计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160055
54(5):952-960,2017

基于拜占庭容错提高 SDN 控制层可靠性的研究

李军飞　胡宇翔　邬江兴
(国家数字交换系统工程技术研究中心　郑州　450002)
(lijunfei90@qq.com)

Research on Improving the Control Plane's Reliability in SDN Based on Byzantine Fault-Tolerance

Li Junfei, Hu Yuxiang, and Wu Jiangxing
(*National Digital Switching System Engineering & Technological R&D Center*, *Zhengzhou* 450002)

Abstract　Software defined network (SDN) proposes the architecture of separating the control logic and forwarding devices in networks, which brings the open API for freely programing and makes the network management more fine. However, while the centralized control of SDN brings innovation and convenience for network applications, it also brings other problems, for example the reliability problem and the scalability problem simultaneously. For the problem of control plane's reliability in SDN, the method that voting deals with the same OpenFlow messages by combing multiple controllers to a quorum view is proposed to tolerate Byzantine faults, which is different from the current OpenFlow protocol. Firstly, we concretely explain the network structure, workflow and exception handling in the application of Byzantine fault-tolerance algorithm with the feature of SDN, and establish the analytical model of multi-controller's deployment. Secondly, we design a heuristic algorithm to solve the problem of multi-controller's deployment. Finally, to verify the fault tolerance method and deploy algorithms by simulation, experimental results show that this method can effectively deal with controllers' faults, improving the reliability of the control layer, but it will sacrifice the system's performance at some level. Meanwhile, the deployment algorithm can effectively reduce the transmission delay of processing OpenFlow request.

Key words　software defined network (SDN); OpenFlow; Byzantine fault tolerance; reliability; controllers' deployment

摘　要　软件定义网络(software defined network, SDN)提出了控制与转发分离的设计结构,实现了开放的可编程网络接口,为网络提供了更细粒度的管理.然而,SDN 在为网络应用带来创新与便利的同时,也面临着一些新的问题.针对 SDN 网络中控制层的可靠性问题,提出了一种容忍拜占庭错误的方法.首先,结合 SDN 网络的特性,具体阐述了在应用拜占庭容错算法时的网络结构、工作流程和异常处理等,并对其中的多控制器位置部署问题建立分析模型;然后,针对该多控制器部署问题,设计了启发式求解算法;最后,通过仿真实验对该容错方法和部署算法进行验证.实验结果表明:该容错方法能够有效处理控制器中的错误,提高控制层的可靠性,但对系统的性能会造成一定程度的影响.同时,该部署算法能够有效降低处理 OpenFlow 请求的传输延迟.

关键词　软件定义网络;OpenFlow;拜占庭容错;可靠性;控制器部署

中图法分类号　TP393

收稿日期:2016-01-26;**修回日期**:2016-08-17

基金项目:国家自然科学基金项目(61521003)

This work was supported by the National Natural Science Foundation of China (61521003).

随着网络应用的快速发展,传统的网络交换设备承载着越来越多的控制逻辑,已难以适应虚拟化、云计算、大数据及相关业务发展对数据高速传输、资源灵活配置、协议快速部署的需求. 软件定义网络(software defined network, SDN)提出了控制与转发分离的设计结构,实现了开放的可编程网络接口,为网络提供了更细粒度的管理,引起了学术界和产业界的广泛研究. 然而,SDN 在为网络应用带来创新与便利的同时,也面临着一些新的问题,如控制器的安全性[1]、交换机的转发效率[2]以及流表的一致性[3]等,制约了其在应用网络中的广泛部署.

例如,相对于传统的网络,SDN 采用集中式控制,使得网络的管理更为简捷和高效,但同时也引入了新的安全威胁. 其中,SDN 控制器大多是部署在计算机上,其智能化程度更高,但往往也具有更多的漏洞,降低了针对控制器的攻击难度. 另一方面,SDN 控制器负责整个网络的管理和维护,其一旦出现错误,将对整个网络的正常运行造成影响,具有更大的危害性.

针对 SDN 控制层存在的上述可靠性问题,OpenFlow v1.2[4]协议提出了多控制器的方法,多个控制器之间构成主备冗余的关系. 在主控制器由于发生错误而停止工作时,选择其中的一个从控制器转变为主控制器. 然而,在实际应用中,控制器更多的是发生拜占庭错误(Byzantine fault),而不仅是造成宕机的良性错误(Benign fault). 拜占庭错误具有一定的隐蔽性,它是在控制器对外表现仍能工作的情况下,对交换机的请求输出任意错误响应,或按照攻击者的意图输出具有破坏性的响应. 因此,目前 SDN 网络中的主备切换机制还难以发现和处理拜占庭错误,给控制层的可靠性造成了严重的威胁.

借鉴已有系统处理拜占庭错误的方法,如冗余文件系统[5]、分布式计算系统[6]等,本文尝试着将拜占庭容错算法(Byzantine fault tolerance, BFT)引入到 SDN 网络中,希望能够引起相关研究者的兴趣. BFT 算法需要多个控制器之间协同工作,对输出给交换机的响应进行表决,以检测出部分控制器出现的异常并予以及时处理,从而达到增强控制层可靠性的效果.

1 相关工作

SDN 控制层是 SDN 网络中的核心部分,负责数据层的管理和维护,但由于其采用集中式控制,在出现错误或遭受攻击时,对网络造成的危害更大. 例如,Diego 等人[1]分析总结了 SDN 网络中的 7 类安全问题,其中 5 类存在于控制层面,且具有更强的破坏性. 因此,控制层可靠性的研究是目前 SDN 领域中的热点,研究者从不同的角度来分析和解决集中式控制带来的安全缺陷.

