研究与开发

基于 LSTM 流量预测的路由规划和切换

诸葛斌，王林超，宋杨，邵瑜，董黎刚，蒋献
（浙江工商大学信息与电子工程学院（萨塞克斯人工智能学院），浙江 杭州 310018）

摘　要：随着近年网络技术的不断发展和演进，用户对网络资源的需求体量越来越大，内容越来越复杂，传统的网络架构很难满足当前灵活的网络需求。软件定义网络的革命性技术思想赋予了网络可编程性和可演进性，实现网络资源更灵活的管理。为了提高网络资源利用率，提出了基于 LSTM 模型的流量预测，为路由规划提供理论依据。通过提前预测网络流量，在流量激增之前进行防御措施保证网络的安全性和稳定性。实现基于流量预测的路由规划、无损路径切换等功能。

关键词：软件定义网络；流量预测；路由规划；路径切换；LSTM 模型
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2022172

Routing planning and handoff based on LSTM traffic prediction

ZHUGE Bin, WANG Linchao, SONG Yang, SHAO Yu, DONG Ligang, JIANG Xian
School of Information and Electronic Engineering (Sussex Artificial Intelligence Institute),
Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: With the continuous development and evolution of network technology in recent years, the user's demand for network resources is becoming larger and larger, and the content is becoming more and more complex. The traditional network architecture is difficult to meet the current flexible network needs. The revolutionary technology of SDN endows the network with programm ability and evolvability, and realizes more flexible management of network resources. In order to improve the utilization of network resources, a traffic prediction agent based on LSTM model was proposed, which provides a theoretical basis for routing planning. By predicting the network traffic in advance, we can take defensive measures before the traffic surge to ensure the security and stability of the network. The function of route planning and lossless path switching based on traffic prediction was realized.

Key words: software defined network, traffic prediction, routing planning ,paths witching, LSTM model

收稿日期：2021−11−23；**修回日期**：2022−05−27
通信作者：蒋献，jiangxian@zjgsu.edu.cn
基金项目：国家自然科学基金资助项目（No.61871468）；浙江省重点研发计划项目（No.2020C01079，No.2021C01036）；浙江省新型网络标准与应用技术重点实验室（No.2013E10012）
Foundation Items: The National Natural Science Foundation of China (No.61871468), The Key Research and Development Program of Zhejiang under Grant (No.2020C01079, No.2021C01036), The Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (No.2013E10012)

0　引言

运营商基于当前快速增长的网络需求和日益复杂的网络环境提出流量预测需求，OMC（operation and maintenance center）系统应支持基于历史流量数据，通过大数据分析算法进行流量预测，即能根据当前软件定义网络（software defined network，SDN）[1]已有的历史流量数据，结合监督学习、无监督学习等人工智能算法，快速全面预测整体网络流量趋势以指导网络提前进行调整或扩容，并且帮助运营商提前进行大客户流量营销。

互联网和通信设备的快速发展创造了更大、更复杂的网络结构，网络的复杂性导致大量流量数据的溢出，并且给网络管理和流量优化带来了挑战，包括路由决策、负载均衡、拥塞控制等服务。同时，研究人员看到两个可行的解决方案帮助更有效地管理网络——SDN 和机器学习。SDN 为所有的网络设备提供了一个集中的访问和控制机制，SDN 控制器不仅可以监视和测量各种网络参数和度量，而且可以对资源分配和路由做出更明智和高效的决策，因为它可以全局地查看网络中的所有内容。然而，SDN 控制器接收的数据量可能非常大。虽然 SDN 控制器本身可以扩展，例如在云上运行它，但是仍然需要有效的算法从接收的数据中提取所需的测量和信息。这是机器学习可以很好辅助 SDN 的地方。许多流量分类和流量预测问题可以通过各种机器学习算法实现，在保持相对简单设计的同时提高了系统性能。流量预测对 SDN 中路由决策、拥塞控制、负载均衡等服务的开展具有重大的促进作用，是一个基础性的研究领域。

1　基于 LSTM 模型的流量预测算法设计

1.1　流量预测框架和机制

本文提出的 SDN 流量预测系统框架如图 1 所示，所提的 SDN 体系架构是一种基于模型化的架构，各层提供的服务和功能均以元模型为基础，通过资源组合满足上层用户的多元化需求。元业务中包含了上层应用的特性和要求，它可以根据应用的特性和要求抽象业务所需的最基本网络服务功能。控制平面中预测模块为整个系统的核心模块，本文将深度学习模型应用到流量预测中进行网络流量的预测和处理机制的研究，通过与其他各层的模块进行交互，可以实现对转发层的优化调度和灵活转发。

在转发层，将物理设备虚拟化成一个资源池，再通过正交分解将转发层虚拟化成各种逻辑功能块，也就是元能力。与流量预测相关的模块包括流量抓取模块和转发信息库，流量抓取模块负责抓取流量信息，并将信息发送至控制层中的流量统计模块。转发信息库主要用于保存控制层中流与动作管理模块下发的流表信息。

控制层中，流量统计模块用于下发统计信息到流量抓取模块，接收流量抓取模块抓取的流量信息进行分析。然后发送至流量预测模块，流量预测模块通过深度学习算法对流量进行预测。然后将预测结果发送至结果分析模块进行流量分析，分析结果分别发送给控制层的策略管理模块和应用层的应用服务管理元业务模块。策略管理模块制定相应的策略发送给流与动作管理模块。流与动作管理模块将具体的策略下发至转发层的转发信息库。

