计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(9):106−114 [doi: 10.15888/j.cnki.csa.009222]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于拥塞及内存感知的 SD-WAN 故障恢复①

庄　捷[1,2], 张奇支[1,2], 郑伟平[1,2], 赵淦森[1,2]

[1](华南师范大学 计算机学院, 广州 510631)
[2](广州市云计算安全与测评技术重点实验室, 广州 510631)
通信作者: 张奇支, E-mail: zhangqizhi@m.scnu.edu.cn

摘　要: 在软件定义广域网 (SD-WAN) 中, 链路故障会导致大量丢包, 严重时会引起部分网络瘫痪. 现有的流量工程方法通过在数据平面提前安装备份路径能够加快故障恢复过程, 但在资源受限的情况下难以适应各种网络故障情况, 从而使恢复后的网络性能下降. 为了保证网络在故障恢复之后的性能并减少备份资源的消耗, 本文提出一种基于拥塞及内存感知的主动式故障恢复方案 (CAMA), 不仅能够将受影响数据流进行快速重定向, 还能实现负载均衡避免恢复后潜在的链路拥塞. 实验结果表明, 与已有方案相比, CAMA 能有效利用备份资源, 在负载均衡上有较好的性能, 且仅需少量备份规则即可覆盖所有单链路故障情况.

关键词: 软件定义网络; 软件定义广域网; 故障恢复; 负载均衡; 备份资源

引用格式: 庄捷,张奇支,郑伟平,赵淦森.基于拥塞及内存感知的 SD-WAN 故障恢复.计算机系统应用,2023,32(9):106–114. http://www.c-s-a.org.cn/1003-3254/9222.html

SD-WAN Failure Recovery Based on Congestion and Memory Awareness

ZHUANG Jie[1,2], ZHANG Qi-Zhi[1,2], ZHENG Wei-Ping[1,2], ZHAO Gan-Sen[1,2]

[1](School of Computer Science, South China Normal University, Guangzhou 510631, China)
[2](Key Laboratory on Cloud Security and Assessment Technology of Guangzhou, Guangzhou 510631, China)

Abstract: In software-defined wide area networks (SD-WANs), link failures can result in substantial packet loss, leading to partial network paralysis in severe cases. The existing traffic engineering approaches can expedite failure recovery by installing backup paths in advance on the data plane. However, it is difficult to adapt to various network failures with limited resources, which degrades the network performance after recovery. To maintain the network performance after failure recovery and reduce the consumption of backup resources, this study proposes a proactive failure recovery scheme based on congestion and memory awareness (CAMA), which can not only redirect the affected data flows quickly but also realize the load balancing to avoid the potential link congestion after recovery. Experimental results demonstrate that compared with existing schemes, CAMA can effectively utilize backup resources, performs well in load balancing, and requires only a small number of backup rules to cover all single-link failure scenarios.

Key words: software-defined networking (SDN); software-defined wide area network (SD-WAN); failure recovery; load balancing; backup resource

1　引言

1.1　研究背景及介绍

软件定义网络 (software-defined networking, SDN) 解耦了控制平面和数据平面, 使得逻辑集中的控制器拥有网络拓扑的全局视图, 能够决定底层设备的转发行为并监控各交换机的状态[1]. 利用强大的管控能力,

① 基金项目: 国家重点研发计划 (2019YFB1804003); 广东省重点领域研发计划 (2019B010137003); 广东省科技基金 (2016B030305006, 2018A07071702); 广州市科技基金 (201804010314)
收稿时间: 2023-02-22; 修改时间: 2023-03-22; 采用时间: 2023-03-30; csa 在线出版时间: 2023-07-17
CNKI 网络首发时间: 2023-07-18

控制器能够收集交换机和链路的状态信息, 为网络建立合理的转发方案, 提升网络的服务质量.

由于硬件设备的内部冗余, 节点故障的概率远低于链路故障的概率, 而多链路故障的概率又远低于单链路故障概率[2]. 但即使是单链路故障, 其影响也不容忽视. 软件定义广域网 (software-defined WAN, SD-WAN) 中通常使用流量工程 (traffic engineering, TE), 也就是通过动态路由选择和带宽分配实现网络可用性和有效资源间的平衡. SDN 中两种最主要的故障恢复方法分别是主动式 (proactive) 和反应式 (reactive) 恢复[3]. 反应式恢复方法在故障发生之后生效, 控制器根据数据平面发送的故障消息和状态做出决策, 为受影响流量计算备份路径. 因此在故障恢复过程中, 控制器与交换机之间的通信、新路由的计算以及设备上转发规则的配置等过程, 会为故障恢复带来较大的开销和时延. 相反, 主动式方法在故障发生前为各种可能的故障制定备份策略, 提前在相关交换机上安装备份规则, 从而避免了恢复过程中控制器的介入. 通常来说主动式恢复方法需要提前占用交换机更多的存储空间.