系统结构上,Casado 等人[7]提出了 FortNOX 架构,它是基于 OpenFlow 控制器 NOX 的软件扩展,提供了身份认证和安全约束机制. FortNOX 使得 NOX 控制器能够实时地检查流表项中的冲突,及时发现具有破坏意图的上层恶意应用. 然而,该结构仅是针对单个控制器内的安全问题,并不能解决控制器节点的失效问题. 同时,Shin 等人[8]提出了基于模块化组合的 FRESCO 架构,通过调用脚本 API 来监测代码运行或发现安全威胁. 因此,FRESCO 的防护效果取决于模块化库的功能强弱,同样也难以应对节点失效的安全问题.

具体应用上,Hu 等人[9]研究了多控制器部署位置的问题,提高了 SDN 网络在单控制器节点失效情况下的可恢复力. Jagadeesan 等人[10]提出了网络分块控制的思想,使得单个控制器在被攻破的情况下,整个网络依然有安全保证. Hu 等人[11]在 SDN 网络中引入了防火墙的技术,结合 SDN 的特性,通过精确检测和规则过滤,提高了网络的抗攻击能力.

另外,Li 等人[12]提出了一种新颖的控制方法,多个冗余的控制器之间相互协作,打破传统的主备切换的方式,使得控制层具有容忍拜占庭错误的能力. 然而,文献[12]重点研究了交换机与控制器之间多对多的对应关系,仅简单介绍了该算法的容错原理,没有结合 SDN 网络的具体场景对其工作机制和实现方法做进一步的研究. 而且,该研究没有给出 BFT 算法对 SDN 网络在性能方面上造成的影响.

2 基于拜占庭容错的 SDN 架构

BFT 算法是冗余文件系统中一种常用的容错算法[5,13],通过冗余体之间的通信交互和投票表决,使得即使在一些冗余体出现错误或遭受攻击的情况下,系统依然可以对外提供正确的服务. 典型的 BFT 算法具有容忍 $\lfloor (n-1)/3 \rfloor$ 个错误的能力,其中 n 表示系统中冗余体的总数. 然而,BFT 算法由于会带来额外的通信开销和资源消耗,往往也会造成一定的性能损失.

2.1 工作流程

与传统 SDN 控制层处理 OF(OpenFlow)请求的方式不同,基于 BFT 算法的 SDN 网络需要多个控制器之间协同处理,对输出响应进行表决,以防止异常控制器对网络造成破坏. 该容错方法不仅能够应对造成控制器宕机的良性错误,而且也能够抵抗由于外部攻击或自身错误导致的控制器输出异常的拜占庭错误. 同时,基于 BFT 算法的 SDN 网络需要多个控制器相互协作,彼此之间相互检测对方是否发生错误,但不需要复杂的差错检验机制来验证自身是否出现错误.

2.1.1 算法流程

对于要容忍 f 个控制器错误的 SDN 网络,典型的 BFT 算法通常需要 $3f+1$ 个控制器,构成处理 OF 请求的冗余系统. 其中,$3f+1$ 个控制器构成的集合称为群视图,记为 v. BFT 算法的流程如图 1 所示:

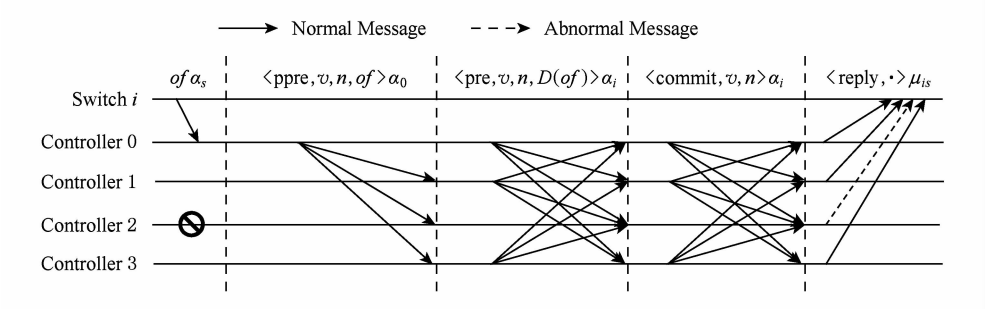

Fig. 1 The procedure of BFT in SDN

图 1 SDN 网络中 BFT 算法的流程

图 1 中,of 表示交换机发出的 OF 请求,ppre,pre,commit,reply 表示 BFT 算法运行的状态,v 表示控制器以自己的视角所看到的群视图,n 是对当前 of 消息的编号,$D(of)$ 表示 OF 消息的摘要. BFT 算法大致分为 5 个步骤:

1) 主控制器接收到交换机发送的 OF 请求,验证其正确性. 若正确,则给请求分配序列号 n,系统进入 pre-prepare 阶段,并对外广播该请求.

2) 各控制器接收到 ppre 消息,验证其相对于自身的正确性. 若正确,控制器进入 prepare 阶段,对外广播确认接收到请求的 prepare 消息,其中,$D(of)$表示 OF 请求的摘要.

3) 各控制器接受到 prepare 消息,验证其相对于自身的正确性,并检查其是否在当前的群视图. 若满足条件,且接收到 $2f$ 个来自不同控制器的 prepare 消息,则进入 commit 阶段,对外广播 commit 消息.

4) 各控制器接受到 commit 消息,验证其正确性,并检查其是否在当前的群视图. 若满足条件,且接收到 $2f$ 个来自不同控制器的 commit 消息,则处理该 OF 请求,并将响应 reply 消息返回给交换机.

