而减少迁移次数，在较短的时间内尽可能地降低拥塞流的数目、降低最大链路拥塞。拥塞流迁移的实现，通过控制器向该流的入口交换机下发指令修改流表中转发路径上流量分配权重完成。

算法1第1行表示，当网络最大链路利用率$\mu_{\max}>1$时开始循环；第2行进行变量初始化和赋值，其中，e^* 表示网络链路利用率最大的链路，F^* 表示经过链路 e^* 的流的集合，C_r 表示剩余容量。第3～8行将经过拥塞最高的链路 e^* 的流 f 加入到集合 F^* 中，并标记出每条流 $f\in F^*$ 经过链路 e^* 的路径 $p_c(f)=p$；第9～15行计算出每一条流 $f\in F^*$ 的每一条路径 $p\in P(f)$ 的剩余容量 $C_r(f,p)$；第16～18行表示，若任何一条流的任何一条路径都没有剩余容量则终止算法；第19行找出剩余容量最大的流 f^* 及其对应的路径 p^*；第20～22行将流 f^* 在拥塞路径 $p_c(f^*)$ 上的流量部分迁移到存在剩余容量的路径 p^* 上；第23行执行算法1更新链路利用率集合 μ；之后进行循环，直至 $\mu_{\max}<1$ 或在经过 e^* 的流中无法找到存在剩余容量的路径，则结束算法循环。

2.2.3 最小化瞬时拥塞更新算法

在更新策略进入阶段4之后，网络拥塞已经得到较好的缓解，此时采用最小化瞬时拥塞更新算法，计算出给定数目中间状态的网络配置。

假设给定网络更新期间中间状态数目为 n，则网络状态集合为 $S=\{S_0,S_1,\cdots,S_{n+1}\}$，其中 S_0 表示初始网络状态，S_{n+1} 表示最终网络状态，$S_i(i\in\{1,2,\cdots,n\})$ 表示第 i 个中间状态，$S_i\rightarrow S_{i+1}$ 表示网络从第 i 个状态转变为第 $i+1$ 个状态的更新过程。

定义1 最小化瞬时拥塞更新

已知 S_0 和 S_{n+1} 状态下的网络配置，计算 n 个中间状态的网络配置，从而最小化网络更新期间的瞬时拥塞。

定义2 瞬时拥塞

对于任意的 $S_i\rightarrow S_{i+1}$ 期间，链路 e 上的瞬时拥塞 μ_e^i 定义为 $S_i\rightarrow S_{i+1}$ 期间可能经过链路 e 的最大流量与链路容量 C_e 的比值，即

$$\mu_e^i=\sum_{f\in F}d_f\sum_{e\in p:p\in P_o(f)\cup P_n(f)}\max(x_{f,p}^i,x_{f,p}^{i+1})/C_e \tag{2}$$

因此最小化瞬时拥塞更新算法可以形式化为

$$\begin{aligned}&\text{minimize}\quad \max_{e\in E,\ i\in\{0,1,\cdots,n\}}\mu_e^i\\&\text{s. t.}\\&\begin{cases}\sum\limits_{f\in F}d_f\sum\limits_{e\in p:p\in P_o(f)\cup P_n(f)}\max(x_{f,p}^i,x_{f,p}^{i+1})\leqslant\mu_e^iC_e & \forall e\in E,\ \forall i\in\{0,1,\cdots,n\}\\ \sum\limits_{p\in P_o(f)\cup P_n(f)}x_{f,p}^i=1 & \forall f\in F,\ \forall i\in\{1,2,\cdots,n\}\\ x_{f,p}^i=0 & \forall f\in F_a,\ \forall p\in P_a(f),\ \forall i\in\{1,2,\cdots,n\}\\ x_{f,p}^i\geqslant 0 & \forall f\in F,\ \forall p\in P_o(f)\cup P_n(f),\ \forall i\in\{1,2,\cdots,n\}\end{cases}\end{aligned} \tag{3}$$

