

基于多控制器的 SDN 故障检测算法研究

樊自甫, 杨先辉, 姚 杰

(重庆邮电大学 通信与信息工程学院, 重庆 400065)

摘 要: 针对 SDN 控制器故障检测算法中通常采用固定超时时间间隔的检测方式, 本文提出一种基于灰色预测的故障检测算法. 该算法对网络的动态变化采用自适应的方式预测下一次消息的到达时间并设定门限值. 仿真结果表明, 本文所提算法能够快速实现故障定位, 并且保证了检测点的可靠性.

关键词: SDN; 故障检测; 灰色预测; 自适应性

中图分类号: TP393

文献标识码: A

文章编号: 1000—7180(2018)08—0073—05

The SDN Fault Detection Algorithm Research Based on Multi-Controller

FAN Zi-fu, YANG Xian-hui, YAO Jie

(College of Communication and Information Technology, Chongqing University of
Posts and Telecommunications, Chongqing 400065, China)

Abstract: For the SDN controller fault detection algorithm is usually used fixed time interval detection method, a fault detection algorithm based on gray prediction is presented in this paper. The algorithm predicts the arrival time of the next message and sets the threshold value in an adaptive way for the dynamic change of the network. The simulation results show that the algorithm proposed in this paper can achieve fault location quickly and ensure the reliability of the detection point.

Key words: SDN; fault detection; gray prediction; adaptation

1 引言

软件定义网络 (Software Defined Network, SDN) 作为一种新型的网络体系架构, 利用其转控分离、可编程的特性解决了传统网络中网络设备复杂、配置困难等问题. 典型的单控制器软件定义网络虽然能够在继承 SDN 特点的基础上实现网络构建, 但是单控制器网络存在一些难以克服的固有缺陷, 如单点故障以及可扩展性问题^[1-4]. 为解决 SDN 的单点故障以及可扩展性问题, SDN 多控制器架构应运而生^[5-7]. 对于 SDN 中多控制器控制节点的故障检测有两种基本的检测方案: 方案一, 在控制节点之外采用外部故障检测器^[8]. 通过外部故障检测器进

行各个控制节点的故障检测, 如 HyperFlow, RMS 等. 然而这类方案在实现检测器与控制器之间的串口时相当繁琐, 并且在大多数应用场景中故障检测器都只有一个, 因此检测作业将在检测器发生故障时无法进行; 方案二, 控制器之间的相互检测^[9], 以信息交互的方式为主. 在不引入外在负载的情况下充分利用检测方的剩余资源进行故障计算, 以最终计算结果判断被检测方的状态.

文献[10-11]提出控制器之间以相互发送心跳信息的方式进行故障检测 (以 A、B 控制器相互发送心跳信息为例), 控制器 A 以固定时间间隔向控制器 B 发送心跳信息并设置超时门限值, 在接收到心跳信息之后控制器 B 告知 A 自身的存活状态. 在超

收稿日期: 2017—11—23; 修回日期: 2017—12—26

基金项目: 国家自然科学基金(61701064); 重庆市教委科学技术项目(KJ1600424); 重庆邮电大学博士科研启动基金(A2015—41); 重庆邮电大学青年科学基金(A2015—62)

时时间间隔内,控制器 A 收到 B 的存活信息则判定 B 正常,否则判定 B 发生故障. 文献[12]中每个控制器记录各自的当前时间戳(TimeStamp),并引入额外的服务器作为故障检测点,以固定时间间隔向其他控制器发布获取时间戳消息并设置超时时间,其他控制器在收到消息后向检测控制器发送自身的时间戳信息,在超时时间间隔内检测点未收到其中某个控制器的时间戳信息,则将此控制器设置为怀疑状态,于是通知所有控制器向处于怀疑状态的控制器发送心跳信息判断故障与否,通过依次询问所有控制器的查询结果判定故障是否发生. 文献[13]对文献[12]的检测机制进行了改进,使得每个控制器以固定时间间隔向角色管理服务器(Role Management Server, RMS)发送自身当前 CPU 利用率, RMS 将超时时间间隔内未发送 CPU 利用率的控制器设置为怀疑状态,并通知所有控制器向该控制器发送心跳信息查询其运行状态. 为了减小消息量开销, RMS 向查询控制器发送查询消息得知故障是否发生,发送的消息量为查询控制器个数的一半.