其中，应用层的元业务是一个单独的微服务，它具有平台无关性和可以支持独立部署的特点。流量预测相关的业务主要包括应用服务管理模块和应用处理模块，应用服务管理元业务用于接收结果分析模块得出流量分析结果。应用处理元业务用于接收应用服务管理元业务转发的操作指令，对具体应用做出对应的操作。

SDN 流量预测框架的流量预测机制是整个架构的核心部分，流量预测机制为流量预测系统提供了高效的防护策略。

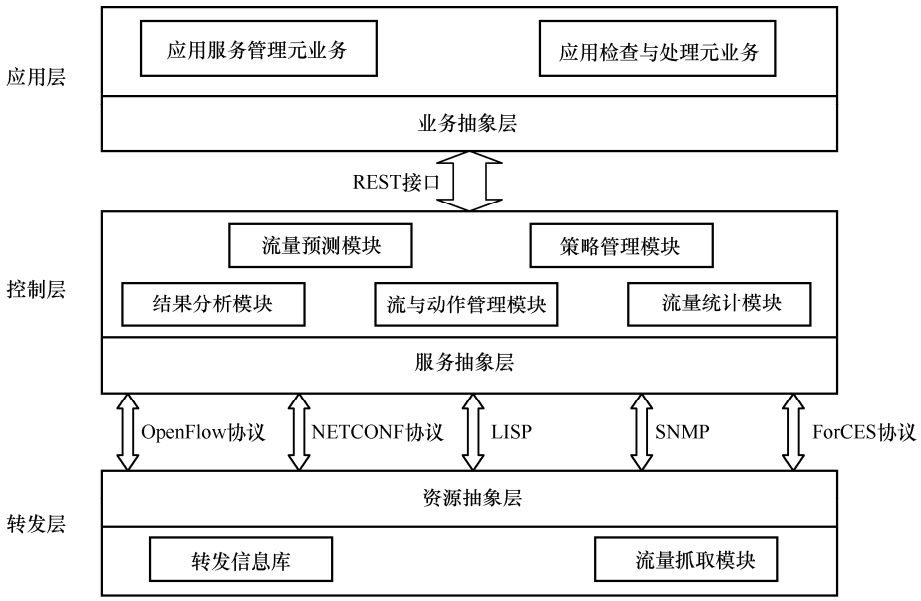


图 1　基于 SDN 的流量预测框架

1.2　数据预处理

目前，流量矩阵的估算主要依赖合成数据的流量矩阵评估某个算法的具体效果。几乎没有主流的机构可以提供流量矩阵的数据，直到 Uhilg 等[2]公布了他们在欧洲探索与学习网 GEANT 上获得的真实可用的流量矩阵数据集。这份流量矩阵数据集囊括了 GEANT 中全内部网关协议（Interior gateway protocol，IGP）的流量信息、抽样了 NetFlow 数据和边界网关协议（border gateway protocol，BGP）路由信息一同建立的流量矩阵。真实公开的流量矩阵可以加强对真实流量矩阵及其动态性的理解。因此，本文采用了 CEANT2 上部署的 25 个节点，收集节点之间链路的流量作为数据集，然后通过流量数据测量和估算方法计算流量矩阵，再通过流量矩阵进行流量预测。

首先对数据集进行处理，因为下载的数据集以 15 min 为间隔，为期两周左右的 977 个 XML 文件，每个 XML 文件中包含了 25×25 条流量信息。但是本文需要的数据集格式以不同的流划分文件，每个文件包含了两周时间内一条流以 15 min 为间隔的全部数据，文件输出格式为常用数据集格式 CSV。创建一个数组 array=[]，设置文件的 path。读取目录下的文件名称，存储在变量 files 内。对 files 循环遍历该目录下的所有文件，每次循环调用 dom.minidom.parse 打开 XML 文件，打开文件之后通过 rootdata.get Elements By Tag Name 设置读取范围在标签<dst>…</dst>之间的元素。标签内的即需要的流量数据。通过 itemlist.firstChild.data[5]确定选择每个 XML 文件中的第几条数据，并将读取出来的数据保存到数组中。使用 to_csv 函数将数组 array=[]输出 CSV 文件保存至相同目录下。将数据拆分成训练集和测试集，将第 n_0～n_12 个数字作为特征。第 n_13 个数字为标签。依次循环累加 x 将第 n_x 个数字至第 $n_(x+12)$个数字作为特征。第 $n\_(x+13)$ 个数字作为标签，最后，返回 x_train 训练集的特征，y_train 测试集的标签，x_test 测试集的特征，y_test 测试集的标签。最后再将数据集进行标准化。

1.3　BL-LSTM 模型搭建

人们对一个问题的思考不会从头开始，如一个人能够理解一篇文章是因为这个人已经有了一定的语言基础，人能够不断地累积知识，是因为

人能够记忆一些已经学习过的知识。但是传统的神经网络做不到这一点，传统的神经网络对时序信息的预测和处理是非常困难的。如预测一句话的结尾，或者文章内容的下一段。虽然理论上递归神经网络（recurrent neural network，RNN）可以处理这种“长依赖问题”，但在实际过程中 RNN 无法学习这种特征。Hochreiter[3]和 Bengio[4]深入研究过为什么 RNN 不能学习到该特征。于是长短期记忆（long short-term memory，LSTM）网络——一种新的 RNN 模型应运而生。由于 RNN 绝大分都由相同的神经网络复制链接而成，而神经网络的结构非常简单，LSTM 也具有这种链式结构，但是它不同于简单的 RNN 神经网络单元里只有一个网络层，LSTM 的内部有 4 个网络层。LSTM 的核心是网络单元设计，LSTM 的网络单元设计类似一个细胞，一个网络单元内的状态传输结构简单来说是由一个点乘器和一个加法器构成的，数据的传输过程像一个传送带，贯穿了整个细胞而且鲜有分支。一个 LSTM 里有 3 个门，而 LSTM 的细胞状态是由这 3 个门控制的。门是由一个 sigmoid 函数和一个点乘操作组合而成的。门的作用是为了对细胞状态进行删除和更改，sigmoid 函数输出的值在 0～1，这表示有多少数据可以通过 sigmoid 层，0 表示全丢弃，1 表示全接受。LSTM 细胞状态如图 2 所示。