线性规划式(3)的目标函数，即最小化网络最大瞬时链路利用率。约束条件的第1条表示在任意的 $S_i\rightarrow S_{i+1}$ 期间，任意链路 e 在最严重的情况下的拥塞程度不超过 μ，即链路 e 上待减少的流量还没有减少，但待增加的流量已经增加了；第2条表示网络流量需求守恒；第3条表示受路径失效影响的流 $f\in F_a$ 在失效路径 $p\in P_a(f)$ 上的权重为0，约束第4条表示流的多路径路由约束。通过求解线性规划式(2)即可计算出最小化网络拥塞更新期间 n 个中间状态对应的网络配置。

3 仿真结果及分析

本文的仿真实验运行在CPU为Intel Xeon，主频为3.2 GHz，内存16 GB的商用服务器上。本文采用的仿真软件为Exata 5.1网络仿真软件。EXata是针对新型无线通信技术而设计的半实物网络仿真平台，精确程度与真实网络相媲美，采用先进的并行算法，能够仿真上千个节点的大

型无线网络,特别适合集群式计算系统的复杂仿真项目。本节基于 EXata 网络仿真平台对本文提出的 CM-TC 网络更新策略进行仿真分析。同时与 One Shot、SWAN、CM 等网络更新算法进行比较,从而验证分析其性能。

3.1 仿真参数设置

仿真参数设置如表 1 所示。在仿真区域内随机生成网络节点,仿真场景区域根据航空集群作战范围设定为 1 000 km×1 000 km,节点数目由小规模到大规模集群在 20~100 范围内变化,根据节点通信半径生成网络拓扑图,节点通信半径为当前航空数据链系统的稳定传输距离 200 km 每个节点随机移动,节点移动速率按照作战飞机巡航速度从低速到高速在 200~1 200 km · h^{-1} 的范围内变化。然后在节点间随机生成业务流,其需求服从均值为 5、方差为 1 的正态分布。控制器根据全网视图从全局出发为每条业务流计算出多条边不相交的路径,将每条业务流拆分到多条路径进行传输,通过线性规划计算流量划分,并将转发规则下发到每个节点,节点根据转发规则进行流量转发、处理。而后在每一个流量工程周期内根据当前网络拓扑重新计算新转发路径,执行网络更新策略。

仿真选择对比的算法描述如下:

1) SWAN。该算法为无拥塞更新算法,将初始网络状态和最终网络状态作为输入,通过线性规划,计算出一系列的中间状态,从而实现无拥塞网络更新。由于该算法需要为每条链路预留一定的带宽,不会产生拥塞,在本场景中只进行流表规则数目和丢失数据包数目的比较。

表 1 仿真参数设置

Table 1 Simulation parameter setting

仿真参数	数值
场景区域/(km×km)	1 000×1 000
节点数目/个	20~100
节点通信半径/km	200
节点移动速率/(km · h^{-1})	200~1 200
业务流数目/条	40~200
业务流需求/Mbps	$N(5,1)$
链路容量/Mbps	20

2) One Shot。该算法是指直接由控制器向网络中的所有交换机节点下发所有的流表,所有交换机同时更新,没有中间状态过渡,直接由初始网络状态更新到最终网络状态。

3) CM。该算法以给定的中间状态数以及网络的初始和最终状态作为输入,通过线性规划,求解出一系列中间网络状态,从而实现给定中间状态数情况下的最小拥塞更新。

3.2 仿真结果分析

图 5 为网络更新期间丢失数据包数目的比较,可以看出 One Shot、SWAN、CM 这 3 种算法的丢包率远远高于 CM-TC。由于该场景中网络节点随机移动,网络拓扑动态变化,当网络拓扑发生变化时,初始网络状态中的部分路径将会失效,继续在其上传输业务将会丢失大量数据包。而 One Shot、SWAN、CM 并没有考虑到网络拓扑的变化,3 种算法随着业务流速率的增大将会丢失大量数据包。其中,由于 One Shot 直接从初始网络状态同时更新所有交换机到最终网络状态,导致网络过度拥塞,因此丢失了大量数据包。CM 和 SWAN 都采用了中间状态进行过渡,CM 中拥塞较小、SWAN 中无拥塞,但由于需要相对较长的更新时间而导致失效路径上的数据包丢失较多。而 CM-TC 采用了重路由机制,将受影响的业务流分配到失效路径上的流量,线性规划后重路由到其他的可用路径上,大大减少了更新数据包的丢失。相比于 SWAN 和 CM 算法,CM-TC 降低了约 81.32%的数据包丢失。

