软件定义航空集群机载网络自适应更新策略

邹鑫清[1]，吕 娜[1]，陈柯帆[1]，胡诗骏[2]
(1. 空军工程大学信息与导航学院，西安，710077；2.空军装备部上海局驻芜湖地区军代表室，安徽芜湖，241000)

摘要 软件定义网络的出现为突破当前机载网络在航空集群作战应用中存在的固有技术瓶颈开辟了全新途径，然而软件定义机载网络动态的网络拓扑以及有限的链路容量导致网络更新过程中链路拥塞现象频发，造成网络更新过程极易引发网络拥塞，并降低网络更新的成功率。针对该问题，提出一种基于拥塞避免的软件定义航空集群机载网络更新策略。首先通过混合更新策略计算得到各业务流的初始更新操作序列；然后提出拥塞链路感知算法，实现更新过程中对潜在拥塞链路的感知；最后提出基于拥塞避免的软件定义机载网络更新算法，根据各业务流的初始操作序列以及感知到的潜在拥塞链路状态，计算无拥塞更新操作约束，并最大限度实现网络的无拥塞更新。仿真结果表明，与现有网络更新策略相比，所提更新策略能够有效避免网络更新过程中软件定义机载网络的拥塞，提升网络更新的成功率。

关键词 航空集群；机载网络；软件定义网络；自适应更新；拥塞避免
DOI 10.3969/j.issn.1009-3516.2021.02.006
中图分类号 TN820 **文献标志码** A **文章编号** 1009-3516(2021)02-0034-08

An Adaptive Update Strategy for Software-Defined Airborne Network of the Aviation Swarm

ZOU Xinqing[1], LYU Na[1], CHEN Kefan[1], HU Shijun[2]
(1. Information and Navigation College, Air Force Engineering University, Xi'an 710077, China;
2. Shanghai Bureau Military Representative Office in Wuhu, Air Force Equipment Department, Wuhu 241000, Anhui, China)

Abstract: Aimed at the problems that the dynamic network topology and the limited link capacity lead to frequent network link congestion phenomenon in the process of updating, and the network update process is easy to cause network congestion, reducing the network update success rate, this paper proposes a software-defined network update strategy based on congestion avoidance. Firstly, the initial update operation sequence of each data stream is calculated by the mixed update strategy, and then a congestion link sensing algorithm is proposed to realize the potential congestion link sensing in the update process. Finally, a new algorithm based on congestion avoidance is proposed to calculate the congestion free update operation constraint according to the initial operation sequence of each data stream and the perceived potential congestion link state, and to realize the maximum congestion free update of the network. The simulation results show that compared with the existing network updating strategy, the proposed updating strategy can effectively avoid the network congestion of the software-defined airborne network during the network updating

收稿日期：2019-08-31
作者简介：邹鑫清(1995—)，女，福建福州人，硕士，研究方向：军事航空通信。E-mail：xinqingzou127@foxmail.com

引用格式：*邹鑫清，吕娜，陈柯帆，等. 软件定义航空集群机载网络自适应更新策略[J]. 空军工程大学学报(自然科学版)，2021，21(2)：34-41. ZOU Xinqing，LYU Na，CHEN Kefan，et al. An Adaptive Update Strategy for Software-Defined Airborne Network of the Aviation Swarm [J]. Journal of Air Force Engineering University (Natural Science Edition)，2021，21(2)：34-41.*

process and improve the success rate of network updating.

Key words: aviation swarm; airborne network; software-defined networking (SDN); adaptive update; congestion avoidance

近年来，随着现代战争逐步趋向信息化和网络化，航空作战平台的更新换代正向智能化快速发展。为了更好地协调、重组各类航空作战平台，以积极应对未来未知战场环境带来的挑战，衍生于生物集群的航空集群概念被提出[1]。航空集群是由规模一定、功能多样的有人/无人航空平台组成，集群成员间通过简单、智能和高效的协作完成复杂任务。

机载网络作为航空平台间信息交互的枢纽，是航空集群成员间进行灵巧协同配合的重要基础。但长期以来，机载网络从传统的“链”（航空数据链）发展到“网”（航空自组织网），都是针对特定作战场景所设计，仅能满足特定作战任务的需求，且网络自身的灵活性、可扩展性及互操作性较差，难以满足未来网络中心战作战环境中不同系统间的信息交互需求[2]。

软件定义网络（software-defined networking，SDN）作为近几年发展迅猛的一种新型网络范式[3]，为当前机载网络面临的困境提供了全新的突破方法。SDN 将网络的控制平面和数据平面分离，基于可编程的开放接口，通过逻辑集中的控制器实现对网络更灵活、细粒度地管理[4]。由于 SDN 的优势恰好可以弥补机载网络的缺陷，文献[5]将 SDN 的概念与技术应用于机载网络中，构建了新一代面向航空集群的网络——软件定义航空集群机载网络（software-defined airborne network of the aviation swarm，SDAN-AS），为机载网络未来的发展方向提供了重要参考和借鉴，但却没有对 SDAN-AS 存在的具体关键技术问题进行深入挖掘与研究。

作为 SDN 网络运行的基础，网络更新问题一直受到相关研究者的广泛关注[6]。为保证业务传输的连续性，当网络链路、节点、拓扑等状态发生变化时（如节点故障、链路拥塞等），需要对网络进行适应性调整，如改变传输路径、调整链路负载等，通常称为网络更新。网络更新问题是通信网络的基础性问题，更是 SDN 控制平面的核心问题，直接关系到网络运行效率和业务通信质量[7]。