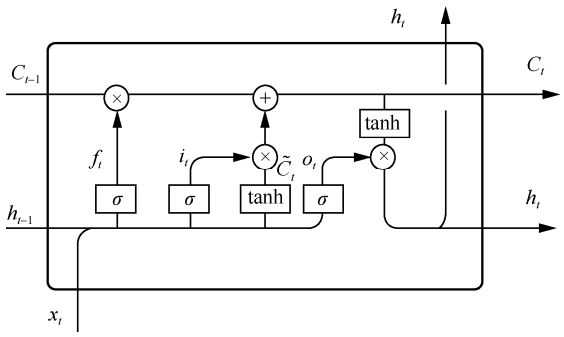


图 2　LSTM 状态传输

本文基于 LSTM 模型[5]修改的深度学习模型 BL-LSTM（based linear regression-LSTM），BL-LSTM 算法模型如图 3 所示。采用 BL-LSTM 进行 SDN 流量预测模型的搭建。具体来说由两个部分组成，一个是 LSTM 网络，另一个是线性回归模型。LSTM 网络由 m 个堆叠的 LSTM 层组成，用于对网络流量的时空特征进行建模。输出层采用线性回归模型计算未来的流量大小。

开始训练模型时，不会直接把整个数据集导入模型中，而是使用 batch 的方式。深度学习的优化算法总的来说是梯度下降法，每次的权重和偏置更新可以分为以下两种极端。第一种是遍历完所有数据后再计算损失函数，然后根据新的参数计算新的梯度。这种方法因为数据集的增长和内存的限制，一次性导入整个数据集，变得非常困难。另一种是每次只训练一个样本，每载入一个数据就计算一次损失函数，然后求出梯度更新参数，这种方法虽然速度较快，但是收敛性能较差，可能在最优点附近来回移动，无法到达最优点。或者几次参数的更新互相抵消最终导致目标函数的强烈震荡，为了克服上述的两种极端现象，一般采取折中的方法。

训练模型时将数据分批次载入，这样一批数据共同决定了梯度方向，方向不容易跑偏，降低了随机性。另一方面因为一批内的样本数与整个数据集相比小很多，计算量也大大减小。定义迭代变量 iterations，一个 iterations 等于使用 batch_size 个样本训练一次。定义轮询变量 epochs，epochs 被定义为向前和向后传播所有批次的单次训练迭代，这意味着一个 epoch 是整个输入数据的单次向前和向后传递。简单来说，epoch 指的是训练过程中数据将要被轮询多少次，即需要完整遍历数据集多少次。如果 epoch 数量太少，神经网络就有可能发生欠拟合（即对定型数据的学习不够充分），如果 epoch 数量太多，则有可能发生过拟合（即神经网络对定型数据中的“噪声”而非信号进行拟合）。BL-LSTM 模型搭建的流程如下。

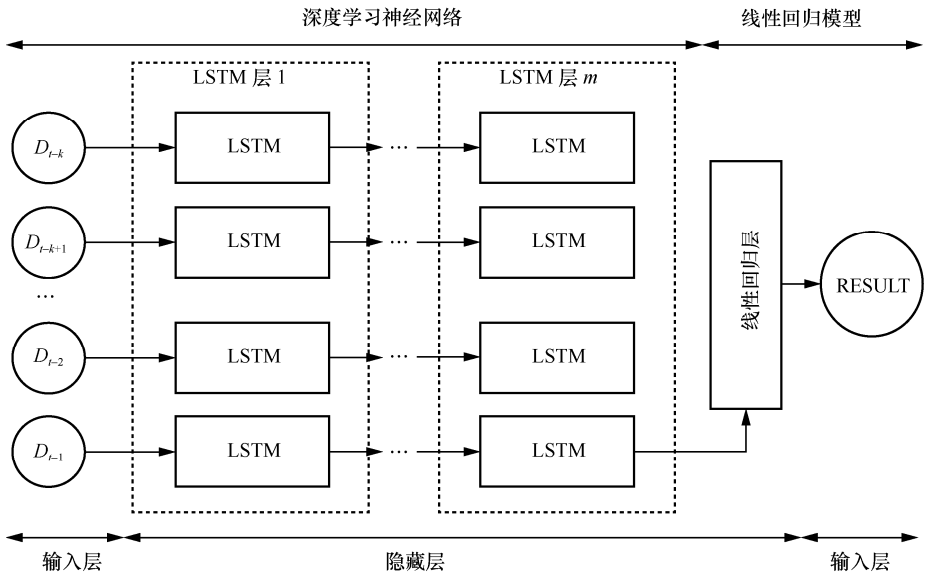


图 3　BL-LSTM 算法模型

步骤 1　创建一个 ArgumentParser 对象，添加参数并解析，设置一组特征数量 lag 为 12。

步骤 2　经过多次调试最终设置 batch 值为 256，设置 epochs 值为 600。调用数据处理函数获取 *x*_train 训练集的特征和 *y*_train 训练集的标签。

