DOI:10.13718/j.cnki.xsxb.2017.03.025

SDN 的监控框架与自适应监控算法①

姜 彬[1], 吴东领[2]

1. 南通航运职业技术学院 管理信息系,江苏 南通 226000; 2. 唐山职业技术学院 信息工程系,河北 唐山 063000

摘要:针对软件定义网络(SDN)的监控效果问题,提出了一种可对网络应用屏蔽底层细节的软件定义网络监控框架及一种自适应的的网络监控算法.该算法通过可变频率流量统计策略动态调整监控采样周期,从而在监控准确率和网络负载之间维持平衡.监控框架通过应用程序接口为上层应用提供监控算法接口,并与底层网络进行数据交互,从而实现了屏蔽底层网络复杂性的监控.实验结果表明,该框架与算法具有较小的监控误差和较低的流量负载,是一种理想的 SDN 监控方法.

关 键 词:软件定义网络;监控;自适应;流量负载;OpenFlow

中图分类号:TP391 **文献标志码**:A **文章编号**:1000-5471(2017)03-0151-06

软件定义网络(Software Defined Networks, SDN)是近年来提出的全新网络架构,将网络设备控制面与数据面分离开来,从而实现了网络流量的灵活控制,为核心网络及应用的创新提供了更加灵活的平台[1].在 SDN 网络设备控制面与数据面交互方面,目前最成熟的协议是 OpenFlow,本文的工作也将基于此协议.OpenFlow 的基本思路是网络设备维护一个 FlowTable,并且只按照 FlowTable 进行转发,FlowTable 本身的生成、维护、下发完全由外置的控制器来实现[2].

对于软件定义网络而言,网络监控是具有核心重要性的工作,是影响 SDN 应用效果的关键[3].因此,它引起了这一新兴领域内相关学者的高度重视,目前已经有一些研究成果问世.

MonSmap 是一种基于 OpenFlow 的 SDN 监控算法,该算法通过对流量采样的优化实现了低网络负载的监控[4].然而,该方法仅实现了 SDN 的 QoS 监控.对于 SDN 网络监控而言,除了 QoS 之外,流量监控、负载均衡等其他因素也很重要[5],因此需要更加全面的监控方法.分布式协作监控(DCM)是一种分布式的 SDN 监控系统,它可以实现数据流级的运行监控[6],但该系统不是基于 OpenFlow 的,其通用性比较有限.Dixon 等[7]学者提出了一种基于随机轮询的 SDN 监控方法,该方法可以在一定程度上降低监控给网络带来的额外开销,但也影响了监控的效果.文献[8]提出了一个专门针对基于 OpenFlow 的 SDN 网络监控模型 AC-SDN,该模型通过对不同数据流聚合的监控,实现了较高的监控精度,但该模型的问题是监控给网络带来的额外负载有多少.

1 系统架构

典型的 SDN 与本文提出的监控架构构成了本文研究的总体系统,该系统的体系结构如图 1 所示.

在图 1 中,OpenFlow 网络的控制器可以是 NOX[9]等方法.控制器可以为客户端的应用和开发提供一个忽略底层网络复杂性和异构性的平台,为上层开发提供应用编程接口(Application Programming Interface, API).上层网络应用可以为流量控制提供接口,并使上层了解如数据流、数据包等不同聚合程度的数

① 收稿日期:2016-04-17
基金项目:江苏省教育科学"十二五"规划重点资助课题(苏教科规领[2015]1 号);南通航运职业技术学院院级课题(2013HYJY/18).
作者简介:姜 彬(1980-),男,江苏如皋人,硕士,讲师,主要从事人工智能与智能系统研究.

据流量统计.

1.1 本文监控框架

对数据流量统计聚合程度等的要求往往因应用而异. 比如对于互联网运营商而言,对流量统计聚合程度的要求是尽可能达到用户路由级,也就是需要聚合经过同一个用户路由的所有流量. 然而,现有的工具,无论是OpenFlow还是基于OpenFlow的控制器,都无法做到这一点.