1.2 相关工作

在 SDN 数据平面的故障恢复研究领域已有大量的研究成果, 主要采用了各种主动式恢复方案进行流量的重定向, 以减少故障发生时产生拥塞的概率. Liu 等人[4] 提出一种具有 k 个并发故障容错能力的链路故障恢复机制 FFC. 一旦发生故障, 受影响流量将通过备份路径进行传输. 然而方案需要链路预留一定容量, 容易导致资源的浪费. Shojaee 等人[5] 提出的 Safeguard 方案为受影响的数据流遍历所有备份路径, 检查链路剩余容量, 通过贪心算法将数据流重定向至剩余资源更充足的备份路径. 这种方法的目的是在多路径传输中尽量避免碎片化造成的资源浪费, 但是也可能导致链路上的带宽资源耗尽, 因为其没有考虑网络的动态性. Zheng 等人[6] 的 Sentinel 方案则不需要保留空闲容量, 方案通过提前安装备份路径并解决 ILP 模型来重路由流量, 同时最小化最大链路利用率. 这些基于数据流的解决方案均存在缺陷, 即在选择路径和分配流量时并没有考虑交换机内存的限制. 交换机内存耗尽会增加表查找和更新时间, 导致大量丢包[7]. 针对交换机容量的限制, Wang 等人[8] 提出一种基于共享环的备份路径方案. 方案通过节点重要性的排序选举出一组核心交换机, 并在这组交换机中定义共享环. 由于共享环被所有备份路径共享, 因此环中备份流表项复用率高. 该方案在保证故障恢复时延的同时, 在备份资源消耗方面具有很好的性能. 但是共享环不能规避恢复后的链路拥塞风险. 而 Wang 等人[9]、Barakabitze 等人[10]、Tian 等人[11] 在流量工程中采用 Segment Routing 的思想, 通过在入口节点为数据包头添加有序的路径信息, 有效地减少了交换机内存的消耗. 由于其技术特点, 所采用的方法引入了报文头开销与更多的延迟. 考虑到网络的动态性, Wang 等人[12] 提出了一种基于反应式和主动式策略协同的链路故障恢复机制, 首先保证故障发生后网络连通性的即时恢复, 再根据当前网络状态信息对备份路径进行调整. 该方案通过反应式和主动式策略的协同工作, 实现了恢复时间和资源利用上的有效平衡.

1.3 基于拥塞及内存感知的链路故障恢复

考虑到数据平面故障恢复的需求, 以及当前研究存在的一些不足, 本文提出一种基于拥塞及内存感知的主动式故障恢复方案, 命名为 CAMA (congestion and memory awareness). CAMA 方案使用较少的备份规则即可实现所有链路的备份路径配置, 完成故障恢复的同时避免潜在的链路拥塞, 从而实现负载均衡.

本文工作的贡献在于:

(1) 提出基于链路的保护方法, 使用 VLAN 标签聚合故障链路流量, 减少交换机流表项条数.

(2) 提出将部分受故障影响的数据流提前导回到原主路径的流表设计方案, 减少链路带宽的消耗.

(3) 提出一种基于拥塞及内存感知的建模方法, 并通过启发式算法在合理时间内求解.

通过实验评估, CAMA 方案在节省备份资源上表现出了较优的性能, 能有效减少交换机内存的消耗, 避免故障恢复后的链路拥塞.

2 问题描述

本文的故障恢复方案在每个 TE 间隔内包括两个步骤: (1) 为每个链路故障计算备份路径, 下发相应流表项和组表项至交换机; (2) 当发生链路故障时交换机自动切换到备份路径. 本文的重点是故障链路备份路径的计算, 以实现负载均衡和资源优化. 本节首先通过一个简单示例展示 CAMA 的整体设计思路和优势 (如图 1 所示).

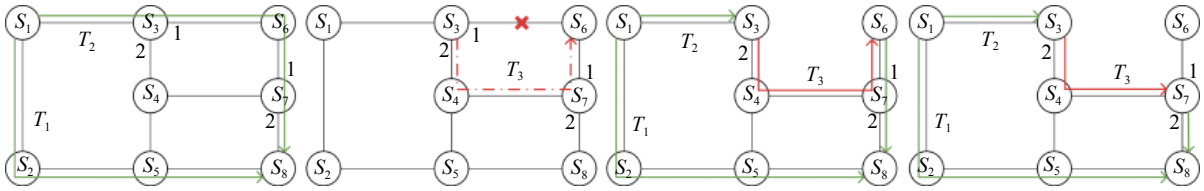

图 1 简单示例

为简单起见, 将两个通信节点之间产生的流量合称为一条数据流. 考虑到负载均衡因素, 该数据流会在这两个通信节点之间的多条路径上传输 (称之为主路径), 每条路径由多段链路组成. CAMA 会为网络中的每一段链路计算备份路径. 备份路径通过快速恢复组表 (OpenFlow fast failover group tables, FF) 进行配置, 并在发生故障时完成检测和即时恢复. 在 SND 交换机中, 一个 FF 组表项包含一系列的动作桶, 每个动作桶关联一个交换机端口, 负责监测端口的状态并执行相应的动作. 如果监测到端口无法使用, 数据包将被快速切换到列表中正常工作的下一个动作桶.

在图 1(a) 的示例拓扑中有 8 个交换机, 每条链路容量均为 10 Gb/s. 考虑从 S_1 到 S_8 带宽需求为 10 Gb/s 的数据流 f, 路径 T_1, T_2 是控制器为 f 计算得到的两条主路径. 当 S_3 和 S_6 之间的链路发生故障时, 路径 T_2 上传输的流量将中断, 设相应的备份路径为 T_3, 如图 1(b) 所示. 表 1 展示了示例中的路径信息. S_1 中通过选择组表 (OpenFlow SELECT group tables, SELECT) 实现流量的权重划分. SELECT 组表包含一系列动作桶, 通过为不同端口设定权重来实现多路径传输, 具体实现办法可以参考 Safeguard[5]. 在示例中每条路径负载为 5 Gb/s, 即权重设定为 (1/2, 1/2).