步骤 3　对 *x*_train 进行 reshape，将数据转变为一列的三维数组。设置输入层、隐藏层和输出层的参数。

步骤 4　初始化一个顺序模型，加入一个 LSTM 层，设置 return_sequences=True，每个时间步都输出 h_{t-1}。

步骤 5　将 LSTM 的输出结果，输出到多元线性回归模型进行训练。

步骤 6　获得完整的模型，输入测试集 *x*_test，取得预测结果。

步骤 7　将归一化后的结果转化为原始数据。

2　基于流量预测的路由规划和切换

传统路径切换主要为如何在限制条件完成最快的更新，如何解决网络变化时流量波动等问题。

2.1　SDN 路径切换解决方案

针对多条路径更新时，因为链路的带宽限制使路径切换后出现网络拥塞的问题[9]，本文使用 BP-ACO 算法[10]进行路由规划，GEANT 获得的流量矩阵作为数据集通过 BL-LSTM 算法进行训练，预测后短时间内的流量趋势。BP-ACO 通过对流量预测的分析进行路由规划避免流量拥塞，对这个问题本文通过上述基于 LSTM 模型的流量预测的方法进行处理。

针对域间路由更新导致流表部署时间不同的问题，以及网络变化时会产生控制器下发的流表和交换机内实际接收到流表的不一致问题。

本文提出了一种新的方法解决 SDN 的一致性多交换机更新问题，方法是通过协调不同控制域内的控制器，中央控制器向各控制域下发流表，实现控制域的有序更新。但是域内的控制器向不同高质量的，多层虚拟交换机（open vSwitch，OVS）下发流表且 OVS 部署流表会产生不同的时延，导致转发数据流时没有流表而开始丢包。因为新的域内路由有空闲 OVS，传输时延较低，传输过程中因为有一些数据包在链路传输，所以数据包传输失序。虽然失序现象可以通过传输控制协议（transmission control protocol，TCP）的重传机制缓解，但是随着数据发送速率的不断上升，

传输的数据包开始变大，随机失序情况会越来越明显。

为了解决域内的一致性问题，本文提出了一种新的方法，该方法通过在拆除现有路由之前检查新路由的运行状态。具体的识别过程是通过多播的方式向每一个既在新路径又在旧路径上的 OVS CN_i 发送两条相同的转发流表。CN_i 收到相同的数据包之后确认当前网络的新旧两条链路都能正常转发数据包，再断开旧的链路。然后，新的路径开始正式使用，最后将新的路径剩余带宽、吞吐量提交给控制器，控制器重新进行路由规划。

2.2　无损路径切换算法

一个 OVS 连接 SDN 控制器时，触发 Event OFPS witch Features 事件，在事件内实例化一个连接 OVS 到控制器的网桥，实例化并解析 OpenFlow 协议，创建一个匹配流表头部的 Match 域，控制器向 OVS 发送报文，OVS 接收报文后通过默认流表向控制器发送确认字符（acknowledge character，ACK），控制器通过收到的反馈报文，确认 OVS 和控制器建立连接。

SDN 中央控制向所有只在新链路 (L_{new}) 的控制器 $\left(C_{\text{new}} \cap \bar{C}_{\cap}\right)$ 发送流表，OVS 接收 SDN 控制器发送的流表后开始安装收到的流表，OVS 安装完流表后发送 ACK 返回给控制器，中央控制器接收到所有 OVS 的 ACK 后开始下一步。

SDN 中央控制向所有既在新链路 (L_{new}) 又在旧链路 (L_{old}) 的控制器 $\left(C_{\cap}\right)$ 发送流表，OVS 接收 SDN 控制器发送的流表后开始安装接收的流表，OVS 安装完流表后发送 ACK 确认报文返回给控制器，中央控制器接收所有 OVS 的 ACK 后开始下一步。

SDN 控制器对所有域内链路，从远到近发送多播信号，当 $R_{\cap}$ 收到两个相同的数据包时，确认两条链路都可以正常工作，则断开 L_{old} 链路中属于本次转发链路的部分。路径切换流程如图 4 所示。

2.3　基于多播的无损路径更新技术

在 SDN 中，控制器可以删除、添加或替换路由表中的源目流表。采取以多播的方式更新路径，旨在最小化节点资源消耗和链路资源消耗总成本。本文通过使控制器向 OVS 下发相同的源目流表。然后数据包通过先进先出队列（first input first output，FIFO）的方式沿着某一条路径传输。如果两个数据包 P_1、P_2 沿着同一条路径传输，然后都经过了 OVS 交换机 S_1、S_2。如果 P_1 在 P_2 之前遍历 S_1，那么 P_1 必须在 P_2 之前遍历 S_2。当 $S_i \neq S_d$ 收到了流表之后，部署安装流表，经过该 OVS 的数据包按照新的路由表开始转发数据包。

SDN 控制器可以指示 OVS S_i 向控制器转发一段路径中的所有数据包，通过对这段路径中的 S_i 打上标签，在 S_i 这些的路由表中添加一条转发给控制器的流表。通过控制器识别数据包是否相同，而不是通过交换机 S_i 识别。值得注意的是，只有整条转发链路的最后一个 OVS 需要将数据包转发给控制器，而不是转发链路中的每一个 OVS。因此，OVS 的标准操作和 OpenFlow 命令可以支持多播转发所需要的操作。多播传输中复制的数据包通过每个数据包的源地址和标识字段辨别。通过这个方法可以分辨当前数据包是多播传输中复制的数据包还是通过 TCP 重传机制重新发送到 S_i 的数据包。