目前 SDN 网络更新问题的研究普遍基于拓扑相对稳定、链路质量相对可靠的有线网络环境，针对多样性业务传输下的网络拥塞问题研究相应的网络更新策略。对于航空集群，由于其 OODA 的闭环作战需求，SDAN-AS 的业务具有与 SDN 有线网络具有相同的多样性特点；但是航空集群成员移动速度较快、成员相对位置变化频繁，且存在敌方干扰的情况，导致 SDAN-AS 的链路状态高度不稳定，从而加剧了链路拥塞和业务传输的不连续，严重影响 SDAN-AS 的业务转发效率。

针对上述问题，从解决 SDAN-AS 链路拥塞、提升业务转发效率的角度，首先通过 FLIP 混合更新算法对各业务流的更新顺序进行规范，得到表示更新顺序的操作序列；在此基础上，设计链路拥塞的感知算法，并在混合更新算法中加入链路拥塞的感知，对操作序列加以约束条件，进一步改进混合更新算法提出基于拥塞避免的网络更新算法（congestion avoidance based algorithm of update，CA-AU），以增加链路感知能力，减少链路拥塞的发生，从而提高 SDAN-AS 的业务转发效率及网络更新可靠性。

1　相关理论与模型构建

1.1　理论介绍

现有针对 SDN 网络的更新算法主要分为 3 类，分别为：次序更新[8]、两阶段提交更新[9]以及混合更新[10]。其中，次序更新通过计算交换机转发规则的替换顺序来保证网络更新的准确和可靠，当网络规模较小时该方法具有较高的可靠性且更新速度较快、效率较高；两阶段提交更新通过在交换机上同时保留业务流的新、旧转发规则，并在网络入口给业务流打标签，决定业务流的转发规则来保证网络更新的准确和可靠。但研究发现，当网络规模较大、交换机数量较多时，使用次序更新方法可能无法保证网络更新过程的准确和可靠；而当存在交换机的内存不足以额外存储一套新的转发规则时，无法使用两阶段提交更新方法。可以看出，当航空集群规模较大时，次序更新方法不适用；当集群成员存在内存不足的情况时，两阶段提交方法不适用。因此，本文后续基于混合更新方法，加入对链路拥塞的感知，提出适用于航空集群机载网络的更新策略。下面对混合更新算法——FLIP（fast lightweight policy）进行介绍。

算法首先将更新问题分解至每一条业务流，并对每条业务流独立计算各自的操作序列。算法以一个网络更新问题作为输入，具体内容包括：网络初始状态（拓扑、节点和链路的相关信息）、最终状态下的路由以及需要维护的给定策略。算法输出为所有业

务流子操作序列混合所得的操作序列。其中,后一个子序列中的任何操作需在前一个子序列中的所有操作完成后执行。

算法首先提取各操作之间的顺序来保证更新过程中的转发正确性,不出现更新过程中的中间状态无可匹配的下一跳转发节点,或数据包在有限节点间相互转发的情况。其次,提取保留给定策略的约束条件,以保证更新过程中不违背给定的转发规则。当保留给定策略约束条件与部分转发正确性约束条件之间存在矛盾时,需要求解线性规划以确定一组更新开销最低的更新操作序列。

1.2 场景及架构描述

图 1 直观地展示了 SDAN-AS 的基本架构。与 SDN 相同,SDAN-AS 将整个网络在逻辑上由南到北分割成 3 个平面,分别为数据平面、控制平面和应用平面[9]。其中,数据平面由负责数据存储与转发的网元设备组成(简称为数据转发设备)。这些数据转发设备的通信系统通常搭载于航空集群成员中的中小型空中平台(例如战斗机、侦察机、电子干扰机、无人机等)之上。控制平面由负责逻辑集中地掌控网络全局的网络控制器组成。这些网络控制器的通信系统通常搭载于航空集群成员中的大型空中平台(例如预警机和指通机)之上[11]。通过南向接口,网络控制器可以及时地掌握全局网络视图并向底层的数据转发设备下发相关指令消息。通过北向接口,网络控制器能够向上层实时地反馈底层网络状态信息,以便网络应用根据网络的实际情况进行动态调整,达到全网最优。应用平面由网络需要的、用户定义的各种业务与应用组成[12]。