表 1 图 1 中路径的描述

路径Id	路径类型	标号	路径信息
1	primary	T_1	<S_1, S_2, S_5, S_8>
2	primary	T_2	<S_1, S_3, S_6, S_7, S_8>
3	backup	T_3	<S_3, S_4, S_7, S_6>

根据 SafeGuard 及 Sentinel[5,6] 中的算法, 故障发生后的传输路径如图 1(c). 此时交换机 S_3 上相关的流表和组表如表 2 所示. 当它监测到首选端口状态变为 down 后, 会根据下一个动作桶的配置将数据包转发到备份路径 T_3 上. 然而, 这种基于数据流的备份方法产生的流表项会随数据流的数量增加而剧增. 本文从另一个角度出发, 考虑基于链路的保护方法, 即在故障节点处为受影响的数据包添加故障标签 (VLAN ID), 同时在备份路径的每个节点上增加一条对应的流表项. 通过匹配流表项中指定的 VLAN 字段, 可以将中断的流聚合到同一流表项中. 这样, 所有受影响的数据包在备份路径上传输时, 只需匹配一条流表项即可完成重定向. 因此, 本文提出的基于链路的保护方法能够有效地节省交换机存储空间[13]. 本例中针对交换机 S_3, 它修改后的组表项如表 3. 在备份路径的设计上, 文献 [5,6] 都主张将数据流从故障链路的入口节点尽快重定向到出口节点, 而不是重新规划一条从故障链路的入口节点到终点的新路径. 这是因为将数据流重定向至原工作路径能减少对其他链路上正常流量的影响, 避免产生拥塞[14]. 本文我们仍然沿用这种思路, 并对其进行优化.

表 2 图 1 示例中交换机 S_3 的流表和组表

		匹配域			指令
流表	SrcAddr	DstAddr	Tag	Pri	
	S_1	S_8	—	1	Group 4.1
	组Id	组类型	动作桶		
组表	Gr 4.1	Fast Failover	Bucket1: watch_port=1, output=1 Bucket2: watch_port=2, output=2		

表 3 交换机 S_3 修改后的组表

组Id	组类型	动作桶
Gr 4.1	Fast Failover	Bucket1: watch_port=1, output=1 Bucket2: watch_port=2, Tag←id of link S_3–S_6, Gr 4.2

在本例中, 主路径 T_2 与备份路径 T_3 存在重叠, 会使数据流在 S_6 和 S_7 之间的链路上往返传输, 从而浪费了带宽. 针对这个问题, 本文提出改进算法, 让合适的数据流在备份路径上提前找到出口, 以减少在往返链路上的传输. 我们的做法是让相关数据包在相交节点处匹配到高优先级的流表项, 提前去除故障标签. 去除标签的数据包通过多级流表回环, 继续匹配属于主路

径的工作流表项, 从而在备份路径中提前找到出口并完成后续的传输, 传输路径如图 1(d). 本例中, S_7 是我们要找的出口节点, 它的转发规则如表 4 所示. 表 3 与表 4 中动作指向的组表项 Group 4.2、Group 7.2 类型为 SELECT, 由后续的备份路径选择算法提供.

表 4 图 1 示例中交换机 S_7 的流表和组表

	匹配域				指令
	SrcAddr	DstAddr	Tag	Pri	
流表	—	—	Id of link S_3–S_6	1	Output: 1
	S_1	S_8	—	1	Gr 7.1
	S_1	S_8	Id of link S_3–S_6	2	Pop tag, Gr 7.1

	组Id	组类型	动作桶
组表	Gr 7.1	Fast Failover	Bucket1: watch_port=2, output=2 Bucket2: watch_port=1, Tag←Id of link S_7–S_8, Gr 7.2

综上所述, CAMA 方案在为故障链路计算备份方案时需要考虑两点: (1) 如何在可用路径中选取合适的备份路径并计算权重; (2) 如何为受影响流量在备份路径上找到出口, 以降低链路利用率.

3 系统建模

本节将 SD-WAN 中的单链路故障恢复问题表述为一个混合整数线性规划 (mixed-integer linear programming, MILP) 问题, 所提出的解决方法将在第 4 节详细阐述.

3.1 网络描述

本文将 SDN 网络表示为一个有向图, 记作图 $G=(V,E)$. 其中 $V=\{1,2,\cdots,n\}$ 为网络中 SDN 交换机节点的集合, E 为网络中边的集合, 每条链路 $(i,j)\in E$ 的链路容量为 $c_{i,j}$. 两个通信节点之间的数据流记为 f, 带宽需求为 b_f. 数据流 f 在入口交换机处经过权重划分后在不同的路径上传输. 这些不同路径的集合记为 P_f, 称之为 f 的主路径集合. 我们约定 P_f 中的路径彼此不存在链路重合的情况. 本文主要考虑单链路故障, 此时故障将影响经过该链路的所有数据流. 当链路 e 发生故障时, 假设存在一个候选路径集 Q_e, 其中每条路径 $p\in Q_e$ 都不包含故障链路且彼此不相交, 使得流量能从故障链路的入口节点重定向至出口节点. 由于 P_f 中的路径彼此链路不重合, 因此故障发生时最多影响 f 的一条主路径. 由于经过故障链路的流可能不止一条, 本文将经过该故障链路的所有数据流视为一条聚合流 F_e. 根据后续的备份路径选择算法, 集合 Q_e 中的一条或多条路径将被选中为链路 e 发生故障时的备份路径, 并根据权重对链路 e 上的数据流进行重定向. 本文用到的主要符号如表 5 所示.