本文介绍算法的一个例子，SSRU 切换开始之前的网络拓扑如图 5 所示，其中，一次只建立 (s,d) 路径的一个部分，并替代当前的等效部分。D_x 表示一个域，$C_x, x \in \{0,1,2,\cdots,n\}$ 表示旧链路上的 OVS，$N_y, y \in \{0,1,2,3,\cdots,n\}$ 表示新链路上的 OVS，

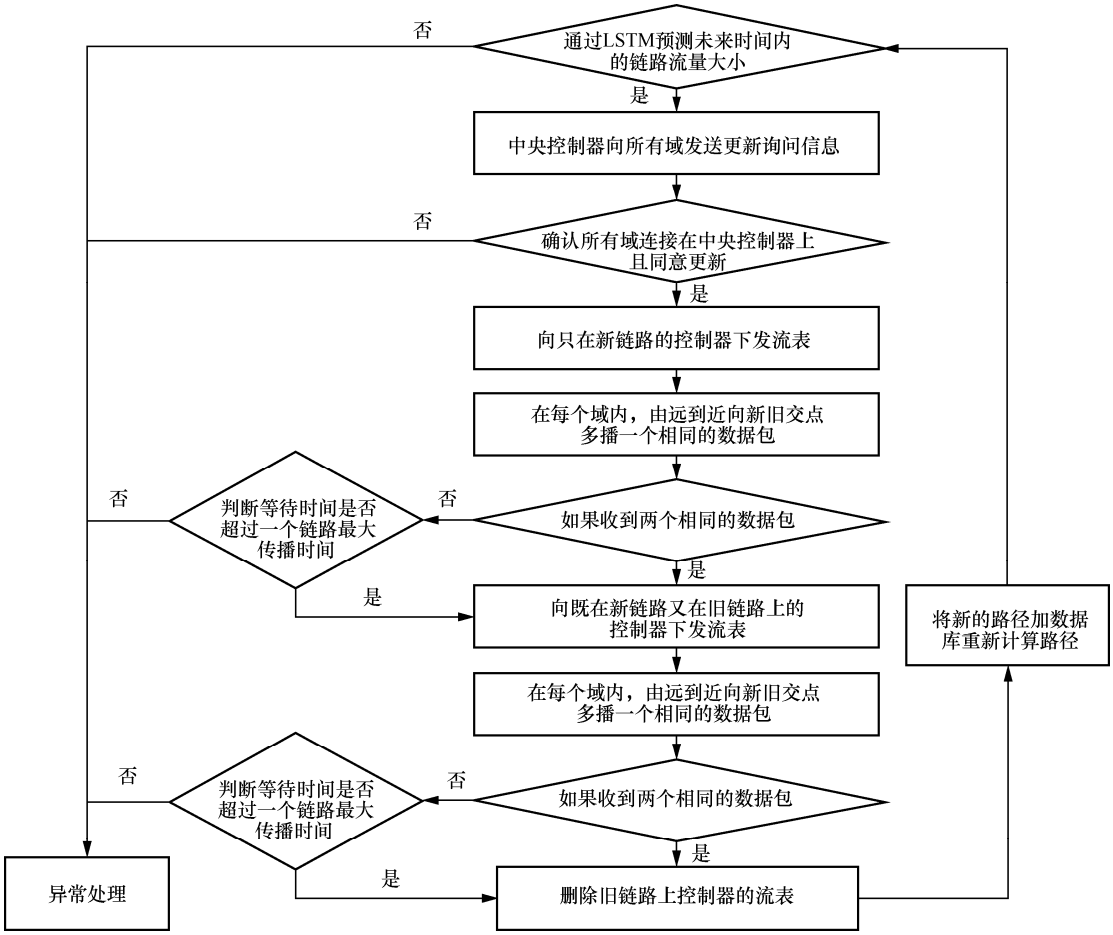


图 4　路径切换流程

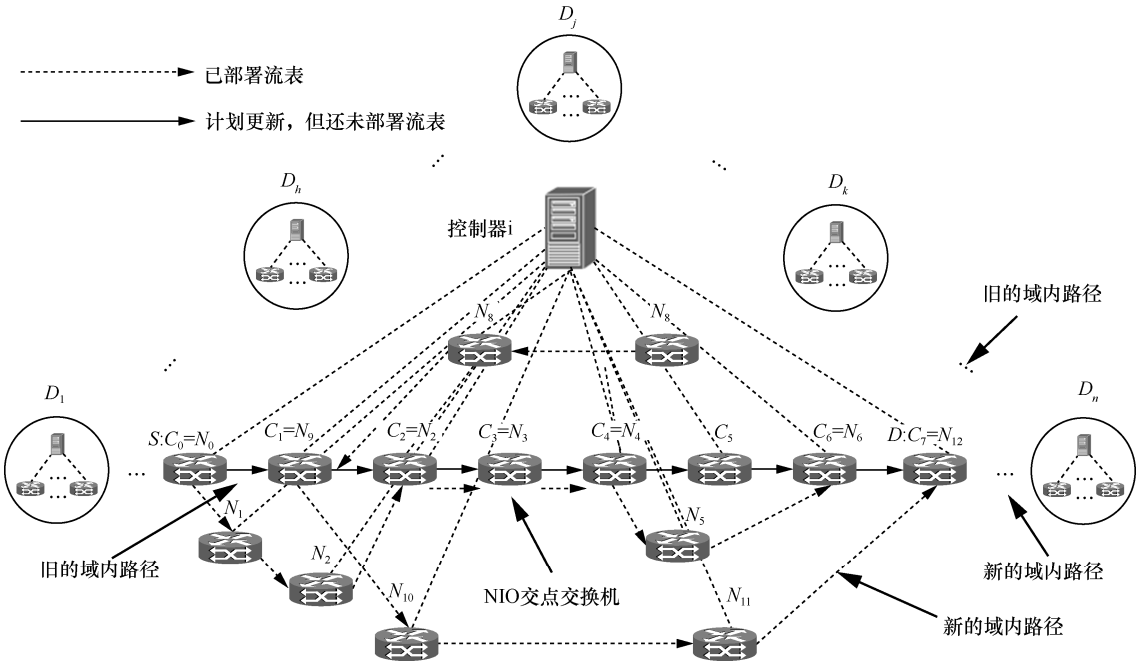


图 5　SSRU 切换开始之前的网络拓扑

$\mathrm{CN}_i : C_x = N_y$，$x \in \{0,1,2,\cdots,n\}$ 表示既在新链路又在旧链路上的 OVS。

控制器查询新旧两条域间路径上 OVS 的 dpid，如果找到两组相同的 dpid 就可以确认该域处于新旧路径的交点。首先，向只在新域间路径的控制器下发流表，然后再部署域内 OVS 的流表。

切换从 CN_i 下标最大的替换路径开始，第一段需要切换的路径 N_9～N_{12}。控制器向 N_9～N_{12} 新链路上的 OVS 下发流表。

然后，起点 S 交换机向终点 D 交换机发送两条相同的数据包，一条沿着 $C_x, x \in \{0,1,2,\cdots,n\}$ 传输，一条沿着 $N_y, y \in \{0,1,2,\cdots,n\}$ 传输。当相交节点 N_{12} 收到两个相同的数据包时确认两条路径都可以正常转发，删除旧链路上的流表，停止旧链路的转发。

通过算法计算第二段切换的路径为 N_0～N_9，因此，向 N_0～N_9 新链路上的节点下发流表。