综上所述,目前对于多域 SDN 中控制器的故障检测算法中通常采用固定超时时间间隔的方式. 由于网络动态性,控制器在不断收集网络参数进行统计的同时,还要根据需求进行重路由、负载均衡等工作,将会消耗大部分资源,因此消息的发送将会产生随机延迟. 为此,本文提出一种基于灰色预测的故障检测算法,对网络的动态变化采用自适应的方式预测下一次消息的到达时间,设定门限值,实现了故障的快速定位,并且保证了检测点的可靠性.

2 故障检测模式

心跳(HeartBeat)是故障检测中一种常用模式,通过固定时间周期向对方发送心跳信息检测对方是否存活. 按实现方式划分为两种模式: Pull 模式与 Push 模式.

(1) Push 模式: Push 工作模式如图 1 所示,进程 p 以设定的固定时间间隔 T_1 不断的向进程 q 发送“*I am alive*”心跳信息,其目的是表明自身的存活状态. 如果心跳信息的到达时间超过了进程 q 设定的超时门限值 T_2 ,其中 $T_2 = T_1 + t$ (t 为心跳信息在传输过程中的延迟),此时将认为被进程 p 发生了故障.

(2) Pull 模式: 在 Pull 模式中,心跳信息的工作原理如图 2 所示,进程 p 按照一定的时间间隔 T_1 向进程 q 发送“*Are you alive?*”消息查询信息,进程 q

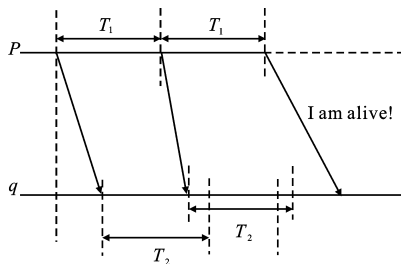


图 1 心跳信息发送的 Push 模式

设定一个超时门限值 T_2 ,如果进程 q 在 T_2 时间段内收到此信息后会向进程 p 发送一个“*yes*”信息以做出回复,进程 p 在收到回复信息后认为 q 是正常. 然而这两种信息(“*Are you alive?*”, “*yes*”)在传输的过程中有可能会发生丢失,无论哪种信息的丢失都会对进程 P 的判断造成影响.

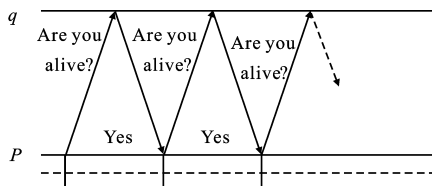


图 2 心跳信息发送的 Pull 模式

本文采用 Pull 模式与 Push 模式相结合检测方式,由于 Push 模式检测过程较短的时间周期和少量的通信开销,因此在常规检测中全部使用 Push 模式发送心跳信息. 当在 Push 检测模式下心跳信息到达时间超过了进程 q 设定的超时门限值 T_2 后,则认为 p 疑似失效并迅速启动 Pull 检测机制. 此时 q 通过 Pull 模式发送询问信息查询 q 的存活状态,如果得到 q 的回复,判定 q 正常,否则判定 q 发生故障.

3 自适应灰色预测算法

3.1 灰色预测算法

本文采用一种自适应的灰色预测的方法(The Detection Of Adaptive Gray Model Algorithm, 后文简称 DAGMG)对 SDN 控制器进行故障检测,算法思想描述如下:

假设集合 $C_i = \{C_1, C_2, C_3, \dots, C_i\}$ 为控制器集合, $\eta_i = \{\eta_1, \eta_2, \eta_3, \dots, \eta_i\}$ 为任意 C_k 上存在的故障检测模块集合,集合 η_i 中的任意元素包含了一定时间内消息到达时间集合 σ_i .

以控制器 A、B 为例,控制器 A、B 上分别存在进程 P_A 、 P_B . 进程 P_A 向进程 P_B 以固定时间间隔 T 发送心跳信息,则进程 P_A 发送的原始心跳信息的到达时间集合 $\sigma_i = \{\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n\}$,其中 σ_n 为第 n 个心

跳信息的到达时间.心跳信息到达时间呈现随机性,为使得无规律的到达时间呈现规律性,将心跳信息的到达时间序列进行累加.

$$\sigma_i(1) = \{\sigma_1(1), \sigma_2(1), \sigma_3(1), \dots, \sigma_n(1)\} \quad (1)$$

式中, $\sigma_K(1) = \sum_{i=1}^K \sigma_i(1)$, $K = 2, \dots, n, \sigma_1(1) = \sigma_1$