因此,需要一个中间层以便从网络和控制器应用中抽象出监控的复杂性. 本文提出了高效率的监控框架(HEMF, high efficiency monitoring framework). 它基于OpenFlow控制器的上层,并提供一种更上层的复合RESTful架构风格的API. 监控架构将负责转换上层的监控请求,并隐藏底层的统计链接和存储管理细节. 监控框架HEMF自身的细节如图2所示.

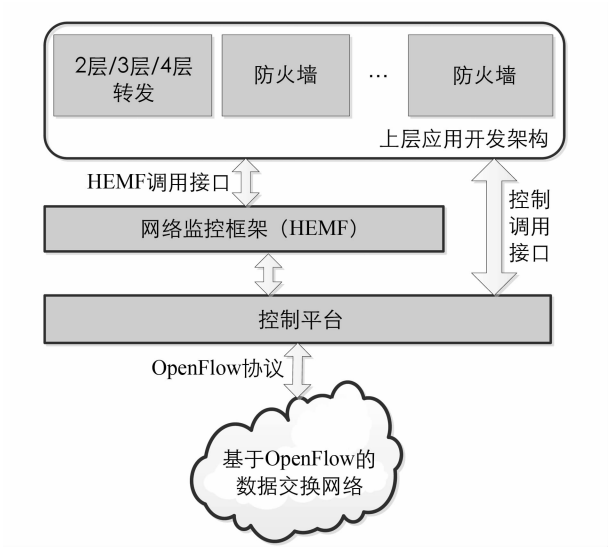

图1 基于SDN的监控系统体系结构

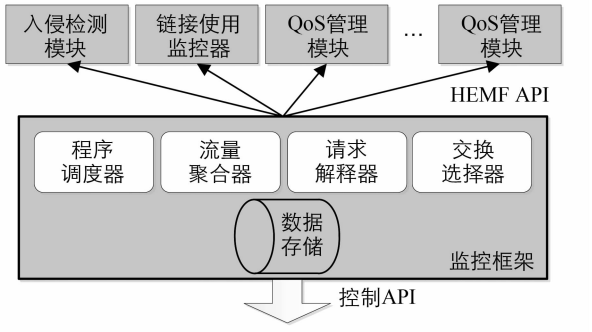

图2 HEMF内部结构

在图2中,程序调度器对网络交换进行轮询以收集数据统计信息. 基于OpenFlow的交换设备可以提供每个数据流的统计、每个队列的数据统计及每个端口的数据统计. 该模块将根据它从应用端接受到的请求来决定轮询何种类型的统计,并且引入了时间戳的机制.

流量聚合器与数据存储单元将汇集程序调度器所选择的轮询交换的原始数据,并将原始数据聚合以计算监控所需的聚合级信息. 数据存储单元存储的实际上是数据持久化系统的概要数据,它可以是关系型数据库、键值型数据库[10]甚至非结构化的文件系统.

请求解释器是用来翻译上层应用请求和底层网络数据的转换模块,它也是其他模块与上层应用及底层网络数据交互的接口.

交换选择器是为统计链接事件选择需要链接的交换设备的单元. 该模块将在轮询时间戳内定义需要轮询的交换器集合,从而使监控框架获得所需的数据统计信息.

1.2 本文的监控API

HEMF监控框架将为上层提供一个符合RESTful架构风格的API,以便支撑不断变化的监控应用需求. 任何编程语言都可以调用这个API,并且高层网络原语可以通过该层得到所需的底层监控数据.

下面将通过实例,演示网络应用如何通过HEMF API访问监控数据. 首先,每个网络应用需要创建一个监控请求对象MoniReq,并将它在HEMF框架注册. MoniReq对象如图3所示.