起点 S 交换机向终点 D 交换机通过多播发送两条相同的数据包，当相交节点 N_9 收到两个相同的数据包，确认两条转发链路建立完成。删除旧链路的转发流表，流量按照新链路开始转发，完成切换，SSRU 切换结束之后的网络拓扑如图 6 所示。

中央控制器向源目管理域内的控制器下发流表，然后域内通过上文域内的多播算法对域内路径进行切换，完成新域间路径的切换之后，删除旧域间路径上的转发流表。

3　基于 RYU 的服务链功能部署开发与测试

本实验使用如下硬件设备：华为 FusionCloud 服务器 5 台、华为 CE6800 交换机两台、华为 CE12800 交换机两台、华为 S1700 交换机一台、AR 交换机一台。本实验使用如下软件：4.18 版本的 Ryu 控制器、2.3.0 版本的 mininet、16.04 版本的 Ubuntu。本节主要介绍了 BL-LSTM 算法的流量预测和在预测结果下基于 BP-ACO[10]的路由规划和切换实验。

3.1　BL-LSTM 算法的流量预测实验

本文以 GEANT2 部署的 25 个节点之间的流量信息作为数据集，然后通过流量数据测量和估算方法计算流量矩阵，再通过流量矩阵进行流量预测。通过 BL-LSTM 实现流量的预测，并将该算法和门控循环单元（gated recurrent unit，GRU）[5]

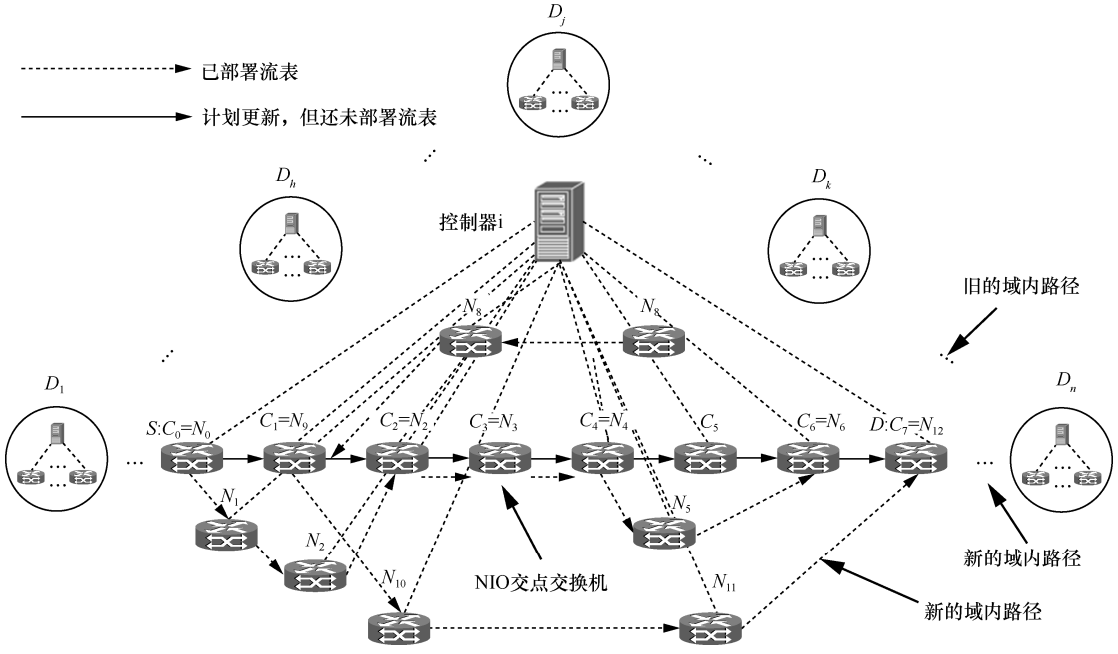


图 6　SSRU 切换结束之后的网络拓扑

以及栈式自编码（stacked auto-encoder，SAE）深度学习模型[8]进行比较。选择第 12 个端口到第 23 个端口的链路作为被测对象，数据集每隔 5 min 进行一次测量，持续两周。该链路共记录 977 条流量数据。预处理之后的数据集见表 1。