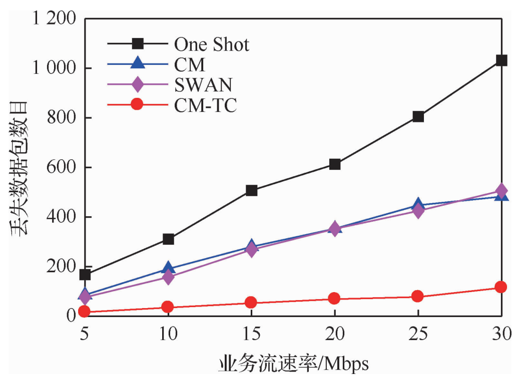

图 5 丢失数据包数目比较

Fig. 5 Comparison of number of lost packets

图 6 为网络更新过程中拥塞链路的瞬时链路利用率比较,SWAN 为无拥塞更新算法,因此这里不做比较。CM 算法和 CM-TC 算法中,设定中间状态数目为 3。从仿真结果可以看出,One Shot 算法的最大瞬时链路利用率达到了将近 200%,正是由于 One Shot 直接同时更新网络中所有交换机所导致的严重网络拥塞。而 CM 和 CM-TC 算法都大大降低了网络更新中出现的瞬时拥塞,最大瞬时链路利用率只有大约 130%,与 One Shot 算法相比,最大瞬时拥塞降低了约 30.73%。在实际运行的网络中,对弹性流量和优先级较低的业务流而言,短暂的速率降低或流量拥塞是可以忍受的,因此 CM 和 CM-TC 算法带来的瞬时链路拥塞是可以接受的。CM-TC 相比于 CM 算法效果稍稍有所降低,这是由于 CM-TC 采用的重路由机制在大大减少数据包丢失的情况下也带来了一定的网络拥塞,但两者基本处于同一水平,CM-TC 算法在降低网络更新期间的瞬时拥塞方面仍然表现出较好的水平。

图 7 为网络更新过程中产生拥塞的业务流数目,拥塞业务流的数目随着业务流总数目的增加基本呈现线性增加。由于 One Shot 算法没有中间过渡状态,同时更新网络中的所有交换机,导致网络中将近 85%的业务流产生拥塞。与之相比,CM 和 CM-TC 算法避免了网络更新期间大部分业务流的拥塞,只有接近 30%的业务流受到影响。CM-TC 相较于 CM 算法,拥塞业务流数目有所上升,但差距并不明显,说明由重路由机制带来的网络拥塞经过贪婪流迁移算法的处理后有了较大的改善,效果较好。