5) 交换机接收到 $f+1$ 个来自不同控制器的相同的且验证通过的 reply 消息时,则认定该响应为请求处理的结果.

上述流程主要是为了在多个控制器共同处理来自不同交换机的多个 OF 请求时,保证其处理请求顺序的一致性以及返回响应的一致性. 其中,对于步骤 5,由于其与目前交换机的工作方式不同,我们可以通过在交换机里加入该部分的控制逻辑,或在交换机与控制器之间加入相应的代理器(类似于 FlowVisor)来实现对多个输入的表决判断. 另外,控制器之间的消息交互,可以通过东西向的接口来实现. 对于算法流程中的消息验证,如 a_i,μ_{is} 等是为了防止来自破坏者的虚假消息,以及消息在网络传输过程中被篡改,我们可以采用公钥密码对消息的来源和完整性进行验证.

2.1.2 容错机制

BFT 算法的主要作用就是能够及时发现群视图中控制器成员的异常,对其进行主动恢复或切换当前的群视图,以防止其对系统的正常工作造成影响. 对于 SDN 网络中的从控制器异常,如图 1 中控制器 Controller 2,当其为良性错误时,则停止工作,不再与其余控制器进行交互,使得该错误能够被主控制器及时发现;当其为拜占庭错误时,则行为异常,甚至输出具有破坏目的的响应. 由于其输出的结果与大多数控制器的结果不一致,交换机(或代理器)在择多判决多个控制器的输出结果时,能够及时发现该控制器的异常.

对于主控制器的异常,如图 1 中控制器 c_0,当其为良性错误时,则与所管理的交换机失去连接,这些交换机尝试连接至某一从控制器,进而发现主控制器的错误;当其为拜占庭错误时,造成其余控制器在交互时关于视图 v 无法达成一致,或交换机得不到

$f+1$ 个相同的输出结果.此时,由于无法完成正常的 BFT 流程,可以判断主控制器发生了异常.

2.1.3 群视图切换

当群视图中异常控制器的数量达到 $h(h\leqslant f)$,或主控制器出现错误时,为保证控制层能够正确处理交换机的 OF 请求,需要进行群视图切换,选择一些功能正常的控制器作为新的视图.如图 2 所示,原群视图 $v=\{c_0,c_1,c_2,c_3\}$,在主控制器 c_0 出现异常时,视图中其余的成员发起视图切换的请求,将其对外广播.控制器接收到视图切换请求的消息,选取其中编号较小的控制器作为主控制器,并向其发送视图确认的消息.某一控制器在接收到 $3f$ 个来自不同控制器的视图确认的消息后,则认定为主控制器,产生新的群视图 $v'=\{c_0,c_1,c_2,c_3,c_4\}$,并向 v' 中的其余控制器广播新群视图的消息.

群视图的切换有助于 SDN 网络在应用 BFT 算法能够容忍 f 个错误的基础上,进一步增强网络的可靠性和安全性.但同时也意味着,网络中要有更多的控制器,以在视图切换时有正常的控制器可供使用.进一步地,可以在该网络中引入主动恢复机制[14],使得即使在有限数量的控制器下,控制层依然可以在不同的时刻进行群视图切换.

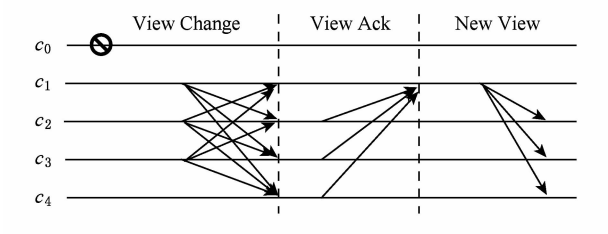

Fig. 2 The procedure of change quorum view

图 2 群视图的切换过程

2.2 网络结构

在运行 BFT 算法时,交换机可以以任一控制器为主控制器,且可以在其之间动态切换,因此,交换机与控制器之间要实现多对多的连通.另一方面,控制器之间为了实现处理请求和网络视图的一致性,需要相互通信,因此,控制器之间是要相互连通的.如图 3 所示,对于带外通信(Out-Band)的 SDN 网络,要实现 BFT 算法,不仅控制器之间需要互相连通,而且交换机与其群视图(quorum view)内的控制器也要全部实现互连;对于带内通信(In-Band)的 SDN 网络,其 OF 控制流和控制器之间的数据流可以通过数据平面中的路径传输,大大降低了网络互连的复杂度,但同时也增加了通信的延迟和开销.