表 1　预处理之后的数据集

统计编号	线路 1 流通量	统计编号	线路 1 流通量
1	67 746.500 0	11	60 187.555 6
2	66 788.506 7	12	53 070.968 9
3	59 951.386 7	13	53 844.088 9
4	147 980.524 4	14	58 859.048 9
5	142 226.871 1	15	121 630.168 9
6	170 772.764 4	16	130 752.986 7
7	174 522.826 7	17	134 375.217 8
8	115 441.662 2	18	143 795.431 1
9	107 204.693 3	19	155 778.044 4
10	92 442.328 9	20	166 465.457 8

首先读取 977 个 XML 流量数据文件，其次将每份数据集中的 12～23 端口之间的流量依次插入到新的 newdata.CSV 文件。

然后通过预处理脚本对获取的数据进行分组、归一化、矩阵转换等数据预处理，脚本处理算法见算法 1。

算法 1　预处理脚本

开始

train, test = [], []

for *i* in range(lags, len(flow1)):

train.append(flow1[*i* − lags: *i* + 1])

for *i* in range(lags, len(flow2)):

test.append(flow2[*i* − lags: *i* + 1])

train = np.array(train)

test = np.array(test)

np.random.shuffle(train)

*X*_train = train[:, :−1]

*y*_train = train[:, −1]

*X*_test = test[:, :-1]

*y*_test = test[:, -1]

return *X*_train, *y*_train, *X*_test, *y*_test, scaler

结束

本节实验将本文提出的 BL-LSTM 算法与 GRU 算法、SAE 算法进行对比，首先将 BL-LSTM 算法的预测结果与真实数据进行对比，BL-LSTM 预测结果曲线如图 7 所示，BL-LSTM 曲线很好地拟合了真实值的曲线，但低点较真实值偏高。将归一化的测试集和预测结果进行反归一化，纵坐标变为流量的真实大小。

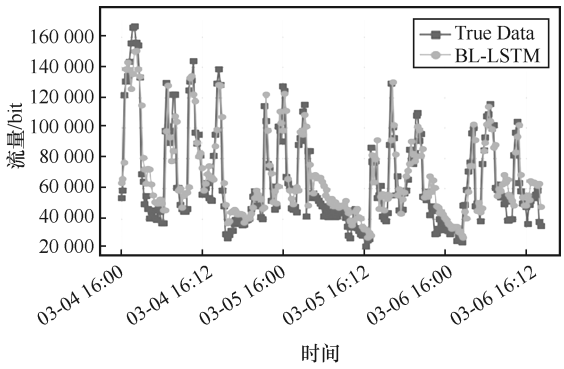


图 7　BL-LSTM 预测结果曲线

其次，将 3 种算法的预测结果和真实值进行对比，预测曲线对比如图 8 所示。BL-LSTM 算法对真实值曲线的拟合效果最好，SAE 的效果次之，GRU 的效果最差。

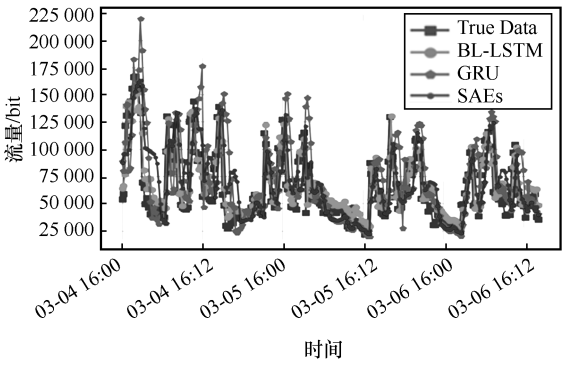


图 8　预测曲线对比

使用归一化过后的测试集和预测结果计算相应的预测评估结果，量化算法之间的区别。其中，以 12～23 端口之间的流量数据集为例，对测试集的评价结果见表 2。

表 2　评价结果

姓名	MAPE	MAE	MSE	RMSE	R2
BL-LSTM	22.361 037%	0.044 330	0.003 890	0.062 366	0.679 550
GRU	43.088 212%	0.088 303	0.015 412	0.124 145	0.26 975
SAE	35.915 936%	0.073 148	0.009 76	0.098 79	0.195 930

由表 2 可知 BL-LSTM 模型的 MAPE 为 22.361 037%，误差最小；SAE 次之，GRU 的 MAPE 为 43.088 212%，误差最大。从 BL-LSTM、SAE、GRU 模型的预测结果看，BL-LSTM 模型的预测准确率要高于 SAE 和 GRU。虽然 LSTM 和 GRU 都是循环神经网络，但是在处理时间序列问题上，具有长期记忆特性的循环神经网络 LSTM 优势更大。

25 组数据中不同算法的 MAPE 值如图 9 所示，采用 25 组不同端口之间的数据作为数据集，各种算法预测结果的 MAPE 值区别明显。BL-LSTM 预测精度最高，MAPE 值在 22.5%左右，SAE 和 GRU 略有波动，SAE 的 MAPE 平均值为 35.2%，GRU 的平均值为 45.8%。