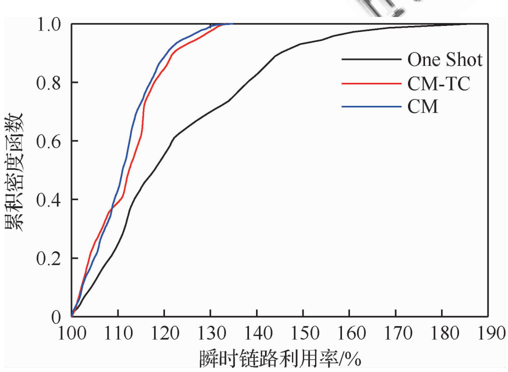

图 6 瞬时链路利用率比较

Fig. 6 Comparison of instantaneous link utilization

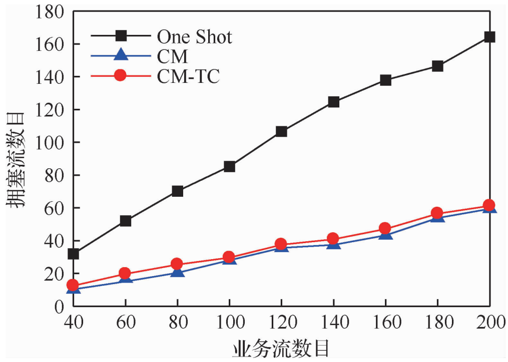

图 7 拥塞业务流数目比较

Fig. 7 Comparison of congested flow numbers

图 8 为网络更新期间产生的流表规则开销,即添加、修改和删除的流表规则数目。可以看出,One Shot 算法对应的流表规则开销最低,SWAN 算法对应的流表规则开销远远高于 CM 与 CM-TC。One Shot 算法没有中间状态过渡,只需要添加、修改和删除必要的流表,因此具有最低的流表规则开销。与之相反,SWAN 算法为了实现无拥塞的更新需要更多数目的中间状态进行过渡,带来了最多的流表规则开销。CM 与 CM-TC 不追求无拥塞更新,旨在实现限定中间状态数目情况下的最小拥塞更新,只需要较少数目的中间状态即可完成网络更新,从而实现了较小的流表规则开销。与 SWAN 算法相比,CM-TC 降低了约 41%的流表规则开销;与 CM 算法相比,仅仅增加了 5%左右的流表规则开销。

根据以上仿真结果,可以看出,CM 和 CM-TC 算法能够有效简化网络更新问题,牺牲较少的网络性能换取更新速度提高的同时,减少了流表规则开销。本文提出的CM-TC算法在网络拓

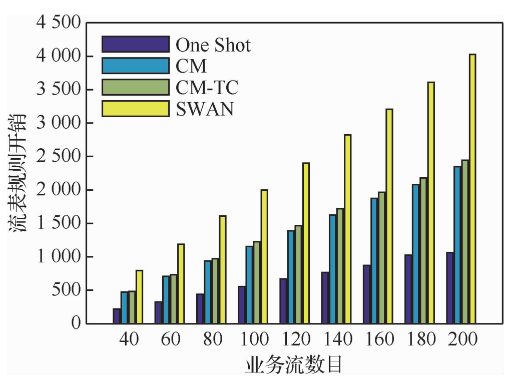

图 8 流表规则开销比较

Fig. 8 Comparison of overhead of rules

扑动态变化的航空集群场景中，与传统拥塞一致性更新算法相比，显著减少了网络更新期间的数据包丢失。虽然采用了重路由机制可能产生一定的拥塞，但本文进一步设计的贪婪流迁移算法有效降低了网络更新期间的瞬时拥塞，减少了拥塞流数目。与 CM 算法相比，在规则开销、网络拥塞方面仍然表现较好，同时显著降低了网络更新期间的数据包丢失。

4 结 论

1）提出的 CM-TC 算法对拓扑变化进行考虑，在更新之前采用重路由机制进行处理，与其他拥塞一致性更新算法相比，大大降低了丢包率，减少了拓扑变化导致的网络更新期间的严重数据包丢失。

2）针对重路由机制导致的拥塞上升进一步设计了贪婪流迁移算法，有效地降低了网络更新期间的瞬时拥塞。

3）CM-TC 用规则开销、网络拥塞方面的少量提高换取了网络更新期间数据包丢失的显著降低，适应航空集群场景的网络更新需求。

参 考 文 献

[1] 梁晓龙，何吕龙，张佳强，等. 航空集群构型控制及其演化方法[J]. 中国科学：技术科学，2019，49(3)：277-287.
LIANG X L, HE L L, ZHANG J Q, et al. Configuration control and evolutionary mechanism of aircraft swarm[J]. Scientia Sinica Techologica, 2019, 49(3): 277-287 (in Chinese).

[2] 梁一鑫，程光，郭晓军，等. 机载网络体系结构及其协议栈研究进展[J]. 软件学报，2016，27(1)：96-111.
LIANG Y X, CHENG G, GUO X J, et al. Research progress on architecture and protocol stack of the airborne network[J]. Journal of Software, 2016, 27(1): 96-111 (in Chinese).

[3] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: Enabling innovation in campus networks [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[4] KREUTZ D, RAMOS F M V, ESTEVES V P, et al. Software-defined networking: A comprehensive survey [J]. Proceedings of the IEEE, 2014, 103(1): 10-13.

[5] MONSANTO C, REICH J, FOSTER N, et al. Composing software defined networks[C]// 10th USENIX Symposium on Networked Systems Design and Implementation, 2013: 1-13.

[6] 吕娜，刘创，陈柯帆，等. 一种面向航空集群的集中控制式网络部署方法[J]. 航空学报，2018，39(7)：321961.
LYU N, LIU C, CHEN K F, et al. A method for centralized control network deployment of aeronautic swarm[J]. Acta Aeronautica et Astronautica Sinica, 2018, 39(7): 321961 (in Chinese).