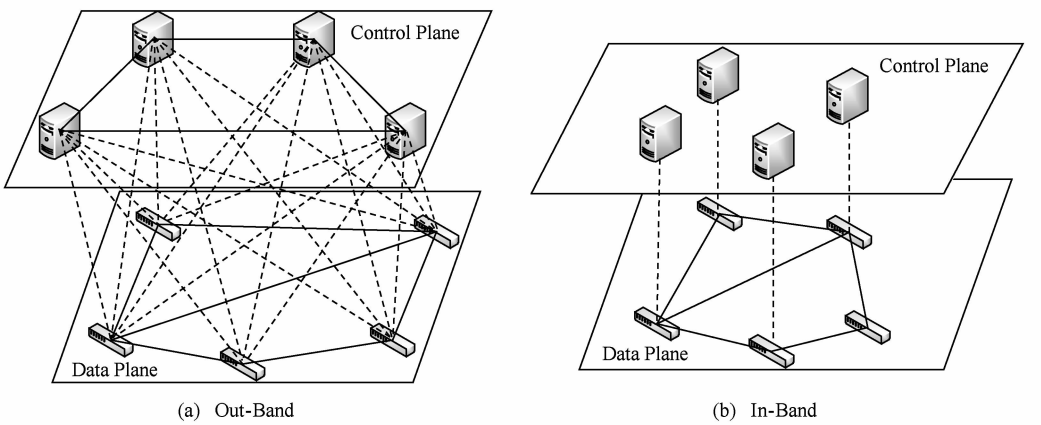

Fig. 3 The SDN network based on BFT

图 3 基于 BFT 算法的 SDN 网络结构

另外,在具体设计基于 BFT 算法的 SDN 网络结构时,有 3 点值得注意:1)控制器之间的网络拓扑具有较强的连通性,即单个控制器故障不会影响其余控制器间的通信;2)交换机与控制器之间具有较近的拓扑距离,以降低控制流的传输时延;3)控制器之间的网络连接具有较高的可靠性,保证 BFT 算法的通信效率,以提高服务请求的处理速度.

2.3 CDB 问题

分析上述基于 BFT 算法的 SDN 网络,为了提高其控制层的可靠性,带来了大量的控制器冗余及通信开销.所以,在部署该网络时,我们需要考虑交换机与控制器之间的连接关系,以尽可能地提高网络的性能,减小该方法带来的负面影响.例如,文献[12]针对带外通信的 SDN 网络,研究了交换机与控制器之间多对多的对应关系,以满足运行 BFT 算法的通信延迟、控制器负载等约束条件.对于带内通信的 SDN 网络,研究者更多地关注控制器位置的部署,以提高网络的弹性[15]和可靠性[16].同样,其对

BFT 算法的运行效率也具有重要的影响,因此,本文我们将重点研究 BFT 算法下多控制器的位置部署(controllers' deployment with BFT, CDB)问题.

为了便于表述和计算,结合图 1(b),我们首先针对基于 BFT 算法的 SDN 网络,给出 4 点模型假设:

1) 控制层具有 n 个控制器,设定其能容忍 f 个错误.为了使得在发生错误时,BFT 算法能够切换群视图,需要 $n>3f+1$.

2) 对于交换机 s_i,为了减低 OF 请求的传输延迟,选择距离其最近的控制器 j 作为主控制器 p_i,即 $\forall c_h \in C, \delta_{ij} \leqslant \delta_{ih}$.

3) 对于主控制器 p_i,为了降低控制器之间的通信延迟,选择距离其最近的 $2f$ 个控制器,构成基于 BFT 算法交换机 i 的控制器集合 ϕ_i,即 $\forall c_j \in \phi_i$, $c_h \notin \phi_i \wedge c_h \in C, \gamma_{ij} \leqslant \gamma_{ih}$.

4) 为了降低单个交换机失效对控制层造成的影响,假定一个交换机上最多部署一个控制器,即 $\forall c_i, c_j \in C$ 且 $i \neq j, \psi(i) \neq \psi(j)$.

CDB 问题可以针对不同的优化目标,如传输时延、通信开销、资源利用率等,分别对应着不同的目标函数.本文将以传输时延为例,研究 SDN 网络中控制器的优化部署,以期提高 BFT 算法下处理 OF 请求的速度.

3 CDB 问题的求解

3.1 计算过程

对于一个具有 m 个交换机的 SDN 网络,其数据层的网络拓扑可以用邻接矩阵表示为 $\boldsymbol{G}=(a_{ij})_{m\times m}$.其中,$a_{ij}$ 是边$\langle s_i, s_j\rangle$的权重,表示该路径上的传输延迟,当 s_i 与 s_j 无直接连接时,$a_{ij}=+\infty$.进而,可以使用 Dijkstra 算法求得任意 2 个交换机间的最短距离,即 $\boldsymbol{G}'=(d_{ij})_{m\times m}$.对于 CDB 问题,在给定常量矩阵 $\boldsymbol{G}'$和变量函数$\psi(i)$的情况下,根据 2.3 节中的假设,可以计算出表 1 中的中间量 p_i, ϕ_i, δ_{ij}, γ_{ij}.例如:

$$\begin{cases}\delta_{ij}=d_{ih}, \text{ s.t. } h=\psi(j),\\ \gamma_{ij}=d_{pq}, \text{ s.t. } p=\psi(i), q=\psi(j).\end{cases}$$

控制器在运行 BFT 算法时,处理交换机 s_i 的一次 OF 请求需要经历 5 个阶段,如图 2 所示.对于 request 阶段,其路径传输延迟为

$$T_{\text{req}}=\delta_{ij}/v, j=\psi(p_i), \tag{1}$$

Table1 Symbols in CDB Problem

表 1 CDB 问题中的符号

Symbol	Meaning
$\boldsymbol{G}$	topology of data plane
C	controller set
s_i	switch i
α_i	the amount of OF sent by switch i
c_i	controller i
p_i	master controller of switch i
ϕ_i	controller set of switch i based on BFT
δ_{ij}	the shortest distance between switch i and controller j
γ_{ij}	the shortest distance between controller i and controller j
$\psi(i)$	position that deploys controller i