当序列 $\sigma_i(1)$ 满足光滑性检公式(2)、准指数规律检验公式(3)以及级比检验公式(4)时,称序列 $\sigma_i(1)$ 具有指数增长规律.

$$\rho(i) = \sigma_i(0)/\sigma_{i-1}(1) < 0.5 \quad (2)$$

$$\delta(i) = \sigma_i(1)/\sigma_{i-1}(1) \in [1, 1.5] \quad (3)$$

$$\xi(i) = \sigma_i(0)/\sigma_{i-1}(0) \in (e^{-\frac{2}{a+1}}, e^{\frac{2}{a+1}}) \quad (4)$$

利用 $\sigma_i(1)$ 建立 GM(1,1) 模型相应方程,其 a 与 b 分别表示灰色系统参数与作灰量.

$$\frac{d\sigma_i(1)}{di} + a\sigma_i(1) = b \quad (5)$$

利用最小二乘法对 GM(1,1) 模型方程进行求解,求解如式(6)所示.

$$[a \ b]^T = (B^T \cdot B)^{-1} \cdot B^T \cdot \sigma_i(1) \quad (6)$$

其中 $B = [a \ b]^T$ 矩阵,可通过式(8)求得:

$$B = \begin{bmatrix} -0.5(\sigma_1(1) + \sigma_2(1)) & 1 \\ -0.5(\sigma_2(1) + \sigma_3(1)) & 1 \\ \vdots & 1 \\ -0.5(\sigma_{i-1}(1) + \sigma_i(1)) & 1 \end{bmatrix} \quad (7)$$

通过对式(5)的求解可得预测模型:

$$\sigma_i(2) = (\sigma_i(1) - \frac{b}{a}) \times e^{a(i-1)} + \frac{b}{a} \quad (8)$$

序列 $\sigma_i(2)$ 通过累减的得到初始序列 σ_i 的下一预测值 σ_{i+1}' .

$$\sigma_{i+1}' = (\sigma_i(2) - \sigma_{i-1}(2)) \quad (9)$$

3.2 自适应性

采用固定窗口的方式对预测结果进行处理并设定超时间隔.在初始状态时设置进程 P_B 接收心跳信息的门限值为 T .假设初始序列为 $\sigma_i(1)$,在下次心跳信息到达后,去掉预测序列中初始值并带入实际到达值到初始序列进行下一次预测.

$$\sigma_i(1) = \{\sigma_2(1), \sigma_3(1), \dots, \sigma_i(1), \sigma_{i+1}(1)\} \quad (10)$$

式中, $\sigma_{i+1}(1)$ 为下一次心跳信息到达时间的实际值.在下次预测时对序列 $\sigma_i(1)$ 进行下一次的预测,从而实现自适应性.

为了提高预测精度,设定预测偏差 ω ,表示窗口大小为 a 时原始序列的波动情况.

$$\omega = a^{-1} * \sum_{i=1}^n (\sigma_i(1) - \bar{\sigma}_i(1)) \quad (11)$$

式中, $\sigma_i(1)$ 表示进行第二次预测时前一次心跳信息的实际到达值; $\bar{\sigma}_i(1)$ 表示序列中心跳信息到达的均值.

经过第二次预测后,将预测结果与预测偏差相加得到预测门限值 θ_{i+1} .

$$\sigma_{i+1}(1) > \theta_{i+1} \quad (12)$$

若下一次心跳信息到达的实际值 $\sigma_{i+1}(1)$ 大于预测门限值 θ_{i+1} ,则进程 P_B 怀疑进程 P_A 发生了故障,此时 P_B 通过 Pull 模式发送问询信息查询进程 P_A 的存活状态,如果得到 P_A 的回复,则判定控制器正常,否则判定控制器发生故障.

4 仿真分析

为验证所提算法的有效性,采用 Python2.7.3 系统仿真实现客户端及服务端,其中客户端代表了被检测进程 p ,服务端代表了检测进程 q .为保持实验环境相同,设置消息的固定发送周期 $T=1$ s.

如图3所示,在随机时间段内选取部分心跳信息的预测时间求得平均值,与静态灰色预测时间值进行对比.由计算得知,经同等级量化后,实际序列、静态灰色预测、自适应灰色预测的方差值分别为:0.16,0.004 1,0.068.由结果可知,自适应灰色预测算法能随着网络的动态变化而发生变化,与实际序列的波动性更加接近.