[7] 赵尚弘，陈柯帆，吕娜，等. 软件定义航空集群机载战术网络[J]. 通信学报，2017，38(8)：140-155.
ZHAO S H, CHEN K F, LYU N, et al. A software defined airborne tactical network for aeronautic[J]. Journal on Communications, 2017, 38(8): 140-155 (in Chinese).

[8] FOERSTER K T, SCHMID S, VISSICCHIO S. Survey of consistent software-defined network updates[J]. IEEE Communications Surveys & Tutorials, 2018, 21(2): 1435-1461.

[9] REITBLATT M, FOSTER N, REXFORD J, et al. Abstractions for network update[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 323-334.

[10] JIN X, LIU H H, GANDHI R, et al. Dynamic scheduling of network updates[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 539-550.

[11] KOMAJWAR S, KORKMAZ T. Challenges and solutions to consistent data plane update in software defined networks[J]. Computer Communications, 2018, 130: 50-59.

[12] LUDWIG A, MARCINKOWSKI J, SCHMID S. Scheduling loop-free network updates: It's good to relax! [C]// Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing. New York: ACM, 2015: 13-22.

[13] MAHAJAN R, WATTENHOFER R. On consistent updates in software defined networks[C]// Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks. New York: ACM, 2013: 1-7.

[14] MATTOS D M F, DUARTE O C M B, PUJOLLE G. Reverse update: A consistent policy update scheme for software-defined networking[J]. IEEE Communications Letters, 2016, 20(5): 886-889.

[15] KATTA N P, REXFORD J, WALKER D. Incremental consistent updates[C]// Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2013: 49-54.

[16] HONG C Y, KANDULA S, MAHAJAN R, et al. Achieving high utilization with software-driven WAN[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 15-26.

[17] ZHENG J, XU H, CHEN G, et al. Congestion-minimizing network update in data centers[J]. IEEE Transactions on Services Computing, 2016, 12(5): 800-812.

[18] WANG W, HE W, SU J, et al. Cupid: Congestion-free consistent data plane update in software defined networks

[C]//IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2016: 1-9.

[19] ZHENG J, XU H, ZHU X, et al. We've got you covered: Failure recovery with backup tunnels in traffic engineering[C]//2016 IEEE 24th International Conference on Network Protocols (ICNP). Piscataway: IEEE Press, 2016: 1-10.

[20] FORSTER K T, WATTENHOFER R. The power of two in consistent network updates: Hard loop freedom, easy flow migration[C]//2016 25th International Conference on Computer Communication and Networks (ICCCN). Piscataway: IEEE Press, 2016: 1-9.

(责任编辑：李丹)

Congestion-minimization network update strategy for topology changes

LYU Na[1], CHEN Kun[1,*], CHEN Kefan[2], ZHU Haifeng[1], PAN Wu[1]

1. *College of Information and Navigation, Air Force Engineering University, Xi'an 710077, China*
2. *Unit 94860 of PLA, Nanjing 210000, China*

Abstract: The emergence of software-defined networking has accelerated the development of airborne networks of aviation swarm, while also presenting challenges for their consistency update. To solve the failure problem of available paths during network update caused by topology changes, a congestion-minimization network update strategy for topology changes is proposed. Firstly, to reduce the packet loss caused by the failure of available paths, the rerouting mechanism is used for processing before the update. Then, for the network congestion caused by the rerouting mechanism, the greedy flow migration algorithm is designed to reduce the network congestion. Finally, the entire network update is completed by the instantaneous congestion minimization update algorithm. The simulation results show that compared with the traditional congestion consistency update algorithm, the update strategy proposed in this paper allows a small increase in the overhead of rules and instantaneous network congestion in exchange for a significant reduction in packet loss during network update.

Keywords: aviation swarm; airborne networks; software-defined networking; network update; topology changes; capacity consistency

Received: 2019-11-18; **Revised**: 2019-12-16; **Accepted**: 2020-02-27; **Published online**: 2020-04-07 11:08
URL: http://hkxb.buaa.edu.cn/CN/html/20200720.html
Foundation item: National Natural Science Foundation of China (61703427)
* **Corresponding author.** E-mail: 648315977@qq.com