在图3中"Type"代表网络应用所需监控的数据类别,如性能、容错性或安全性. 其中各项参数定义如下:

"Metrics"代表对于某种选定监控类型,其所包含的测度指标,例如性能类型应该包括带宽、时延、抖动等指标.

"Entity"是一项可选参数,它取决于不同类型的测度指标. 它定义了网络监控的对象,在HEMF中监控对象可以是交换设备、端口、流量表或者数据流.

“Aggregate Level”代表监控数据的聚合级别，它可以是数据流级、端口级、交换机级或者用户级.

“Priority”是规定不同监控测度指标优先级的参数，通过它 HEMF 可以制定合适的轮询顺序和轮询频率.

“Monitor”是决定监控方式的参数，监控方式可以是直接监控、自适应随机采样监控或者最优化监控.

“Logging”是决定网络应用监控日志输出形式的参数，如果不设定该参数，则输出为默认形式.

在 HEMF 框架中，MinoReq 对象以 XML 或 JSON 格式传播，如不特别指定，则默认 JSON 格式. 当该对象在 HEMF 框架中注册成功后，HEMF 将采集监控请求所需的数据并将其存入数据存储单元. 作为监控请求的响应，HEMF 将对网络监控应用返回一个 Access-ID，从而使网络应用可以通过 Access-ID 定位数据存储单元中自己所需的数据.

以 QoS 监控为例，一个实际的 MoniReq 对象如图 4 所示.

在图 4 中，监控系统调用了数据流级别的实时监控数据以选择最优的路由.

HEMF 框架还提供了对 MoniReq 对象进行排列、更新和删除的 API，在此不再赘述.

```
{"MoniReq": {
"Type": "["performance" | "security" | "failure" | ... ]",
"Metrics": [
{"performance": ["latency", "jitter", "throughput",
                 "packet-drop", ...]},
{"security": ["IDS-alerts", "ACL-violations",
              "Firewall-alerts", ...]},
{"failure": ["MTBF", "MTTR"]}
],
"Entity": ["<uri_to_script>"],
"AggregationLevel": ["flow" | "table" | "port" | "switch" |
                     "user" | "custom": "uri_to_script"],
"Priority": ["real-time", "medium", "low",
             custom: "monitoring-frequency"],
"Monitor" : ["direct", "adaptive", "random-sampling",
             "optimized", "custom": "uri_to_script"],
"Logging": ["default", "custom": "<uri_to_log_format>"]
}}
```

图 3　MoniReq 对象示例

```
{"MonitoringRequest": {
"Type": "["performance"]",
"Metrics": [
{"performance": ["throughput", "latency", "jitter",
                 "packet-drop",]},],
"Entity": ["flow": "<flow_specification>"],
"AggregationLevel": ["flow"],
"Priority": ["real-time"],
"Monitor" : ["adaptive"],
"Logging": ["default"]
}}
```

图 4　QoS 监控请求对象实例

2　自适应监控算法

本节提出一种自适应的监控算法，其目标是以较小的额外负载，实现准确、实时的网络监控. 首先假设交换机设备与控制器之间通过 OpenFlow 进行通信，因此需要简要回顾一下 OpenFlow 的消息机制.

OpenFlow 通过网络层(3 层)数据报的报头来定义数据流. 当交换机接收到的数据流与其转发的任何类型都不匹配时，它将给控制器发送一个 PacketIn 消息. 控制器将根据该消息通过回应一个 FlowMod 消息建立相应的转发规则. 控制器还将为每个转发规则定义一个空闲时隙，从而为不同数据流定义了转发失效时间. 当某个数据流失效时，路由器将对控制器发送一个 FlowRemoved 消息，该消息包含数据流的持续时间及数据流的匹配字节数. 此外，控制器可以通过给交换机发送 FlowStatisticsRequest 消息来统计特定的数据流，交换机将通过 FlowStatisticsReply 消息返回该数据流的持续时间和字节数.