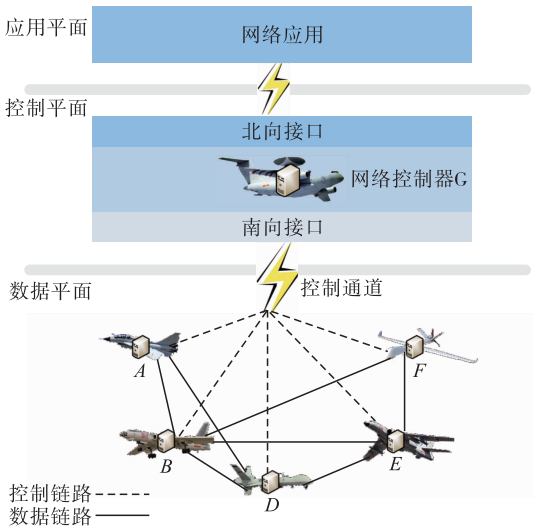

图 1 SDAN-AS 网络架构

为了更加直观地对 SDAN-AS 网络更新问题进行描述,本文将 SDAN-AS 网络建模为一个无向图 $\Omega=(V,E,C)$,其中 $V=\{v_1,v_2,\cdots,v_M\}$ 表示网络拓扑内的全体节点的集合,M 表示拓扑内节点的数量。$E=\{(v_s,v_d)\mid v_s,v_d\in V\}$ 表示网络拓扑中链路的集合;C 表示全体控制器集合,其构成 SDAN-AS 网络的控制平面。显然有 $C\subseteq V$。接下来针对 SDAN-AS 网络更新中的相关变量进行定义:$W=\{w_1,w_2,\cdots,w_m\}$ 表示当前网络更新过程中的待更新节点集合,显然有 $W\subseteq V$,m 表示待更新节点数量;$\Lambda=\{f_1,f_2,\cdots,f_n\}$ 表示待更新业务流集合,n 表示待更新业务流数量。

2 基于拥塞避免的 SDAN-AS 自适应更新策略

2.1 总体策略

图 2 为策略流程,包括更新问题收集阶段(上半部分)和更新操作约束求解阶段(下半部分)。

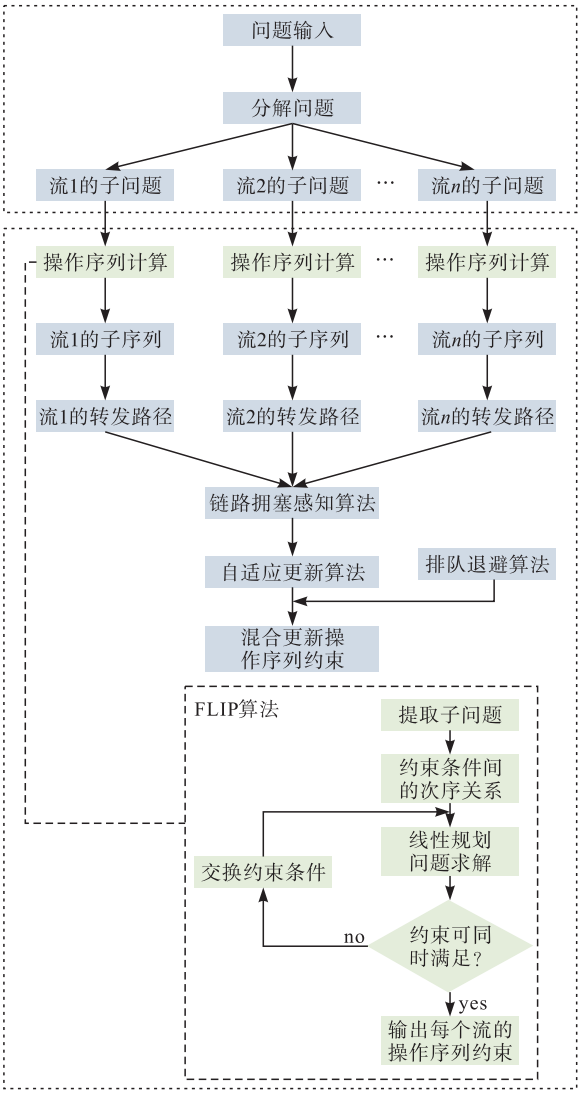

图 2 基于拥塞避免的 SDAN-AS 自适应更新策略流程图

其中,更新问题收集阶段实现了 SDAN-AS 当前网络状态的采集以及更新问题的初步解析。具体

包括①当前网络状态信息的采集。SDAN-AS 网络控制器对当前网络状态信息进行采集，采集的信息包括：网络拓扑、节点和链路的相关信息（包括节点间链路的链路容量、网络中处于传输状态的业务流的优先级以及占用的链路容量等）。②业务流转发路径的采集。控制器通过北向接口，从应用平面获取高层次网络控制逻辑以及网络管控需求，包括网络更新完成后需实现的业务流的转发路径以及更新过程中需保留的给定策略。

更新操作序列求解阶段生成网络更新操作序列，实现 SDAN-AS 网络数据转发策略的更新，并保证更新过程中的一致性。该阶段包括：

1）网络更新问题的分解。根据更新问题收集阶段输入的网络初始转发策略、规划的转发策略、保留的给定策略以及预测的链路状态，提取出每条业务流需要保留的转发规则以及转发正确性需求，作为 FLIP 算法的输入。此外将链路状态信息作为基于拥塞避免的 SDAN-AS 更新算法的输入，具体算法细节将在 2.2 节阐述。

2）连接一致性与策略一致性约束条件计算。FLIP 算法中的基本操作类型包括：①规则替换操作 $rep(s,\ f)$，在节点 s 上对业务流 f 执行的以最终规则替换当前规则的操作；②标签操作 $tag(s,\ f,\theta)$，在节点 s 上对业务流 f 执行对其所有数据包打上标签 θ 的操作；③匹配操作 $match(s,\ f,\theta_i,\theta_f)$。在节点 s 上执行安装初始和最终规则，并对业务流 f 中的新数据包打上标签 θ_f、旧数据包打上标签 θ_i 的操作。当保证连接一致性与策略一致性的约束条件存在矛盾时，需迭代求解线性规划问题：

$$\min \sum_i app(rep(i,f)) + \sum_i app(match(j,f,\theta_1,\theta_2)) \tag{1}$$

$$\text{s.t.} \begin{cases} \text{constrains on } rep(i,f) \\ \forall app(rep(i,f))>0, \forall app(match(j,f,\theta_1,\theta_2))>0 \end{cases}$$