表 5 主要符号及意义

符号	描述
$G=(V,E)$	包含节点集 V 和链路集 E 的网络图
P_f	数据流 f 的主路径集合
Q_e	链路 e 的备份路径集合
F_e	链路 e 故障时受影响的数据流集合
F	链路 e 故障时不受影响的数据流集合
b_f	数据流 f 的带宽需求
$c_{i,j}$	链路 $(i,j)\in E$ 的带宽容量
$t_{i,j}$	链路 $(i,j)\in E$ 的已用带宽
Tc_i	节点 $i\in V$ 的TCAM容量
$r_{ij,p}$	示性函数, 链路 $(i,j)\in p$ 时等于1
$s_{i,p}$	示性函数, 节点 $i\in p$ 时等于1
$o_{i,f}$	示性函数, 节点 i 是 f 的出口时等于1
$l_{i,p}$	节点 $i\in p$ 在路径 p 中的位置 (跳数)
λ	链路带宽利用率最大值
$x_{f,p}$	数据流 f 在路径 p 上分配的权重

3.2 变量与参数

在发生故障时需要将受影响的数据流分配到可用备份路径上, 并在备份路径上寻找出口节点. 因此本文定义一个决策变量 $x_{f,p}\in[0,1]$ 和一个示性函数 $o_{i,f}$. $x_{f,p}$ 大于 0 时既表示路径 p 被选中为数据流 f 的传输路径, 同时也表示数据流在该路径上分配的权重. $o_{i,f}$ 等于 1 时表示节点 i 是数据流 f 在备份路径上的出口节点. CAMA 方案的关键就是针对故障链路 e, 为受影响的聚合流 F_e 计算所有的 $x_{F_e,p}$, $p\in Q_e$, 以及对应的节点集合 $\{o_{i,f}\}$, $f\in F_e$.

本文将链路 $(i,j)\in E$ 的已使用带宽记为 $t_{i,j}$, 同时使用示性函数 $r_{ij,p}$ 与 $s_{i,p}$ 分别表示链路 (i,j) 与节点 i 是否属于路径 p.

$$r_{ij,p}=\begin{cases}1, & \text{if } (i,j)\in p\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

$$s_{i,p}=\begin{cases}1, & \text{if } i\in p\\ 0, & \text{otherwise}\end{cases} \tag{2}$$

3.3 约束条件

(1) 链路和节点容量限制: 约束式 (3) 表示链路已使用带宽不能超过设定的阈值, 即参数 λ 与链路容量 $c_{i,j}$ 的乘积. λ 表示网络中链路带宽利用率的最大值. 故

障恢复后链路(i, j)已使用带宽包含两部分: 受故障影响的数据流在包含该链路的备份路径上分配的带宽大小, 与不受故障影响的数据流在包含该链路的主路径上分配的带宽大小. t_e表示链路e发生故障前的已用带宽. 约束式 (4) 用于防止交换机使用的存储空间超过容量上限. 与链路带宽约束类似, 交换机i已使用的存储空间同样包含两部分: 每种故障情况下交换机新增的备份流表项, 以及交换机上已有的流表项.

$$\sum_{p\in Q_e} t_e x_{F_e,p} r_{ij,p} + \sum_{f\in F}\sum_{p\in P_f} b_f x_{f,p} r_{ij,p} \leqslant \lambda c_{i,j},\ \forall (i,j)\in E \tag{3}$$

$$\sum_{e\in E}\sum_{p\in Q_e} s_{i,p}\left(1+\sum_{f\in F_e} o_{i,f}\right)+\sum_{f\in F}\sum_{p\in P_f} s_{i,p} \leqslant T c_i,\ \forall i\in V \tag{4}$$

(2) 数据流限制: 约束式 (5) 用于满足数据流的带宽需求, 同时也说明当路径p未被使用时$x_{F_e,p}=0$, 否则$x_{F_e,p}$必须是一个正数.

$$\sum_{p\in Q_e} x_{F_e,p} = 1\ x_{F_e,p} \geqslant 0, \forall e\in E \tag{5}$$

3.4 目标函数

在满足约束条件的情况下, 目标函数 (6) 追求在完成故障恢复的同时最小化最大链路带宽利用率. 需要注意的是, 在备份路径上提前出口的数据流数量越多, 新增带宽需求越少, 但备份流表项数量会增多.

$$\min\lambda \tag{6}$$

本文的故障恢复方案作为一个 MILP 问题, 已被证明为 NP 难[15], 且在实践运行中需要花费较长时间. 而流量工程要求的时间间隔一般不超过 5 min, 因此本文提出一种启发式恢复算法, 来保证故障恢复问题能够在合理时间内解决.

4 CAMA 方案

本节主要展示所提出的启发式算法. 算法包含 3 个部分: 计算可用路径影响程度并排序; 在已排序路径中选择备份路径; 在备份路径中为数据流寻找出口节点.

4.1 路径的影响程度

本文采用影响程度描述一条路径的性能, 该指标使用一个三元组来表示, 包括链路负载、跳数、交换机负载. 所使用的 3 个参数都与路径传输性能以及网络资源调度相关. CAMA 方案使用路径中的最大链路利用率表示链路负载, 用路径中的最大流表项数量来表示交换机负载.