其中,v 表示信号在链路中的传输速度.对于 pre-prepare 阶段,主控制器需要与群视图内的其余控制器通信,我们选择其中延迟最大的路径做为该阶段的传输延迟,即:

$$T_{\text{ppre}}=\max(\gamma_{hj})/v,\quad j=\psi(p_i), h\in\phi_i. \tag{2}$$

对于 prepare 阶段和 commit 阶段,群视图内的任意 1 个控制器均需要通信,同理,该阶段的传输延迟为

$$T_{\text{pre}}=T_{\text{com}}=\max(\gamma_{rt})/v,\ r,t\in\phi_i. \tag{3}$$

对于返回响应的 reply 阶段,其传输延迟为

$$T_{\text{rep}}=\max(\delta_{iq})/v,\ q\in\phi_i. \tag{4}$$

由式(1)~(4)求和可得,处理交换机 s_i 的一次 OF 请求的总路径延迟为

$$\begin{cases}T_i=(\delta_{ij}+\max(\gamma_{hj})+2\max(\gamma_{rt})+\max(\delta_{iq}))/v,\\ \text{s.t. } j=\psi(p_i); h,r,t,q\in\phi_i.\end{cases}$$

考虑到各个交换机产生 OF 请求的数量不同,我们采用该 SDN 网络处理 OF 请求的平均传输延迟时间 T 作为度量指标,表示为

$$T=\frac{1}{v\sum_{i=1}^{m}\alpha_i}\sum_{i=1}^{m}\alpha_i T_i,$$

即:

$$\begin{cases}T=\Big(\sum_{i=1}^{m}\alpha_i(\delta_{ij}+\max(\gamma_{hj})+2\max(\gamma_{rt})+\\ \quad\max(\delta_{iq}))\Big)\Big/\Big(v\sum_{i=1}^{m}\alpha_i\Big),\\ \text{s.t. } i\in[1,m]; j=\psi(p_i); h,r,t,q\in\phi_i.\end{cases} \tag{5}$$

因此,该 CDB 问题可以形式化描述为:在给定数据层网络拓扑图 $\boldsymbol{G}$ 的情况下,求控制器的部署位置

$\psi(i)$,使得式(5)中的平均传输延迟 T 取得最小值,以提高 SDN 网络中运行 BFT 算法的效率.

3.2 求解算法

对于式(5),要求得 T 的最小值,需要在大小为 C_m^n 的解空间内进行搜索.然而,随着交换机数量 m 和控制器数量 n 的增长,该问题的解空间呈指数级爆炸式增长.因此,该 CDB 问题是一个典型的组合优化问题,具有 NP 问题的计算复杂度.本章我们提出一种基于解码法的遗传算法(decode genetic algorithm, DGA),以期在合理的时间内求解出该 CDB 问题的近似最优解.

算法 1. DGA 算法.

输入:数据层网络拓扑 $\boldsymbol{G}$、控制器数量 n、BFT 容错上限 f;

输出:较好适应度子代的基因序列 x.

① $\boldsymbol{x}=[y_1,y_2,\cdots,y_m]$,$P(t)=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_p\}$;
② 初始化 $P(0)$; /* 初始化种群 */
③ for $t=0$ to K do
④ for $i=0$ to p do
⑤ $F(i)=T(G,n,f,\boldsymbol{x}_i)$; /* 评估个体适应度 */
⑥ end for
⑦ for $i=0$ to p do /* 种群选择 */
⑧ $P^1\{\boldsymbol{x}_i\}\xleftarrow[F(i)]{\text{Select}}P(t)$;
⑨ end for
⑩ for $i=0$ to $p/2$ do /* 种群交叉 */
⑪ $P^2\{\boldsymbol{x}_{2i},\boldsymbol{x}_{2i+1}\}\xleftarrow{\text{Crossover}}P^1(t)$;
⑫ end for
⑬ for $i=0$ to p do /* 种群变异 */
⑭ $P^3\{\boldsymbol{x}_i\}\xleftarrow{\text{Mutation}}P^2(t)$;
⑮ end for
⑯ for $i=0$ to p do /* 异常个体再解码 */
⑰ for each h in m
⑱ $D(h)=\sum_{j=1}^{m}d_{hj}$;
⑲ end for
⑳ if $|\boldsymbol{x}_i|\neq n$
㉑ $P(t+1)\xleftarrow{D}P^3\{\boldsymbol{x}_i\}$;
㉒ end if
㉓ end for
㉔ end for
㉕ $\boldsymbol{x}_i\leftarrow P(K)$ s.t. $F(\boldsymbol{x})=\max(F(\boldsymbol{x}_1),F(\boldsymbol{x}_2),\cdots,F(\boldsymbol{x}_p))$;

在该 DGA 算法中,染色体 $\boldsymbol{x}$ 表示为($y_1,y_2,\cdots,y_m$),与式(5)中控制器的部署位置 $\psi(i)$ 相对应.其中,若 $y_i=1$,表示交换机 i 处部署有控制器,否则,$y_i=0$.因此,染色体 $\boldsymbol{x}$ 是一个长度为 m 的 01 序列.$P(t)$为种群空间,包含有 p 个个体,个体的适应度 F 采用 OF 请求的平均传输延迟 T 来度量(行⑤).然后,根据适应度 F,对种群中的个体按照一定的概率,依次进行选择、交叉和变异(行⑦~⑮),以期获得适应度更好的子代.

对于交叉变异之后的子代种群 P^3,其中的部分个体可能会出现异常,使得 $\boldsymbol{x}$ 中 $y_i=1$ 的数量不等于 n(行⑳),即其所对应控制器的总量不等于 n.对于子代中异常的个体,采用解码法对其进行纠正,删除或增加 $\boldsymbol{x}$ 中 $y_i=1$ 的项(行㉑),使其总量为 n.在对 $\boldsymbol{x}$ 进行修正时,我们以贪心函数 D 作为衡量指标(行⑰~⑲),即该交换节点与其余节点距离的总和,修改其中对应 $D(i)$较小的 $y_i=1$ 项为 $y_i=0$,或将对应 $D(i)$较大的 $y_i=0$ 项修改为 $y_i=1$.该 DGA 算法依次对种群进行迭代进化,直至循环 T 次,最后选择末代种群 $P(T)$中适应度最好的个体作为输出(行㉕).