该线性规划问题的目标为寻找一组更新操作序列，使更新开销最低。其中，constrains on $rep(i,f)$ 表示全体替换操作的次序约束，当该线性规划问题无解时，通过标签-匹配操作取代更新序列中部分节点替换操作，从而化解约束矛盾，最终计算得到该线性规划问题最优解更新序列作为算法输出。

因此，根据网络更新的连接一致性与策略一致性需求，基于 FLIP 算法分别计算每条业务流的更新操作序列。首先通过线性函数求解该条业务流的规则切换操作序列 $rep(s,f)$。若该线性函数无实解，则通过基于两阶段提交机制的标签-匹配操作 $tag(s,\ f,\theta)$-$match(s,\ f,\theta_i,\theta_f)$，代替部分节点的 $rep(s,\ f)$ 操作，生成新的操作序列。

3）容量一致性约束条件计算：基于 FLIP 计算得到的业务流操作序列，根据网络更新的容量一致性需求，通过基于拥塞避免的自适应更新算法以及排队退避算法对网络更新的容量一致性约束进行计算，提取新的业务流更新操作次序约束。若现有更新操作无法避免更新过程中产生的拥塞，则进一步通过排队退避算法，将部分低优先级业务流加入拥塞链路前序节点的缓存队列中，待高优先级业务流完成更新后释放缓存恢复转发。

2.2　基于拥塞避免的 SDAN-AS 更新算法

2.2.1　SDAN-AS 网络更新拥塞感知

设一次网络更新过程中，待更新业务流集合为 $\Lambda=\{f_1,f_2,\cdots,f_n\}$，其中，$f_1$ 的更新操作子序列为 $seq_i=\{op_i^{(1)},op_i^{(2)},\cdots,op_i^{(m)}\}$；$F_i=\{F_i^{(0)},F_i^{(1)},\cdots,F_i^{(m)}\}$ 为流 f_i 在更新过程中转发路径的子集合，$F_i^{(j)}=F_i^{(j-1)}\otimes op^{(f)}$，表示更新操作 op^j 执行后的转发路径，特别地，F_i^0 表示 f_i 的初始转发路径；f_i 所占用的链路容量为 vol_i。计算机算法如下：

算法 1　链路拥塞感知算法

输入：F，VOL，CAP

输出：C，Ψ

1）$C=\varnothing$

2）*for* $i=1\rightarrow n$ *do*

3）$Trace_i=F_i^{(0)}\cup F_i^{(1)}\cup\ldots\cup F_i^{(card(F_i))}$

4）*end for*

5）*for* e_k *in* E *do*

6）$vol_{e_k}=0$，$\Psi_{e_k}=\varnothing$，$i=1$

7）*while* $e_k\in Trace_i$　*do*

8）$vol_{e_k}=vol_{e_k}+vol_i$

9）$\Psi_{e_k}=\Psi_{e_k}\cup f_i$

10）$i=i+1$

11）*end while*

12）*if* $vol_{e_k}>cap_{e_k}$

13）$C=C\cup e_k$

14）*end if*

15）*end for*

设计 SDAN-AS 网络更新拥塞感知算法如下。算法输入包括全体待更新业务流路径集合 $F=\{F_i\mid 1\leqslant i\leqslant n\}$、占用的链路容量 $VOL=\{vol_i\mid 1\leqslant i\leqslant n\}$、当前网络中全体链路的容量 $CAP=\{cap_{e_k}\mid e_k\in E\}$，其中 E 表示网络中全体链路集合。算法输出为潜在拥塞链路集合 C 以及各链路经过的业务流集合 Ψ。此外，$Trace_i$ 表示业务流在更新过程中所有转发路径遍历的链路集合。

算法 1 第 1 行对链路拥塞感知集合进行初始化；第 2～4 行计算得到全体业务流在更新过程中所

遍历的链路;第 5～15 行计算链路拥塞状态,其中第 7～11 行计算每条链路上可能经过的业务流量,若超出该条链路的容量,该条链路在更新过程中可能出现拥塞情况,将被加入集合 C,同时将该条链路上可能经过的业务流加入集合 Ψ 中。

2.2.2 基于拥塞避免的自适应更新算法

根据总体更新策略,在获取网络更新过程中的链路拥塞状态后,通过基于拥塞避免的自适应更新算法计算混合更新操作序列约束。

基于拥塞避免的自适应更新算法如下所示。算法输入为当前网络更新流的相关信息 $I=(PRI,SEQ,VOL,F)$ 以及拥塞状态信息 $L=(C,\Psi)$。其中,$PRI=\{pri_i\mid 1\leqslant i\leqslant n\}$ 表示业务流处理优先级集合,$pri_i\in \mathrm{N}^+$ 表示 f_i 的处理优先级,其取值越小,表明当出现拥塞情况时 f_i 具有更高的优先处理权。$SEQ=\{sep_i\mid 1\leqslant i\leqslant n\}$ 表示当前网络更新中全体业务流的更新操作集合,$sep_i=\{op_i^{(1)},op_i^{(2)},\cdots,op_i^{(m)}\}$ 表示 f_i 的更新操作子序列;C 表示经过拥塞感知算法计算得到的潜在拥塞链路集合,Ψ 表示各链路经过的业务流集合。算法输出为混合更新操作序列约束集合 $Cons$。算法中,Ω_i 表示流的更新退避业务流集合,其中保存了 Ψ_{e_k} 中可以通过更新释放链路 e_k 容量的业务流,Γ_i 表示 Ψ_{e_k} 中优先级低于 f_i 的业务流集合。计算机算法如下:

算法 2 基于拥塞避免的自适应更新算法(CA-AU)

输入:网络更新流相关信息 $I=(PRI,SEQ,VOI,F)$

拥塞状态信息 $L=(C,\Psi,CAP)$

输出:混合更新操作序列约束集合 $Cons$

1) $Cons=\varnothing$

2) $\underset{pri_i}{rank}(\Psi_{e_k})$

3) $for\ f_i\ in\ \Psi_{e_k}\ do$

4) $if\ \exists F_i^{(j)}\in F_i,\ s.t.\ e_k\in\{F_i^{(j)}-F_i^{(j-1)}\}$

5) $m=j$

6) $else$

7) $continue$

8) $end\ if$

9) $\Omega_i=\varnothing,\Gamma_i=\{f_x\in\Psi_{e_k}\mid pri_x<pri_i\}$

10) $for\ f_j\ in\ \Gamma_i\ do$

11) $if\ \exists F_j^{(l)}\ s.t.\ e_k\in\{F_j^{(l-1)}-F_j^{(l)}\}$

12) $\Omega_i=\Omega_i\cup f_j$

13) $Cons=Cons\cup\{op_j^l<op_i^m\}$

14) $end\ if$

15) $if\ \sum_{\Psi_{e_k}}vol-\sum_{\Omega_i}vol\leqslant cap_{e_k}$

16) $break$

17) $end\ if$

18) $end\ for$

19) $if\ \sum_{\Psi_{e_k}}vol-\sum_{\Omega_i}vol>cap_{e_k}$

20) $QueueRetreat(\ )$

21) $end\ if$

22) $end\ for$

算法 2 第 1 行首先对混合更新操作序列约束集合 $Cons$ 进行初始化,之后通过第 2～24 行,逐链路添加混合更新操作序列约束。其中,第 4～9 行判断流在更新后的转发路径是否经过当前潜在的拥塞链路,若经过则记录造成拥塞的更新操作序号,否则,继续分析更低一级优先级的业务流更新操作子序列。第 10～19 行依次在优先级低于 f_i 的业务流更新操作子序列中搜索可释放链路容量的更新操作,并添加进混合更新操作序列约束集合 $Cons$ 中。若在 Ψ_{e_k} 中搜索完毕后,e_k 的链路容量仍无法保证无拥塞更新,此时调用排队退避算法 $QueueRetreat()$,对 e_k 上的部分低处理优先级业务流添加排队退避以及恢复转发更新操作约束。本文对排队退避操作以及恢复转发操作定义如下:

定义 1 排队等待操作 $queue(f,v,e)$:在节点 v 上将业务流 f 中通过链路 e 转发的数据包加入缓存队列中,此时业务流 f 暂停在链路 e 上的传输,但不影响流 f 在该节点上的更新。

定义 2 恢复转发操作 $recov(f,v,e)$:在节点 v 的缓存队列中,将属于业务流 f 的数据包通过链路 e 转发。显然,对于一条业务流 f,存在约束:$queue(f,v,e)<recov(f,v,e)$。计算机算法如下:

算法 3 $QuecueRetreat(\ )$

输入:网络更新流相关信息 $I=(PRI,SEQ,VOI,F)$

拥塞状态信息 $L=(C,\Psi,CAP)$

f_i 的更新退避业务流集合 Ω

Ψ_{e_k} 中优先级低于 f_i 的业务流集合 Γ_i

输出:排队退避约束集合 $Cons\_QR_i$

1) $Cons\_QR_i=\varnothing,\Phi_i=\varnothing$

2) $while\ \Gamma_i\neq\varnothing$

3) $if\ \min\limits_{pri_x}\Gamma_i\notin\Omega_i$

4) $\Phi_i=\Phi_i\bigcup\min\limits_{pri_x}\Gamma_i$

5) $\Gamma_i=\Gamma_i-\min\limits_{pri_x}\Gamma_i$

6) $end\ if$

7) $if\ \sum_{\Psi_{e_k}}vol-\sum_{\Omega_i}vol-\sum_{\Phi_i}vol\leqslant cap_{e_k}$

8) $break$

9) $end\ if$

10) $end\ while$

11) $while\ \Phi_i\neq\varnothing$

12) $l=1$

13) $que_i(l)=queue(\max\limits_{pri_x}\Phi_i,v,e_k)$

14) $rec_i(l)=recov(\max\limits_{pri_x}\Phi_i,v,e_k)$

15) $\Phi_i=\Phi_i-\min\limits_{pri_x}\Phi_i$

16) $l=l+1$

17) *end while*

18) $Cons\_QR_i=Cons\_QR_i\cup\{que_i<op_i^m<rec_i\}$

算法 3 输入包括待更新流相关信息 $I=(PRI,SEQ,VOI,F)$；拥塞状态信息 $L=(C,\Psi,CAP)$；流 f_i 的更新退避业务流集合 Ω_i；Ψ_{e_k} 中优先级低于 f_i 的业务流集合 Γ_i。输出为业务流 f_i 的排队退避约束集合 $Cons\_OR_i$。算法中 Φ_i 表示流 f_i 的排队退避业务流集合，数组 que_i 与 rec_i 分别保存了排队退避操作与恢复转发操作，若有 $j<k$，则满足 $que_i[j](rec_i[j])<que_i[k](rec_i[k])$。