根据上述的评价参数, CAMA 方案构建一个关于K条候选备份路径的影响程度评价矩阵, 其中$K=|Q_e|$.

$$A=\begin{Bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \vdots & a_{pq} & \vdots \\ a_{K1} & a_{K2} & a_{K3} \end{Bmatrix},\ p=1,2,\cdots,K;\ q=1,2,3 \tag{7}$$

评价矩阵A中, a_{pq}代表第p条候选路径的第q个评价参数的数值. 当$K=1$时, 唯一路径将被直接选中作为备份路径. 由于每个评价参数的单位不同, 本文采用最大最小归一化对参数进行无量纲化处理, 得到标准化评价矩阵Z.

$$Z=\begin{Bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ \vdots & z_{pq} & \vdots \\ z_{K1} & z_{K2} & z_{K3} \end{Bmatrix},\ p=1,2,\cdots,K;\ q=1,2,3 \tag{8}$$

其中, z_{pq}的获取方式如式 (9). 当某列参数$a_{\cdot q}$在所有路径上的值都相等时, 赋予$z_{\cdot q}$相同的数值, 此时路径在此参数上没有区别.

$$\begin{cases} z_{pq}=\begin{cases} \dfrac{a_{pq}-\min\left(a_{\cdot q}\right)}{\max\left(a_{\cdot q}\right)-\min\left(a_{\cdot q}\right)}, & \max\left(a_{\cdot q}\right)\neq\min\left(a_{\cdot q}\right) \\ 0, & \max\left(a_{\cdot q}\right)=\min\left(a_{\cdot q}\right) \end{cases} \\ p=1,2,\cdots,K;\ q=1,2,3 \end{cases} \tag{9}$$

方案定义路径评价指标im_p表示候选路径p的性能, 如式 (10) 所示:

$$im_p=\frac{1}{3}\sum_{q=1}^{3}\omega_q z_{pq},\ p=1,2,\cdots,K \tag{10}$$

其中, ω代表每个参数的权重. 根据上述讨论, 参数分别为链路负载、跳数、交换机负载. 为保证路径的传输性能, 根据各评估参数的重要性以及实际运行中各参数产生的效果, 本文设置$\omega=\{1, 0.8, 0.5\}$.

4.2 备份路径选择

算法 1 展示了单链路故障情况下备份路径的选择过程. 算法将移除故障链路后的网络拓扑、受影响数据流以及可用候选路径作为输入.

算法 1. 备份路径选择算法

输入: 网络拓扑$G'=(V,E\backslash e)$; 聚合流F_e; 可用路径Q_e

输出: 路径分配权重$\{x_{F_e,p}\}$, $p\in Q_e$

1. 计算路径的影响程度$\{im_p\}$, $p\in Q_e$
2. 根据$\{im_p\}$将Q_e升序排序
3. $Cap\leftarrow\{\}$, $res\leftarrow 1$, $sum\leftarrow 0$, $temp\leftarrow 0$
4. **for** $p\in Q_e$ **do**
5. $Cap=Cap\cup cap_p$
6. $sum=sum+cap_p$
7. **for** $cap_{p'}\in Cap$ **do**
8. $\lambda_{p'}\leftarrow$将带宽$t_e\frac{cap_{p'}}{sum}$分配至路径p'得到的最大链路利用率
9. $temp=\max(temp,\lambda_{p'})$
10. **if** $res>temp$ **then**
11. $res=temp$
12. $\{x_{F_e,p}\}=\frac{Cap}{sum}$

算法 1 首先根据影响程度im_p对候选路径进行升序排序, 排序后的候选路径同时代表着路径传输性能的次序. 算法定义一个集合Cap, 用于存储已遍历路径的瓶颈剩余带宽. 依次遍历每条候选路径p, 将该路径的瓶颈链路剩余带宽cap_p (计算方法见式 (11)) 加入集合中Cap, 并在变量sum中累加 (见算法 1 中的第 3–6 行).

$$cap_p=\min_{(i,j)\in p}(c_{i,j}-t_{i,j}) \tag{11}$$

$$\lambda_p=\max_{(i,j)\in p}\left(\frac{t_{i,j}}{c_{i,j}}\right) \tag{12}$$

在算法 1 的内层循环中 (见算法 1 中的第 7–9 行), 根据集合Cap中已遍历路径的剩余带宽的比例$cap_{p'}/sum$分配带宽t_e, 计算分配流量后路径p'的最大链路利用率$\lambda_{p'}$ (计算方法见式 (12)), 循环结束后得到已遍历路径中$\lambda_{p'}$的最大值并保存在变量$temp$中, 用于后续的比较. 在内层循环之后, 当经过流量分配得到的最大链路利用率$temp$小于中间结果res, 说明已计算出使得最大链路利用率更小的权重分配方案, 此时算法 1 更新res并修改路径权重集合$\{x_{F_e,p}\}$, 其中更新的值为集合Cap中的每一项与sum的比值 (见算法 1 中的第 10–12 行).

4.3 数据流出口位置计算

算法 2 展示了备份路径中数据流$f\in F_e$的出口位置计算过程, 其将算法 1 中的权重集合作为输入, 在权重值非 0 的备份路径中为数据流寻找出口位置.

算法 2. 数据流出口位置计算策略

输入: 网络拓扑$G'=(V,E\backslash e)$; 故障链路e, 聚合流F_e;可用路径Q_e
输出: 数据流出口节点集合$\{o_{i,f}\}$, $f\in F_e$

1. **for** $f\in F_e$ **do**
2. $p'\leftarrow$数据流f受影响的主路径
3. $i_1,i_2\leftarrow$链路e的入口节点及出口节点
4. **for** $p\in Q_e$ **do**
5. **if** $x_{F_e,p}=0$ **then**
6. **break**
7. $p=p\backslash\{i_1,i_2\}$
8. **for** $i\in p$ **do**
9. **if** $s_{i,p'}=1$ and $l_{i,p'}>l_{i_2,p'}$ **then**
10. $o_{i,f}=i$
11. $\{o_{i,f}\}=\{o_{i,f}\}\cup o_{i,f}$
12. **break**

对于数据流$f\in F_e$, 由于主路径之间链路不相交, 故只有一条主路径p'受影响. 算法 2 在获取路径p'后遍历备份路径, 寻找主路径与备份路径的相交节点 (见算法 2 中的第 4–12 行). 当节点相交并且该节点在主路径中所处的位置在故障链路之后, 算法 2 将该节点设置为数据流f的出口节点, 即将$o_{i,f}$置 1 (见算法 2 中的第 9–12 行). 由于本文采用有向网络, 为防止出现环路, 算法 2 选取的出口节点在主路径中处于故障链路之后. 算法中$l_{i,p}$表示节点i在路径p中所处的位置 (跳数). 算法 2 最终得到链路发生故障的情况下, 数据流$f\in F_e$的出口位置集合$\{o_{i,f}\}$.