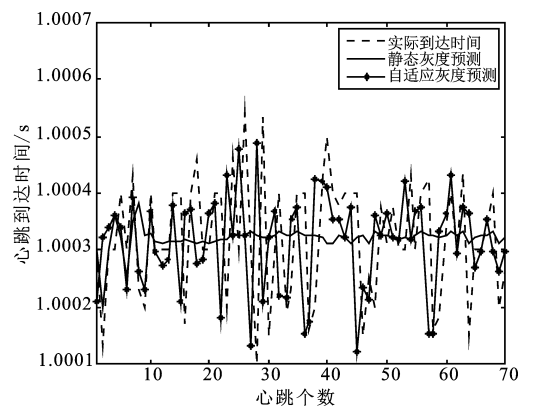


图3 静态灰色预测与自适应灰色预测对比

如表1所示,表示窗口大小对 DAGMG 算法的检测时间与错误率的影响.窗口大小在区间[1 000, 1 500]之间检测时间会发生明显的变化,这是因为随着窗口的增大进程需要计算的次数增多,消耗的时间增大,检测时间整体上随窗口的增大而增加.窗口大小在区间[500, 1 000]时错误率变化的幅度很

大,这是因为当窗口达到 1 000 时,有 1 000 个可进行预测计算的值,预测的准确率在提升,在此之后的错误率的变化比较平稳. 因此对于错误率而言,接收到的信息个数越多,对最后的预测越精确.

表 1 窗口大小对平均检测时间及错误率的影响

窗口大小	平均检测时间/s	错误率(10^{-3})
500	1.000 98	4.131 2
1 000	1.001 01	2.552 1
1 500	1.001 24	2.062 8
2 000	1.001 43	1.742 3
2 500	1.001 56	1.237 7
3 000	1.001 68	0.901 8
3 500	1.001 89	0.673 5
4 000	1.002 06	0.537 2

如图 4 所示,表示 DAGMG 算法与 HyperFlow、RMS 故障算法所需的平均检测时间对比. HyperFlow 在得知客户端未回复核对请求信息时,检测者将依次通知其他所有控制器向怀疑者发送查询信息并按优先级依次询问,只有在所有查询者都向消息发布者发送肯定回答时,才会解除对怀疑者的怀疑判定为正常状态. RMS 在得知客户端未发送 CPU 利用率时,服务端向其他所有客户端发送消息,使得其他客户端向怀疑者发送的查询信息并得到一半以上肯定回答时解除怀疑,因在 RMS 中未定义控制器的优先级,因此 RMS 的消息发送是统一的且仅有一次. DAGMG 算法的信息交互只有一次,显然在相同的发送时间间隔下平均检测时间比两者都低.

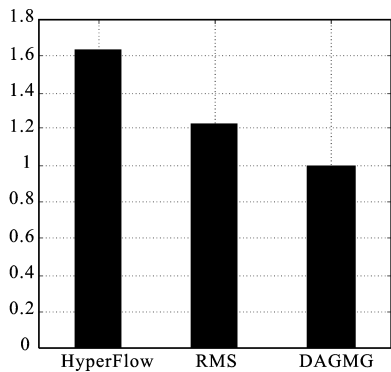


图 4 窗口大小对平均检测时间的影响

如图 5 所示,表示 DAGMG 算法与 HyperFlow、RMS 的平均错误率进行对比. 在 HyperFlow 中,当有控制器发生故障时,消息发布者会通知其他完好的控制器对怀疑者一一发起查询信息,查询的

先后是通过随机定义的优先级,因此 HyperFlow 的确认次数为当前完好控制器的个数,即除怀疑者之外所有控制器的个数. 因此 HyperFlow 的平均错误率是最低的,然而检测所花费的时间是最长的. 在 RMS 中当查询控制器的肯定回答数量达到一半时,服务器会确定怀疑者发生了故障,然而服务器有一定的几率从当前存活的选择怀疑者作为查询对象,导致肯定回答的数量不足一半. 这是 RMS 错误率高于 HyperFlow 的主要原因. DAGMG 算法中通过 1 对 1 的方式进行结果查询,类似于 HyperFlow,在查询消息不发生丢失的情况下 DAGMG 算法的错误率低于 RMS,然而 DAGMG 的确认次数只有一次,远小于 HyperFlow,因此错误率要高于 HyperFlow.