算法 3 第 1 行对 $Cons\_QR_i$ 与 Φ_i 进行初始化，第 2～10 行在 Ψ_{e_k} 中依次选择最低优先级的业务流添加进 Φ_i 中，且需要保证所选业务流不属于集合 Ω_i 中。第 7～9 行表示直至 e_k 上的容量可以保证流 f_i 的无拥塞更新时，停止添加。随后通过第 11～18 添加排队退避约束集合，对于 Φ_i 中的业务流，按照优先级由高至低，依次将相应的排队退避操作以及恢复转发操作分别添加进入数组 que_i 以及 rec_i 中，并最终添加保证流 f_i 无拥塞更新的排队退避约束 $\{que_i<op_i^m<rec_i\}$ 进入集合 $Cons\_QR_i$ 中，表示 Φ_i 中的业务流在流 f_i 的更新操作 op_i^m 之前依次执行排队退避操作，从而提前释放链路容量，保证流 f_i 在链路 e_k 上的无拥塞更新。在更新操作 op_i^m 完成之后立即恢复转发。

图 3 给出一个基于拥塞避免的自适应更新算法示例。图中共有 f_1、f_2 与 f_3 3 条待更新业务流，优先级 $pri_1<pri_2<pri_3$，$vol_1=7$，$vol_2=5$，$vol_3=5$，且任意链路的容量为 10；f_1 的初始转发路径与最终转发路径分别为 $F_1=[A,E,F,G,C]$ 与 $F_1'=[A,E,G,F,C]$，其需要保留的转发策略为 $P(f_1)=\{[F,G],[G,F]\}$；f_2 的初始转发路径与最终转发路径分别为 $F_2=[B,E,G,H,D]$ 与 $F_2'=[B,E,H,G,D]$，所需保留的转发策略为 $P(f_2)=\{[G,H],[H,G]\}$；f_3 的初始转发路径与最终转发路径分别为与 $F_3=[A,B,F,C,H]$ 与 $F_3'=[A,B,F,C,D,H]$，所需保留的转发策略为 $P(f_3)=\varnothing$。由无拥塞条件下的更新算法计算得到 f_1、f_2 与 f_3 的更新操作子序列分别为：$seq_1=[rep(A,f_1),tag(F,f_1,\tau),match(G,f_1,\tau,\tau'),rep(E,f_1),rep(F,f_1),rep(G,f_1)]$、$seq_2=[rep(B,f_2),tag(G,f_2,\theta),match(H,f_2,\theta,\theta'),rep(E,f_2),rep(G,f_2),rep(H,f_2)]$、$seq_3=[rep(A,f_3),rep(B,f_3),rep(F,f_3),rep(H,f_3),rep(D,f_3),rep(C,f_3)]$。

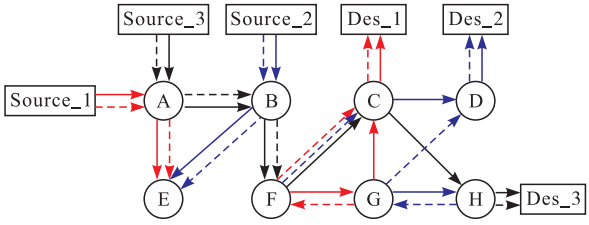

图 3　基于拥塞避免的自适应更新算法示例

经过链路拥塞感知算法可以计算得到潜在的拥塞链路为：$C=\{[E,G],[F,C]\}$，两条链路上经过的业务流集合分别为：$\Psi_{[E,G]}=\{f_1,f_2\}$ 和 $\Psi_{[F,C]}=\{f_1,f_3\}$，然后通过基于拥塞避免的自适应更新算法计算混合更新操作序列的约束。对于潜在拥塞链路 $[E,G]$，首先计算保证高优先级业务流 f_1 的无拥塞更新的约束条件，在 $\Psi_{[E,G]}=\{f_1,f_2\}$ 中搜索低优先级业务流 f_2 的更新操作子序列。其中，操作 $rep(E,f_2)$ 可以在使 f_2 的转发路径从链路 $[E,G]$ 切换为 $[E,H]$，从而释放链路 $[E,G]$ 上的容量，且此时满足 $\sum\limits_{\Psi_{[E,G]}}vol-vol_{f_2}\leqslant cap_{e_k}$，即在执行操作 $rep(E,f_2)$ 后链路上不再产生拥塞。因此，添加混合更新操作序列约束 $rep(E,f_2)<rep(E,f_1)$ 即可保证 f_1 在链路 $[E,G]$ 上进行无拥塞更新。