5 实验评估

本文通过仿真实验来评估 CAMA 方案的性能, 包括故障恢复后的链路利用率、备份路径长度、数据流在备份路径上提前找到出口的概率、备份流表项数量等指标.

5.1 实验设置

本文将 CAMA 方案编写成 Python 应用部署在 SDN 控制器 Ryu (版本 4.34) 上, 数据平面采用 Mininet (版本 2.3.0) 实现, 所有的实验都在一台 3.20 GHz CPU 和 8 GB RAM 的机器上运行. 本文考虑两个网络拓扑: Google B4[16](12 个节点, 38 条链路) 与 ATT[17] (25 个节点, 112 条链路). 每个节点是一个 CPqD[18] 交换机实例, 链路容量被设置为 1 Gb/s 并且包含 1 ms 时延, 实验中每个交换机连接一个主机. 本文通过与现有的 SD-WAN 故障恢复方案 SafeGuard和 Sentinel 进行对比, 验证 CAMA 方案的有效性. 当发生链路故障时, Sentinel 致力于最小化最大链路利用率, 然而没有考虑备份路径长度与交换机有限的存储空间; 而 SafeGuard 利用贪心算法选择剩余带宽更多的备份路径, 容易导致潜在的链路拥塞. 实验通过 iperf 工具在数据平面产生 UDP 流量, 并通过调整数据流的数量进行多次比较.

除非特别说明, 每条流的速率都是设定的 50 Mb/s. 实验结果基于多组实验数据的平均值.

5.2 链路利用率

实验评估不同方案在完成单链路故障之后, 恢复结果对整体网络利用率的影响. 实验分别为 B4 以及 ATT 网络拓扑设置了 60 和 200 条数据流, 并在传输过程中使用 link-down 命令模拟链路故障, 计算恢复后活跃链路的利用率. 图 2 展示了故障后不同方案的活跃链路利用率的累积概率分布 (CDF), 可以看到 CAMA 对流量负载的分配比 Sentinel 和 SafeGuard 更加均匀. 在图 2(a) 中, 所有方案都没有产生链路满载, 在 B4 拓扑中 CAMA 的所有链路利用率都小于 60%. 而 SafeGuard 中接近 20% 的链路利用率大于 80%, 在 Sentinel 中这个数值为 28%. 图 2(b) 中数值差距进一步扩大. 由于数据流数量的增加, 3 种方案在 ATT 拓扑中的链路利用率都有不同程度的提高. 在 SafeGuard 中, 链路利用率大于 80% 的链路约有 40%, 在 Sentinel 中数值提升至 57%, 而 CAMA 将这个数值降低至 2%.

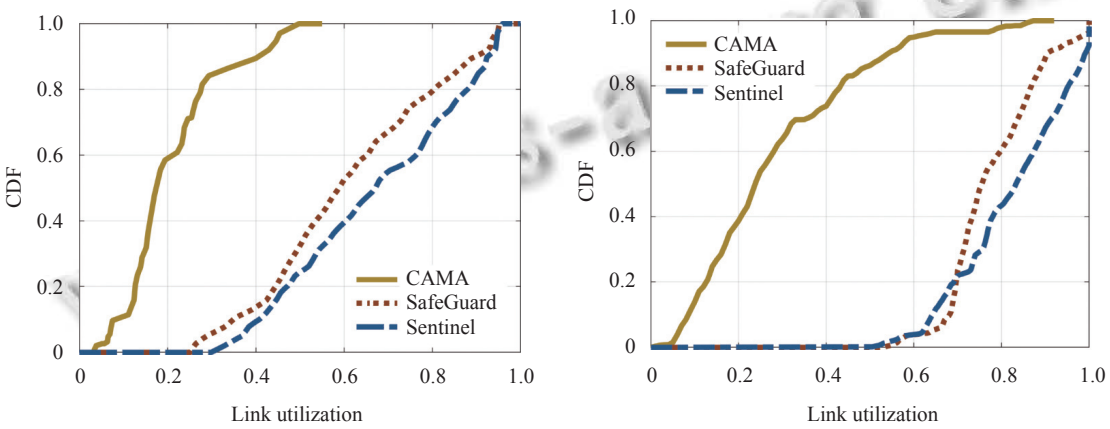

图 2 B4 及 ATT 拓扑的链路利用率累积概率分布

经过分析, CAMA 与 SafeGuard、Sentinel 之间实验数据差异较大的原因有: (1) 在 SafeGuard 中, 主路径没有考虑到环路的存在, 并且在计算过中采用了与备份路径类似的贪心算法, 按照数据流带宽需求从高到低进行分配, 未考虑整体最优的情况; (2) Sentinel 允许主路径之间链路相交, 由于主路径与备份路径均采用基于数据流的计算方法, 且未考虑路径的长度, 部分节点的存储空间已经满载, 导致大量数据包的查表时间增加; (3) 在 CAMA 中, 所选主路径不存在环路, 并且在计算过程中考虑当前路径负载以及路径长度; (4) CAMA 根据路径性能选取备份路径, 并且部分数据流在备份路径上提前找到出口, 有效节省带宽. 这说明 CAMA 具有负载均衡的效果以及较优的实用性.