4 实验验证与分析

4.1 实验环境

4.1.1 仿真实验

为了验证上述方法的有效性,我们采用 C++ 语言实现了一个仿真器,来模拟 SDN 网络中多个控制器之间运行 BFT 算法.该仿真器基于 Internet2 OS3E 网络拓扑,OS3E 是一个遍布美国大部分地区的用于先进科学研究的 SDN 网络[17].如图 4 所示,OS3E 具有 34 个节点和 42 条链路,每个节点表示 1 个独立的大学或组织.对于该网络中的链路距离,我们采用文献[15]中的测试数据,如图 4 中节点 A(Sunnyvale)和 B(Los Angeles)间的链路距离为 503 182 m.

该仿真器中部署 $n(n\leqslant 10)$个控制器,控制器之间可以构成不同规模的群视图,以运行不同容错上限($f=1,2,3$)的 BFT 算法.对于每个控制器,设定其以概率 p 发生错误,其中拜占庭错误的比例为 b,即拜占庭错误发生的概率为 $b\times p$.对于交换机,在采用传统的主备切换方法控制时,我们认为控制层

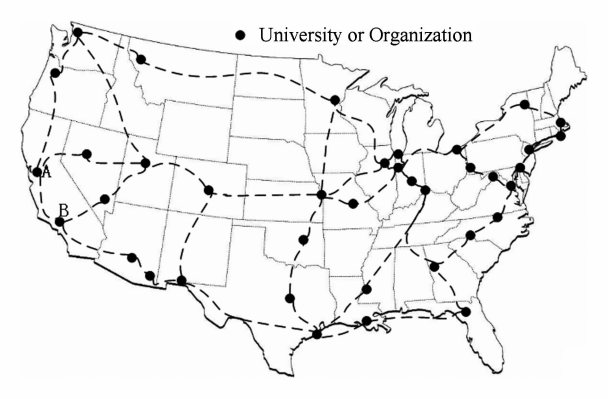

Fig. 4 The topology of OS3E network

图 4 OS3E 的网络拓扑

在出现以下 2 种情况时该交换机发生异常:1)正在工作的主控制器出现拜占庭错误,由于主备切换机制无法检测拜占庭错误而发生异常;2)出现良性错误的控制器的数目达到 n 个,由于控制层无法进行有效的切换而发生异常.在采用 BFT 算法控制时,当交换机群视图内出现错误的控制器达到 $f+1$ 个,则认定该交换机出现异常.

4.1.2 原型测试系统

另外,为了测试该 BFT 算法对系统性能的影响,我们基于 OpenDayLight[18] 开发了简单的原型系统,在其东西向接口上实现控制器之间的协作.负责控制器之间相互协作的 BFT 协议,是基于 BFT-SMART[19] 开源代码来实现的.

测试环境中包括 2 台服务器,其间通过 10 Gb/s 链路相连,每台服务器均配有 2 颗 Intel Xeon E7-8891 处理器(10 核 20 线程、3.7 GHz),64 GB 的内存.服务器上运行 Hypervisor,安装有多个 Ubuntu 14.04.1 的操作系统.其中 1 台服务器运行 Cbench[20],用于模拟 SDN 的数据层,向控制层发送 OF 消息;另一台服务器则同时启动多个操作系统,每个操作系统上运行 1 个 OpenDayLight 控制器,且控制器之间能够运行 BFT 算法.对于 Cbench,我们采用固定的配置,模拟的网络中具有 64 个 Switch,每个 Switch 连接有 10^5 个 Host.

4.2 测试数据

4.2.1 BFT 算法的可靠性

基于上述测试环境,我们首先验证 BFT 算法对 SDN 网络可靠性的影响.在控制器数目 $n=10$ 的情况下,对比分析主备切换 Custom,BFT 容错上限为 $f=1$, $f=2$, $f=3$ 这 4 种方法的容错效果,均以 SDN 网络中异常交换机的个数 u 作为其衡量指标.为降低随机过程带来的偶然性,每个实验均重复 1 000 次,计算 u 的平均值.

如图 5 所示,在拜占庭错误比例 $b=70\%$ 固定的情况下,测试不同的控制器出错所对应的网络可靠性.

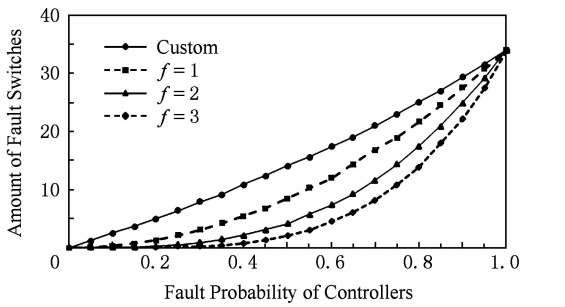

Fig. 5 The reliability of SDN under different fault probability

图 5 不同错误率下 SDN 网络的可靠性

从图 5 中可以得出:1)对于这 4 种方法,其异常交换机的个数 u 均随着错误概率 p 的增大而增加,即主备切换机制和 BFT 算法的可靠性均与控制器的出错概率有关;2)对比主备切换机制和 BFT 算法,在 $b=70\%$ 的情况下,BFT 的容错能力要优于主备切换机制;3)在不同 p 值下,容错上限 f 越大的 BFT 算法,其可靠性均越高.