同理，对于潜在拥塞链路 $[F,C]$，为保证 f_1 的无拥塞更新，在 $\Psi_{[F,C]}=\{f_1,f_3\}$ 中搜索 f_3 的更新操作子序列。不同的是，f_3 在更新前后均经过了链路 $[F,C]$，无法通过更新退避约束实现 f_1 的无拥塞更新。此时调用 *QueueRetreat*( )算法，并添加排队退避约束条件 $queue(f_3,F,[F,C])<rep(F,f_1)<recov(f_3,F,[F,C])$，即在 f_1 的转发路径切换至链路 $[F,C]$ 之前，在节点 F 上将流 f_3 中经过 $[F,C]$ 链路转发的数据包放入缓存队列中，以保证 f_1 在链路 $[F,C]$ 上完成无拥塞更新。f_1 完成更新后，释放缓存，恢复对 f_3 的转发。

3　仿真结果与分析

本节将 CA-AU 更新策略与几种典型的 SDN 更新算法在 SDAN-AS 网络环境下进行仿真分析，并对几类关键性能指标进行比较，验证 CA-AU 的性能。选取的对比更新算法包括次序更新算法、FLIP 混合更新算法以及 zUpdate 更新算法[13]。选取的性能评价指标为更新平均排队时延和平均更新成功率。仿真参数设置如表 1 所示，其中，网络中各业务流所占用的链路容量服从均值为 5，方差为 1 的正态分布；各业务流优先级服从最小值为 1，最大值为 10 的均匀分布。仿真实验参照 Rocketfuel 拓

扑[14],实验过程中可对网络中节点进行增减。当对指定自变量参数进行仿真分析时,其他自变量参数设置为平均值。

表 1 仿真参数

仿真参数	数值
网络仿真区域/km	1 200×1 200
通信半径/km	200
飞行速度/(km/h)	200～1 200
节点数量	20～120
业务流数量	20～120
单位链路容量	100
业务流占用链路容量分布	$N(5,1)$
业务流优先级分布	$U(1,10)$

图 4 为一次成功的网络更新过程中,各待更新业务流的平均排队时延随网络规模变化的趋势。

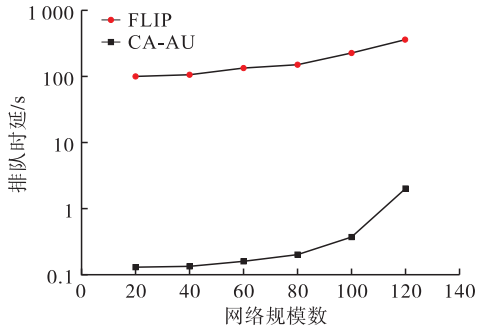

图 4 排队时延与网络规模关系

参与对比的更新策略分别为 CA-AU 策略与 FLIP 混合更新算法,通过对比可以发现,随网络中节点数量的提升,FLIP 与 CA-AU 的排队时延均呈现指数式上升的趋势。而 FLIP 的平均排队时延始终远高于 CA-AU。在网络规模为 20 个节点时,平均时延已接近 100 s,其原因在于 FLIP 算法虽然充分考虑了单条业务流的转发正确性需求与给定策略需求,但没有考虑链路拥塞对网络更新的影响。拥塞现象易造成网络需要通过多次序列计算以及指令下发过程才能最终成功完成更新;相比之下,CA-AU 的更新平均时延显著降低,随网络规模增至 120 个节点,其平均时延始终保持在 10 s 以下。CA-AU 的平均更新排队时延主要来源于低优先级业务流的主动排队退避。

图 5 为一次成功的网络更新过程中,各待更新业务流的平均排队时延随待更新业务流变化的趋势。在该组实验中,网络规模设置为 70 个节点。仿真结果表明,随业务流的增长,各流平均更新排队时延呈现更加急剧的指数增长,原因在于业务流数量的增长直接造成网络中链路拥塞现象的加剧,从而导致 FLIP 算法执行时,需要更多次的更新周期完成一次成功的网络更新;而对于 CA-AU 算法,拥塞链路的增加也造成低优先级业务流需要在更多的拥塞链路的前序节点上进行排队退避。但排队退避所引起的平均排队时延相较于 FLIP 算法仍显著降低。

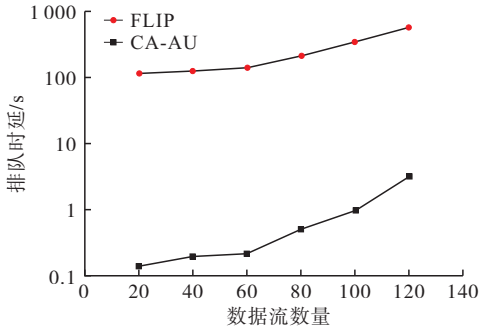

图 5 排队时延与业务流量关系

图 6 为一次成功的网络更新过程中,待更新业务流的平均排队时延随平均飞行速度之间的关系。为验证 CA-AU 在 SDAN-AS 高度动态的网络环境下的适应性,在该组实验中选取了 zUpdate 无拥塞更新策略作为对比。zUpdate 是一种数据中心网络环境下的无拥塞更新方案,通过对比可以发现,随着飞行速度的提升,zUpdate 与 CA-AU 的平均更新排队时延均明显上升,表明随着网络动态性的提升,zUpdate 与 CA-AU 的适应性均有一定程度的下降。其中,zUpdate 的平均排队时延在飞行速度较低时低于 CA-AU,这是因为 zUpdate 可提供无拥塞的更新方案,而 CA-AU 则需要对部分低优先级的业务流进行主动的排队退避。然而随平均飞行速度的提升,zUpdate 的平均排队时延急剧上升,并在飞行速度达到 500 km/h 时开始高于 CA-AU。这是因为 CA-AU 策略通过链路拥塞感知,对潜在的拥塞链路进行了更新约束的计算,相比于适用于静态网络环境的 zUpdate,在动态网络环境下具有更高的容错与适应性能。