5.3 路径延伸

备份路径越长, 故障恢复的时延就会越大, 对活跃链路的影响也会越大. 备份路径长度直接影响了恢复后网络的整体性能. 实验通过调整数据流的数量, 评估随机产生链路故障后不同方案备份路径的延伸长度[19], 其中延伸长度为数据流使用的最长备份路径长度与可能的最短路径长度之间的比值. 实验结果中误差棒上下限为不同组实验中结果的最大和最小值, 每组实验在运行 20 次后取平均值. 图 3 展示了两种拓扑中不同数据流数量的平均路径延伸, 可以看出 CAMA 与其他方案的结果具有相反的变化趋势. 在 B4 拓扑中, 如图 3(a) 所示, 当数据流数量为 20 时 CAMA 中备份路径延伸的数值大于另两种方案, 而当数量增加到 40 时 CAMA 在备份路径延伸上已经优于对比方案. 特别是数量增加到 60, CAMA 与 SafeGuard 相比数值减少了 20% (1.15 比 1.38), 而与 Sentinel 相比减少了 42% (1.15 比 1.64). 在 ATT 拓扑中也有类似的趋势, 图 3(b) 显示当数据流数量超过 120 时, CAMA 备份路径延伸的数值优于另两种方案. 当数量为 200 时, CAMA 与 SafeGuard、Sentinel 相比数值分别减少了约 25% 和 38%. CAMA 在数据流数量较少时, 数据流在备份路径上提前出口的概率较小, 使得备份路径较长. 而随着数据流数量增加, 数据流提前出口的概率增大, 备份路径长度减少. 在 SafeGuard 与 Sentinel 中, 随着数据流数量增加, 网络中的链路利用率也会增加, 为了降低恢复

后的链路利用率, 只能选择更多的备份路径, 从而造成了 CAMA 与对比方案的不同变化趋势. 实验说明, 在路径延伸方面, CAMA 在数据流数量较多的网络中表现更优.

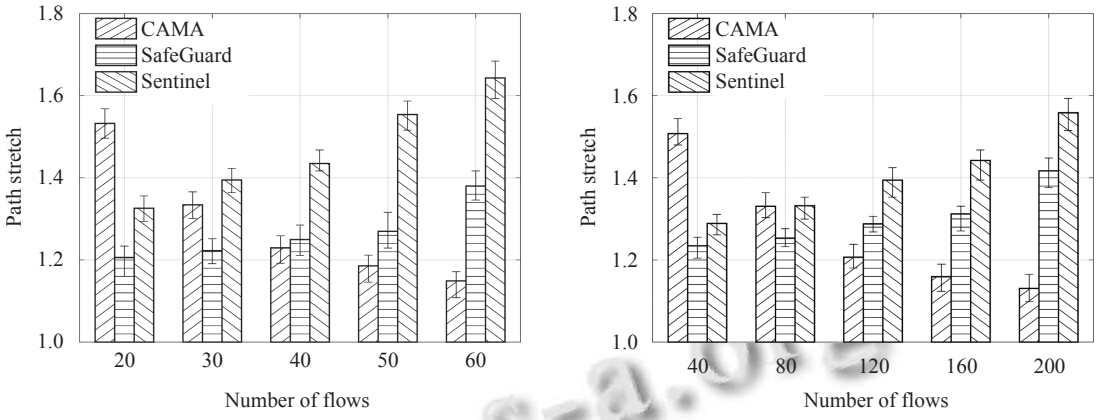

图 3 不同方案的平均路径延伸

5.4 提前出口数据流及备份规则的数量

本节研究 CAMA 方案中对网络整体性能影响较大的因素, 如数据流提前出口的概率, 以及交换机上新增的备份规则数量.

在评估数据流提前出口概率的实验中, 拓扑规模及数据流数量设置同第 5.3 节, 即在 B4 拓扑中实验组别对应的数据流数量分别为 (20, 30, 40, 50, 60), 在 ATT 拓扑中对应的数据流数量分别为 (40, 80, 120, 160, 200). 实验结果取 10 次运行的平均值, 如表 6 所示.

表 6 不同实验组别的数据流提前出口概率 (%)

拓扑	I	II	III	IV	V
B4	16.17	17.22	17.98	18.62	19.38
ATT	11.28	14.4	16.25	18.59	19.16

可以看出在相同拓扑规模下, 随着数据流数量的增加, 数据流提前出口的概率随之增加; 在不同规模拓扑下, 拓扑规模越大数据流提前出口的概率越小. 这是因为当拓扑扩大时, 备份路径与主路径相交的概率降低, 而当数据流数量增加时, 备份路径与主路径相交的概率也会上升.

在评估故障恢复所需备份规则数量的实验中, 规模设置同第 5.2 节, 结果如图 4 所示. 在有效保护每一条链路故障的情况下, 每个交换机只需增加数十条备份规则, 有效节省了交换机有限的存储空间.

6 结束语

本文主要围绕 SDN 数据平面的故障恢复问题展开讨论, 总结已有方法存在的局限性, 并提出了一种基于拥塞及内存感知的主动式故障恢复方案 CAMA. 该方案首先将候选路径进行排序, 然后计算最优的备份方案并为数据流设置出口位置, 以实现恢复后的负载均衡并减少交换机的资源消耗. CAMA 方案在不同拓扑中的负载均衡效果优于对比方案, 能有效节省带宽; 在路径延伸上, CAMA 方案更适用于数据流数量较多的网络, 在少量数据流的网络中没有优势; CAMA 在节省备份资源上表现出了较优的性能, 仅需增加数十条备份转发规则即可覆盖所有单链路故障情况. 综上所述, 本文提出的 CAMA 方案具有一定的实用性及适用性.