一个典型的数据流统计是通过周期性地发送 FlowStatisticsRequest 消息以轮询交换机. 虽然更高频率的轮询将实现更高的统计准确性，但这也将显著增大监控对网络的通信开销. 为了在准确率与网络负载间维持平衡，本文提出了自适应可变频率流量统计连接算法.

当控制器接收到一个 PacketIn 消息时，它将在活动数据流列表中建立一个带有初始统计连接时隙的新数据流对象，其初始时隙为 t 毫秒. 若该数据流在 t 毫秒内过期，则控制器将通过 FlowRemoved 消息得到它的统计数据，否则控制器将返回 timeout 事件，并通过 FlowStatisticsRequest 消息联系交换机以重新得到该数据流的统计数据. 如果在这一时期内收集到的关于该数据流的监控数据变化并不显著，例如数据比

特数的变化没有超过阈值 δ，则时隙 t 将被增大 λ 倍. 对于低数据传输率的数据流，该过程可以重复执行，直到监控采样时隙达到上界 T_{MAX}.

另一方面，当数据流变化很快，超过某一阈值 δ'时，轮询的时隙将被缩减 μ 倍. 对于流量变化剧烈的数据流，此过程也可以重复执行，直到时隙达到下界 T_{MIN}.

以上2个流程还可以通过对同一 timeout 的不同数据流进行 FlowStatisticsRequest 消息的批处理来优化. 这一策略将在不影响监控效果的前提下进一步降低监控的负载. 自适应监控算法的伪代码如图5所示.

```
自适应监控算法伪代码
全局输入：活动数据流、调度表、链接统计量 U
If (e 是初始事件)
    active_flows ←Ø, schedule table ←Ø, U ←Ø
else if (e 是一个 PacketIn 事件)
    f ←(e.switch; e.port; T_MIN; 0)
    schedule_table[T_MIN] ← schedule_table[T_MIN] U f
else if (e 是调度表中的时隙 t)
    for (所有数据流 f ∈schedule_table[t] )do
        发送一个 FlowStatisticsRequest 到 f.switch
else if (e 是数据流 f 的 FlowStatisticsReply 事件)
    diff_byte_count ← e.byte_count –f.byte_count
    diff_duration ← e.duration – f.duration
    checkpoint ← current_time_stamp
    U[f.port][f.switch][checkpoint] ← <dif f_byte_count;
                                       dif f_duration>
    if (dif f_byte count <δ)
        f.t ←min(f.tλ, T_MAX)
        将 f 移到 schedule table[f.t]
    else if (dif f_byte count >δ')
        f.t ← max(f.t/μ; T_MIN)
        将 f 移到 schedule table[f.t]
```

图5 自适应监控算法伪代码

3 实 验

3.1 实验部署

为了验证本文提出的 SDN 监控框架和自适应监控算法，基于 Floodlight 监控平台部署了一个支持 OpenFlow 的实际监控系统. 为了便于测量本文提出方法的性能，本文对系统提出了一些简化的近似假设. 首先，通过源 IP 地址和目标 IP 地址定义数据流，并且在整个实验中只采用用户数据报协议(UDP)模式，并且过滤了其中的动态主机配置协议(DHCP)数据.

部署了具有8个终端节点的三层对称树形网络，除了终端叶子节点，其他节点均为交换机. 网络拓扑如图6所示.

各终端间的 UDP 数据流持续时间为 100 s，每个数据流的空闲时隙被设置为 5 s. 其他参数设置为δ=50 MB，δ'=200 MB，λ=2.5，μ=5.

3.2 监控负载

首先测试了本文提出框架在监控中随时间给网络带来的额外开销，并将 HEMF 方法与周期性轮询策略、MonSamp 算法、DCM 方法及 AC-SDN 方法的实验结果进行了对比. 实验测试周期为 60 s，实验及对比结果如图7所示.