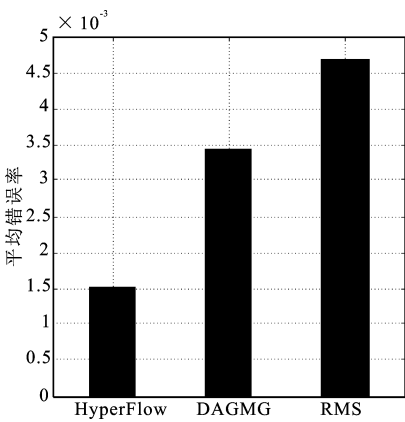


图 5 窗口大小对错误率的影响

5 结束语

针对于目前多域 SDN 控制器故障检测点单一,采用固定超时时间间隔等问题,本文提出一种基于灰色预测的故障检测算法(DAGMG),针对网路的动态变化采用了自适应的方式对以往的消息进行预测,并设定门限值. 通过仿真与 HyperFlow, RMS 算法从检测时间及错误率进行对比. 本文所提 DAGMG 算法在检测时间上低于 RMS 与 HyperFlow,错误率低于 RMS,高于 HyperFlow 错误率. 故本文所提算法在保证检测点的可靠性情况下,快速实现控器故障定位.

参考文献:

[1] 江国龙,付斌章,陈明宇. SDN 控制器的调研和量化分析[J]. 计算机科学与探索,2014, 8(6):653-664.
[2] 李军飞,兰巨龙,胡宇翔,等. SDN 多控制器一致性的量化研究[J]. 通信学报, 2016, 37(6): 86-93.

- [3] 张联镇,黄传河,曾毓珑,等. SDN 中基于时延优化的多控制器部署方案[J]. 计算机工程与设计, 2015, 36(5):1121-1125.
- [4] Murat K, Arjan D. Quality of Service (QoS) in software defined networking (SDN): a survey[J]. Journal of Network & Computer Applications, 2017, 80(15): 200-218.
- [5] Gonzalez A J, Nencioni G, Helvik B E, et al. A fault-tolerant and consistent SDN controller[C]//Global Communications Conference 2016 IEEE. Washington, IEEE, 2016:1-6.
- [6] Athmiya N S, Shobha K R, Sarimela V. Feasibility study and implementation of open flow based SDN controller for tactical scenario[C]//2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). Bangalore; IEEE, 2016:789-794.
- [7] Schiff L, Schmid S, Canini M. Ground control to major faults: towards a fault tolerant and adaptive SDN control network[C]//2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W). Toulouse, IEEE, 2016: 90-96.
- [8] Schmid S, Suomela J. Exploiting locality in distributed SDN control[C]//Acm Sigcomm Workshop on Hot Topics in Software Defined Networking. New York, ACM, 2013:121-126.
- [9] Fonseca P, Bennesby R, Mota E, et al. A replication component for resilient OpenFlow-based networking

- [C]//2012 Network Operations and Management Symposium. Maui, IEEE, 2012:933-939.
- [10] Tam S W, Xi K, Chao H J. Scalability and resilience in data center networks: dynamic flow reroute as an example[C]//2011 The Global Communications Conference, Houston, IEEE. 2011:1-6.
- [11] Ko H, Suh D, Baek H, et al. Optimal placement of service function in service function chaining[C]//2016 Eighth International Conference on Ubiquitous and Future Networks, Vienna, IEEE, 2016:102-105.
- [12] Tootoonchian A, Ganjali Y. HyperFlow: a distributed control plane for OpenFlow[C]//2010 Internet Network Management Conference on Research on Enterprise Networking. San Jose, USENIX Association Berkeley, 2010:3-3.
- [13] Kuroki K, Matsumoto N, Hayashi M. Scalable open flow controller redundancy tackling local and global recoveries[C]//2013 The Fifth International Conference on Advances in Future Internet. Barcelona, IARIA, 2013:61-66.

作者简介:

樊自甫 男, (1977-), 硕士, 教授. 研究方向为下一代网络技术、通信运营管理.

杨先辉(通讯作者) 男, (1990-), 硕士研究生. 研究方向为下一代网络技术、SDN. E-mail: 931425356@qq.com.

姚杰 男, (1994-), 硕士研究生. 研究方向为下一代网络技术、SDN.

(上接第 72 页)

- [6] Dahl G E, Sainath T N, Hinton G E. Improving deep neural networks for LVCSR using rectified linear unit-sand dropout[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. Vancouver, Canada, IEEE, 2013:8609-8613.

作者简介:

仇越 女, (1992-), 硕士研究生. 研究方向为可重构计算. E-mail: 942979548@qq.com.

马文涛 男, (1981-), 工程师. 研究方向为分布式计算.

柴志雷 男, (1975-), 博士, 副教授. 研究方向为嵌入式系统设计技术、FPGA 的可重构计算等.