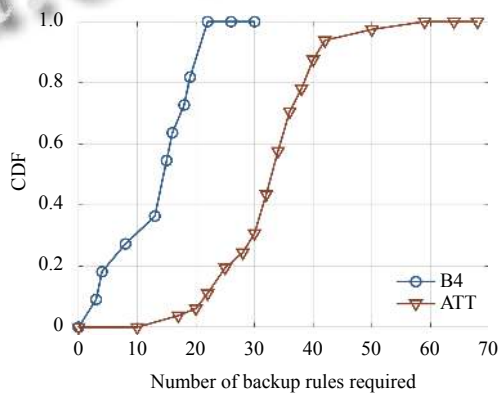

图 4 交换机备份规则数量累积概率分布

参考文献

1 Ali J, Lee GM, Roh BH, *et al*. Software-defined networking

approaches for link failure recovery: A survey. Sustainability, 2020, 12(10): 4255. [doi: 10.3390/su12104255]

2 Farhady H, Lee H, Nakao A. Software-defined networking: A survey. Computer Networks, 2015, 81: 79–95. [doi: 10.1016/j.comnet.2015.02.014]

3 Fonseca PC, Mota ES. A survey on fault management in software-defined networks. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2284–2321. [doi: 10.1109/COMST.2017.2719862]

4 Liu HH, Kandula S, Mahajan R, *et al*. Traffic engineering with forward fault correction. Proceedings of the 2014 ACM Conference on SIGCOMM. Chicago: ACM, 2014. 527–538. [doi: 10.1145/2619239.2626314]

5 Shojaee M, Neves M, Haque I. SafeGuard: Congestion and memory-aware failure recovery in SD-WAN. Proceedings of the 16th International Conference on Network and Service Management. Izmir: IEEE, 2020. 1–7. [doi: 10.23919/CNSM50824.2020.9269119]

6 Zheng JQ, Xu H, Zhu XJ, *et al*. Sentinel: Failure recovery in centralized traffic engineering. IEEE/ACM Transactions on Networking, 2019, 27(5): 1859–1872. [doi: 10.1109/TNET.2019.2931473]

7 Isyaku B, Mohd Zahid MS, Bte Kamat M, *et al*. Software defined networking flow table management of OpenFlow switches performance and security challenges: A survey. Future Internet, 2020, 12(9): 147. [doi: 10.3390/fi12090147]

8 Wang Y, Feng SX, Guo HT, *et al*. A single-link failure recovery approach based on resource sharing and performance prediction in SDN. IEEE Access, 2019, 7: 174750–174763. [doi: 10.1109/ACCESS.2019.2957141]

9 Wang SS, Xu HL, Huang LS, *et al*. Fast recovery for single link failure with segment routing in SDNs. Proceedings of the 21st IEEE International Conference on High Performance Computing and Communications, the 17th IEEE International Conference on Smart City and the 5th IEEE International Conference on Data Science and Systems. Zhangjiajie: IEEE, 2019. 2013–2018. [doi: 10.1109/HPCC/SmartCity/DSS.2019.00278]

10 Barakabitze AA, Sun LF, Mkwawa IH, *et al*. Multipath protections and dynamic link recovery in softwarized 5G networks using segment routing. Proceedings of the 2019 IEEE Globecom Workshops. Waikoloa: IEEE, 2019. 1–6.10. [doi: 10.1109/GCWkshps45667.2019.9024556]

11 Tian Y, Wang ZL, Yin X, *et al*. Traffic engineering with segment routing considering probabilistic failures. Proceedings of the 17th International Conference on Network and Service Management. Izmir: IEEE, 2021. 21–27. [doi: 10.23/919CNSM52442.2021.9615559]

12 Wang LK, Yao L, Xu ZC, *et al*. CFR: A cooperative link failure recovery scheme in software-defined networks. International Journal of Communication Systems, 2018, 31(10): e3560. [doi: 10.1002/dac.3560]

13 Mohan PM, Truong-Huu T, Gurusamy M. Fault tolerance in TCAM-limited software defined networks. Computer Networks, 2017, 116: 47–62. [doi: 10.1016/j.comnet.2017.02.009]

14 Thorat P, Challa R, Raza SM, *et al*. Proactive failure recovery scheme for data traffic in software defined networks. Proceedings of the 2016 IEEE NetSoft Conference and Workshops. Seoul: IEEE, 2016. 219–225. [doi: 10.1109/NETSOFT.2016.7502416]

15 Garey MR, Johnson DS. Computers and intractability: A guide to the theory of NP-completeness. San Francisco: W. H. Freeman & Co., 1979.

16 Jain S, Kumar A, Mandal S, *et al*. B4: Experience with a globally-deployed software defined WAN. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3–14. [doi: 10.1145/2534169.2486019]

17 Knight S, Nguyen HX, Falkner N, *et al*. The Internet topology zoo. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765–1775. [doi: 10.1109/JSAC.2011.111002]

18 Fernandes EL, Rojas E, Alvarez-Horcajo J, *et al*. The road to BOFUSS: The basic OpenFlow userspace software switch. Journal of Network and Computer Applications, 2020, 165: 102685. [doi: 10.1016/j.jnca.2020.102685]

19 Foerster KT, Pignolet YA, Schmid S, *et al*. CASA: Congestion and stretch aware static fast rerouting. Proceedings of the 2019 IEEE Conference on Computer Communications. Paris: IEEE, 2019. 469–477. [doi: 10.1109/INFOCOM.2019.8737438]

(校对责编: 牛欣悦)