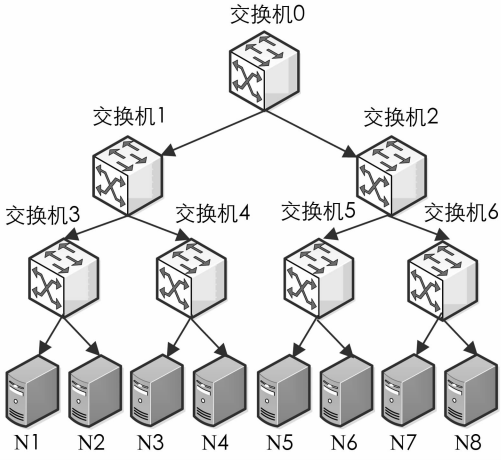

图6 实验网络拓扑

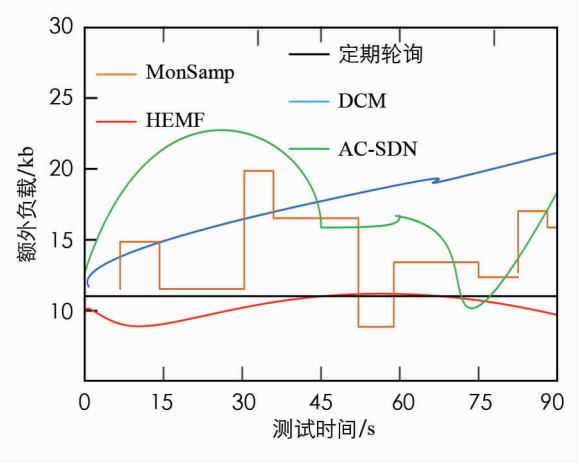

图7 各监控方法带来的网络负载对比

在图 7 中横坐标是测试时间，单位为秒，纵坐标是监控带来的额外负载，单位为 kb. 图 7 表明本文提出的 HEMF 监控框架的网络开销最低，DCM 算法次之，周期性轮询的算法居中，可以最为基线比较方法. AC-SDN 算法和 MonSmap 算法在监控中耗费的流量最大，在网络带宽资源有限的应用场景下，不是理想的选择.

3.3　监控准确度

对于一个网络监控模型或算法的性能评价而言，监控结果的准确性是最重要的指标. 在本小节以监控测量的误差率作为准确度的衡量依据，在不同流量变化率条件下测试了本文提出模型和算法的表现，并与各经典算法做了对比，实验及对比结果如图 8 所示.

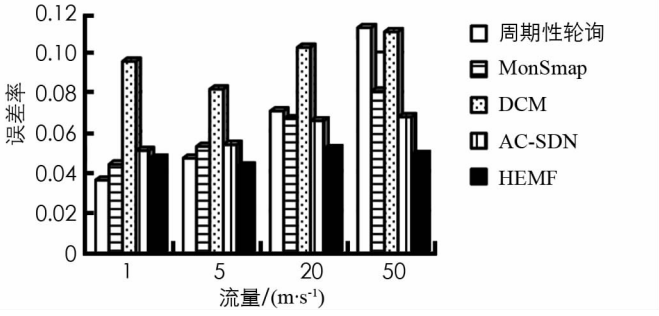

图 8　监控误差率对比

在图 8 中，横坐标是网络的流量情况，纵坐标是监控误差率. 如图 8 所示，周期性的轮询策略由于无法根据挽留的流量进行实时自适应采样，因此其监控误差率随网络流量增大而增大的程度剧烈. 本文提出的 HEMF 框架虽然并非在每种流量下的监控误差率都最低，但其监控性能比较稳定，随网络流量变化而变化的范围最小. 并且，如果考虑各种流量的平均误差率，HEMF 是最低的.

4　总结与展望

作为下一代互联网的关键技术，SDN 是一种革命性的架构[11]，已经引起了学术界和企业界的广泛关注. 网络监控，如用户流量账单监控、QoS 监控、异常监控等是 SDN 应用效果的核心问题.