在控制器出错概率 $p=0.6$ 固定的情况下,测试不同的拜占庭错误比例 b 所对应的网络可靠性,结果如图 6 所示:

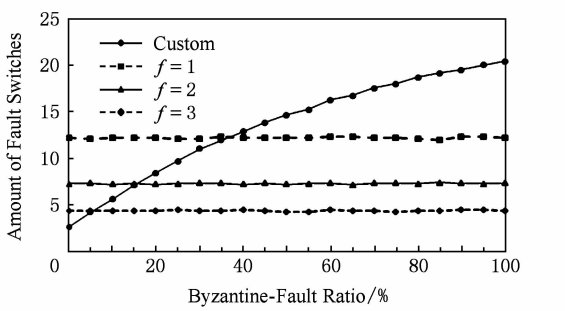

Fig. 6 The reliability of SDN under different Byzantine-fault percentage

图 6 不同拜占庭错误比例下 SDN 网络的可靠性

分析其中的数据,可以得出:1)BFT 算法能够有效应对拜占庭错误,其可靠性与拜占庭错误的比例无关;2)主备切换机制的可靠性随着拜占庭错误比例的增大而降低,在 b 值较大时,其容错能力大幅度降低;3)在不同的 b 值下,BFT 算法的 f 值越大,其容错能力均越高.

综合上述分析,我们可以发现,相对于传统的主备切换机制,BFT 算法不仅能够应对 SDN 网络中的良性错误,而且也能能够处理主备切换机制难以解决的拜占庭错误.同时,随着部署控制器数量的增加,BFT 算法可以选择更大的 f 值,网络的可靠性将会进一步地提高.

4.2.2 BFT 算法的性能

基于 4.1.2 节中所述的原型系统,对 BFT 算法的性能进行测试,测试指标包括处理 OF 请求的吞吐量和平均延迟.我们对比分析主备切换 Custom,以及 BFT 容错上限为 $f=1$,$f=2$,$f=3$ 这 4 种方法下控制器的性能,同样每种方法均重复实验 10 次,每次实验运行 60 s,以 OF 消息包为统计单位,计算控制层的吐吞量(throughput)和处理延迟(latency)的平均值.

从图 7 的测试结果中可以看出,在采用 BFT 算法时,控制器处理 OF 请求的吞吐量有所下降,与传统的主备切换方法相比,吞吐量最大降低 24.07%(当 $f=3$ 时),且不同 f 下 BFT 算法的吞吐量相差在 16.13%以内.另一方面,采用 BFT 算法时,控制器处理 OF 请求的平均延迟增加比较明显,是传统主备切换方法的 2 倍多,但不同 f 下 BFT 算法的平均延迟差别较小,在 6.25%以内.这是由于控制器处理 OF 请求所需的计算和存储资源相对较少,系统中即使加入了 BFT 算法中的其余任务,在处理并发的 OF 请求时,吞吐量降低的相对较小.然而,对于处理一个 OF 请求,由于 BFT 算法的繁杂交互,使得其处理延时明显增加.另外,由于 BFT 算法中控制器数目的增加并未影响 OF 请求的处理流程,这也使得在改变 f 时平均处理延迟的变化较小.

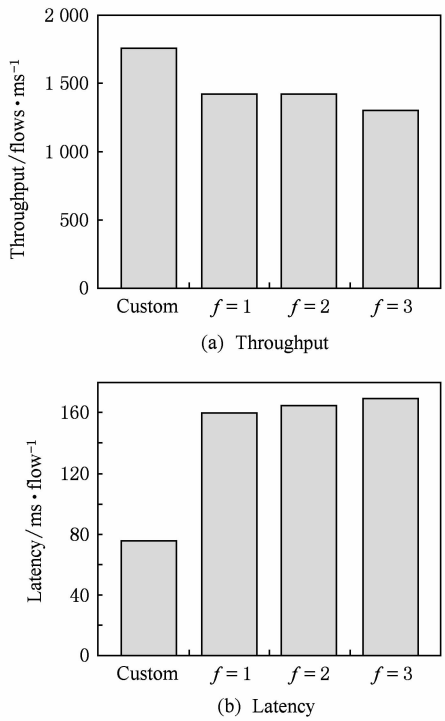

Fig. 7 The performance of BFT

图 7 BFT 算法的性能

4.2.3 DGA 算法的有效性

为了测试 DGA 算法的有效性,我们将其与最优部署(OP)、随机部署(RD)进行比较,计算不同部署方法下控制器传输 OF 请求的平均延迟.对于 OP,该仿真器采用穷举的方法搜索最优解,因此,为了使其运行的时间可接受,CDB 问题的规模选择在 $n\leqslant 7$ 的范围内.同时,我们假定信号在链路中的传输速度 $v=3\times10^8$ m/s.

仿真结果如图 8 所示,我们可以发现,采用 DGA 算法来部署控制器,其传输 OF 的平均延迟时间 T,与采用 OP 方法的最优值较为接近(差距在 86%以内),均维持在 100 ms 以内,远优于 RD 方法的解(与最优值相差近 350%).因此,在多项式时间内求解 CDB 问题,该 DGA 算法具有较好的效果.