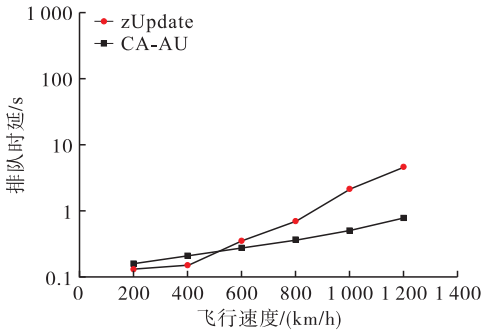

图 6 排队时延与飞行速度关系

图 7 为网络更新成功率随网络规模的变化趋势,该组仿真实验选取次序更新作为对照,在不同规模下的网络中,采用相同的网络拓扑各执行 100 次重复实验。可以发现,随网络规模的提升,次序更新的更新成功率均呈指数式下降,随网络规模的提升,

次序更新与 CA-AU 更新成功率的差距持续增大，当网络规模达到 120 个节点时，更新成功率降至 20%以下。而 CA-AU 的更新成功率虽也随着网络规模的提升而下降，但在网络规模提升至 120 个节点的过程中始终保持在 90%以上。该实验结果表明，在链路状态不稳定的机载网络环境下，基于混合更新的 CA-AU 相比于次序更新的可靠性有了大幅提升，同时也表明相比于次序更新，CA-AU 具有更强的鲁棒性，可在更加复杂的网络环境下获取理想的更新效率。

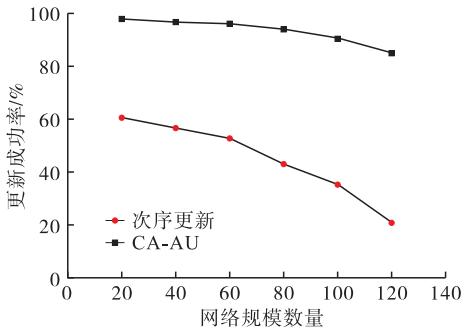

图 7　更新成功率与网络规模关系

4　结语

本文面向软件定义航空集群机载网络的特殊复杂环境，研究适用于该网络的更新策略。首先，基于混合更新算法计算理想链路状态下业务流的更新操作序列，随后针对每条业务流做拥塞链路的感知，并根据链路拥塞情况提出拥塞避免算法，基于业务流优先级生成无拥塞的更新操作约束，以实现网络更新的准确、可靠性。仿真结果表明，本文提出的更新策略在更新过程中降低排队时延的同时，提高了更新的成功率。

参考文献：

[1] 梁晓龙，何吕龙，张佳强，等. 航空集群构型控制及演化方法[J]. 中国科学(技术科学)，2019，49(3)：277-287.

[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.

[3] CHEN T，MATINMIKKO M，CHEN X，et al. Software Defined Mobile Networks：Concept，Survey，and Research Directions [J]. Communications Magazine IEEE，2015，53(11)：126-133.

[4] RAHMAN S U，KIM G H，CHO Y Z，et al. Deployment of an SDN-Based UAV Network：Controller Placement and Tradeoff between Control Overhead and Delay [C]//International Conference on Information and Communication Technology Convergence. [S. l.]：IEEE，2017：1290-1292.

[5] 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.

[6] CHOI J H，MIN S G，HAN Y H. MACsec Extension over Software-Defined Networks for in-Vehicle Secure Communication [C]//Tenth International Conference on Ubiquitous and Future Networks (ICUFN). [S. l.]：IEEE，2018.

[7] HLABISHI I K，ADNAR M. Abu-Mahfouz and Gerhard P. Hancke. A Survey on Software-Defined Wireless Sensor Networks：Challenges and Design Requirements[J]. IEEE Access，2017，2(8)：1872-1899.

[8] FOERSTER K T，SCHMID S，VISSICCHIO S. Survey of Consistent Software-Defined Network Updates [J]. IEEE Communications Surveys & Tutorials，2018：2876749.

[9] REITBLATT M，FOSTER N，REXFORD J，et al. Consistent Updates for Software-Defined Networks：Change You Can Believe In! [C]//ACM Workshop on Hot Topics in Networks. Cambridge，MA，USA：ACM，2011.

[10] VISSICCHIO S，CITTADINI L. Safe，Efficient，and Robust SDN Updates by Combining Rule Replacements and Additions [J]. IEEE/ACM Transactions on networking，2017，25(5)：3102-3115.

[11] 景晓年，梁晓龙，张佳强，等. 航空集群作战编队优化控制研究[J]. 计算机仿真，2017，34(4)：90-94.

[12] 黄韬，刘江，魏亮，等. 软件定义网络核心原理与应用实践[M]. 2 版. 北京：人民邮电出版社，2016.

[13] LIU H H，WU X，ZHANG M，et al. zUpdate：Updating Data Center Networks with Zero Loss[C]//ACM SIGCOMM 2013. [S. l.]：ACM，2013.

[14] REITALATT M，CANINI M，GUHA A，et al. FatTire：Declarative Fault Tolerance for Software-Defined Networks[C]//ACM SIGCOMM Conference. [S. l.]：ACM，2013.

（编辑：姚树峰）