本文提出了一种基于 OpenFlow 的 SDN 监控框架，该框架的应用程序接口为上层提供访问不同监控聚合级别底层网络的能力，并为其屏蔽底层网络的复杂性. 该框架还集成了本文提出的自适应监控算法，该算法可以根据被监控数据流的变化情况，动态调整监控的采样周期，从而在监控的准确率和对网络造成的额外负载之间维持一定的平衡.

实验结果表明，本文提出方法的监控准确率及给网络所带来的开销都优于各对比算法，是软件定义网络理想的监控工具. 由于网络监控的内容和形式比较复杂多样，在实验分析中本文仅对网络负载和监控结果的准确性两大重要指标进行了对比分析，对其他形式的网络监控进行多指标性能对比分析是下一步要重点研究的内容.

随着 SDN 研究的深入，基于 SDN 的无线网络体系结构正在被提出，基于无线、移动 SDN 的监控方法将对应用产生巨大影响，应当引起相关研究者的重视.

参考文献：

[1] FOSTER N, GUHA A, REITBLATT M, et al. Languages for Software-Defined Networks [J]. IEEE Communication Magazine, 2013, 51(2): 128－134.

[2] 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.

[3] 周　烨，杨　旭，李　勇，等. 基于分类的软件定义网络流表更新一致性方案 [J]. 电子与信息学报，2013，35(7)：1746－1752.

[4] RAUMER D, SCHWAIGHOFER L, CARLE G. MonSamp: A Distributed SDN Application for QoS Monitoring [C]. Warsaw: 2014 Federated Conference on Computer Science and Information Systems(FedCSIS), 2014.

[5] 左青云，孙志刚，李　韬，等. 基于 OpenFlow 的 SDN 技术研究 [J]. 软件学报，2013，24(5)：1078－1097.

[6] Ye Yu, Chen Qian, Xin Li. Distributed Collaborative Monitoring in Software Defined Networks [C]. Chicago: ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2014.

[7] DIXON C, OLSHEFSKI D, JAIN V, et al. Software Defined Networking to Support the Software Defined Environment [J].

IBM Journal of Research and Development, 2014, 58(3): 1—14.
[8] TOOTOONCHIAN A, GORBUNOV S, GANJALI Y. On Controller Performance in Software-Defined Networks [C]. Berkeley: The 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, 2012.
[9] GUDE N, KOPONEN J, PETTIT B, et al. NOX: Towards an Operating System for Networks, SIGCOMM Comput [J]. Commun. Rev., 2012, 38(3): 105—110.
[10] 张建英. 多根层次数据分布模型研究 [D]. 大连: 大连理工大学, 2011.
[11] HU Yan-nan, WANG Wen-dong, GONG Xiang-yang, et al. On The Placement of Controllers in Software-Defined Networks [J]. The Journal of China Universities of Posts and Telecommunications, 2012, 19(2): 92—97.

On Monitoring Framework and Adaptive Monitoring Algorithm for SDN

JIANG Bin[1], WU Dong-ling[2]

1. *Department of Management & Information Technology, Nantong Vocation & Technical Shipping College, Nantong Jiangsu 226000, China;*
2. *Department of Information Engineering, Tangshan Vocation & Technical College, Tangshan Hebei 063000, China*

Abstract: A monitoring framework proposed as an adaptive monitoring algorithm has been proposed to solve the performance problem of Software Defined Networks. The presented algorithm can keep balance between monitoring accuracy and network overhead by a variable frequency traffic statistic strategy which can adjust the monitoring period dynamic. The monitoring framework afford application programming interface to high level applications and interconnect with underlying low-level networks. Experimental result reveals the proposed method has lower monitoring error and network overhead. It is an ideal SDN monitoring method.

Key words: Software Defined Networks; monitoring; adaptive; traffic overhead; OpenFlow

责任编辑 夏 娟