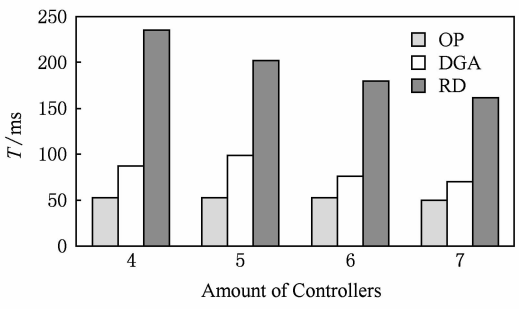

Fig. 8 The average transmission delay time under different algorithms

图 8 不同算法下的平均传输延迟时间 T

5 总结和未来工作

本文针对 SDN 网络中的可靠性问题,首先介绍和分析了该领域的相关工作和研究.然后,提出了在 SDN 中应用 BFT 算法来增强可靠性的方法,并具体阐述了其网络结构、工作流程和异常处理等.再次,针对在 SDN 网络中应用 BFT 算法时存在的多控制器部署问题,给出了求解其近似最优解的启发式算法.最后,将该 BFT 算法与传统的主备切换方法进行对比,实验结果表明,BFT 能够有效提高 SDN 网络的可靠性,但对系统的性能会造成一定程度的影响.在未来的工作中,我们期望能够针对该原

型系统做进一步的优化,调整交互模块与控制器的架构,并引入并行处理机制,以减少对系统性能的影响.

参 考 文 献

[1] Diego K, Fernando M V, Paulo V. Towards secure and dependable software-defined networks [C] //Proc of the 3rd Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2013: 55-60

[2] Banerjee S, Kannan K. Tag-in-tag: Efficient flow table management in SDN switches [C] //Proc of the 10th Network and Service Management (CNSM). Piscataway, NJ: IEEE, 2014: 109-117

[3] Al-Shaer E, Al-Haj S. FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures [C] //Proc of the 3rd ACM Workshop on Assurable and Usable Security Configuration. New York: ACM, 2010: 37-44

[4] Open Network Foundation. OpenFlow Switch Specification Version 1.2 [S]. Reston, VA: The Internet Engineering Task Force, 2011

[5] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery [J]. ACM Trans on Computer Systems, 2002, 20(4): 398-461

[6] Correia M, Costa P, Pasin M, et al. On the feasibility of Byzantine fault-tolerance mapreduce in clouds-of-clouds [C] //Proc of the 31st Reliable Distributed Systems. Piscataway, NJ: IEEE, 2012: 448-453

[7] Casado M, Koponen T, Ramanathan R, et al. Virtualizing the network forwarding plane [C] //Proc of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow. New York: ACM, 2010: 8-17

[8] Shin S, Porras P A, Yegneswaran V, et al. FRESCO: Modular composable security services for software-defined networks [C] //Proc of the ISOC Network and Distributed System Security Symp. New York: ACM, 2013

[9] Hu Yannan, Wang Wendong, Gong Xiangyang, et al. Control traffic protection in software-defined networks [C] //Proc of Global Communications Conf. Piscataway, NJ: IEEE, 2014: 1878-1883

[10] Jagadeesan N A, Pal R, Nadikuditi K, et al. A secure computation framework for SDNs [C] //Proc of the 3rd Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2014: 209-210

[11] Hu Hongxin, Han W, Ahn G, et al. FLOWGUARD: Building robust firewalls for software-defined networks [C] //Proc of the 3rd Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2014: 97-102

[12] Li He, Li Peng, Guo Song, et al. Byzantine-resilient secure software-defined networks with multiple controllers in cloud [J]. IEEE Trans on Cloud Computing, 2014, 2(4): 436-447

[13] Wang Yongjian, Pei Xiang, Li Tao, et al. Nova-BFT: A replicated state machine protocol supporting multiple fault models [J]. Journal of Computer Research and Development, 2011, 48(7): 1134-1145 (in Chinese)
(王永剑, 裴翔, 李涛, 等. Nova-BFT: 一种支持多种故障模型的副本状态机协议[J]. 计算机研究与发展, 2011, 48(7): 1134-1145)

[14] Yung M. The mobile adversary paradigm in distributed computation and systems [C] //Proc of the 2015 ACM Symp on Principles of Distributed Computing. New York: ACM, 2015: 171-172

[15] Hock D, Hartmann M, Gebert S, et al. Pareto-optimal resilient controller placement in SDN-based core networks [C] //Proc of the 25th Teletraffic Congress. Piscataway, NJ: IEEE, 2013: 1-9

[16] Hu Yannan, Wang Wendong, Gong Xiangyang, et al. On reliability-optimized controller placement for software-defined networks [J]. China Communications, 2014, 11(2): 38-54

[17] Internet2 Open Science. Scholarship and services exchange [OL]. [2015-12-10]. http://www.internet2.edu/network/ose

[18] Medved J, Varga R, Tkacik A, et al. Opendaylight: Towards a model-Driven SDN controller architecture [C] //Proc of the 2014 IEEE Int Symp on Network. Piscataway, NJ: IEEE, 2014: 1-6

[19] Bessani A, Sousa J, Alchieri E. State machine replication for the masses with BFT-SMaRt [C] //Proc of 2014 Annual IEEE/IFIP Int Conf on Dependable Systems and Networks. Piscataway, NJ: IEEE, 2014: 355-362

[20] Sherwood R, Yap K. Cbench controller benchmarker [J]. Control Engineering Practice, 2011, 19(11): 1253-1265

Li Junfei, born in 1989. PhD candidate. His main research interests include SDN and network security.

Hu Yuxiang, born in 1982. PhD and associate professor. His main research interests include the next generation information network.

Wu Jiangxing, born in 1953. Professor and PhD supervisor. His main research interests include network architecture and security.