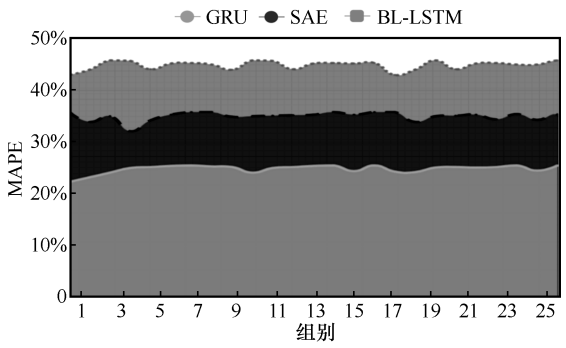


图 9　25 组数据中不同算法的 MAPE 值

3.2　切换实验

运行整理好的 mininet 拓扑，拓扑主要由 4 个控制域构成，每个控制器控制一个控制域，SSRU 切换之前的转发路径如图 10 所示。*C*2 的控制域只在旧的链路上，*C*1 的控制域只在新的链路上。而 *C*0、*C*3 的控制域既在新的链路又在旧的链路，域间的路由切换需要中央控制器对各控制域有序下发流表实现。

每个域内有 6 台 OVS，它们构成两条新旧链路，以 *C*0 的控制域为例，*S*22～*S*24 只在旧链路上的交换机、*S*18～*S*S20 是只在新链路上的交换机、*S*17 和 *S*21 既在新链路又在旧链路上的交换机。域内的路径切换需要通过多播数据包确认路由更新状态，进而实现路径切换，SSRU 切换之后的转发路径如图 11 所示。

运行写好的 RYU 控制器和 mininet 拓扑，控制器连接 OVS 会触发装饰器 Event OFPS witch Features，每次 OVS 向控制器泛洪报文的时候将定义的标志（Flags）加 1。控制器连接 OVS 如图 12 所示。

经过测试发现 55 次左右的装饰器触发能够建立完整的连接。本文设置 Flags 到达 55 之后控制器开始执行路径切换。

传统的 Ryu 二层交换机作为控制器，发送 37 个数据包进行测试，75%的数据包在切换的过程中丢失。二层交换交换机丢包现象如图 13 所示。

在应用无损路径切换算法时，发送多组数据包进行测试，没有数据包在切换的过程中丢失，该算法的丢包率和失序率都明显优于传统路径切换算法。SSRU 切换丢包现象如图 14 所示。

发送速率从 100 Mbit/s 到 500 Mbit/s 的丢包数量实验结果见表 3，当发送速率从 100 Mbit/s 到 500 Mbit/s 时，丢包率始终维持在 0，但是其中伴随了一些包失序问题，这是因为在路径切换过程中，新路径的网络设备会出现空闲状态，该路径上的传输速率更快，于是切换过程中后续的数据包经过新链路可能会先到达目的地址，导致在一

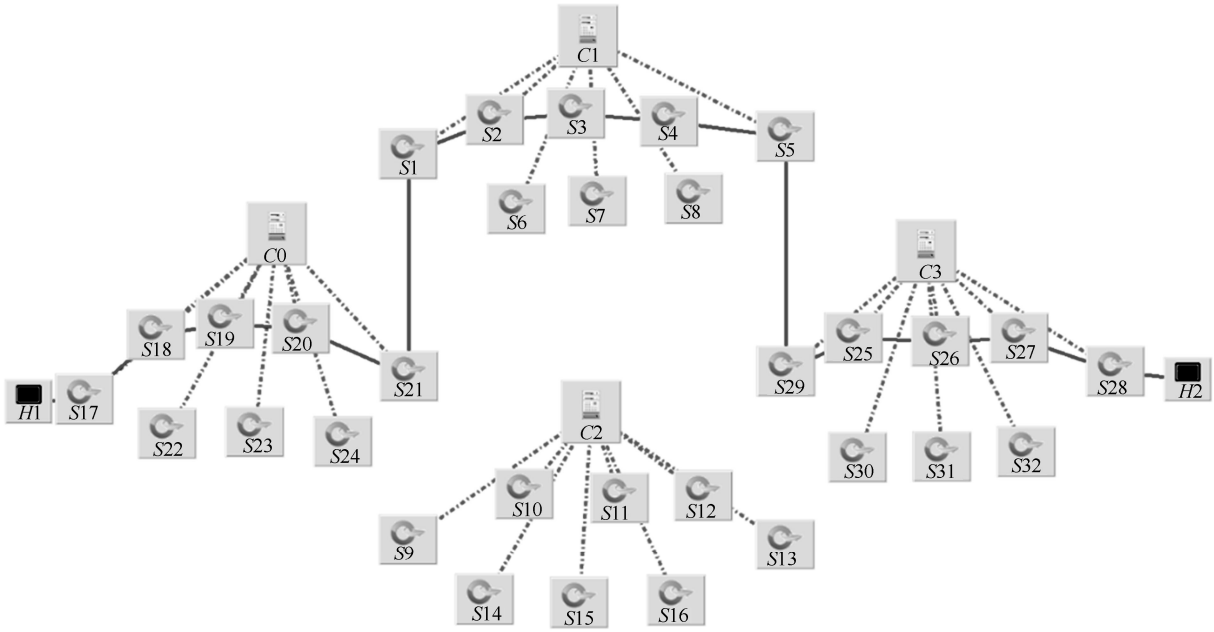


图 10　SSRU 切换之前的转发路径

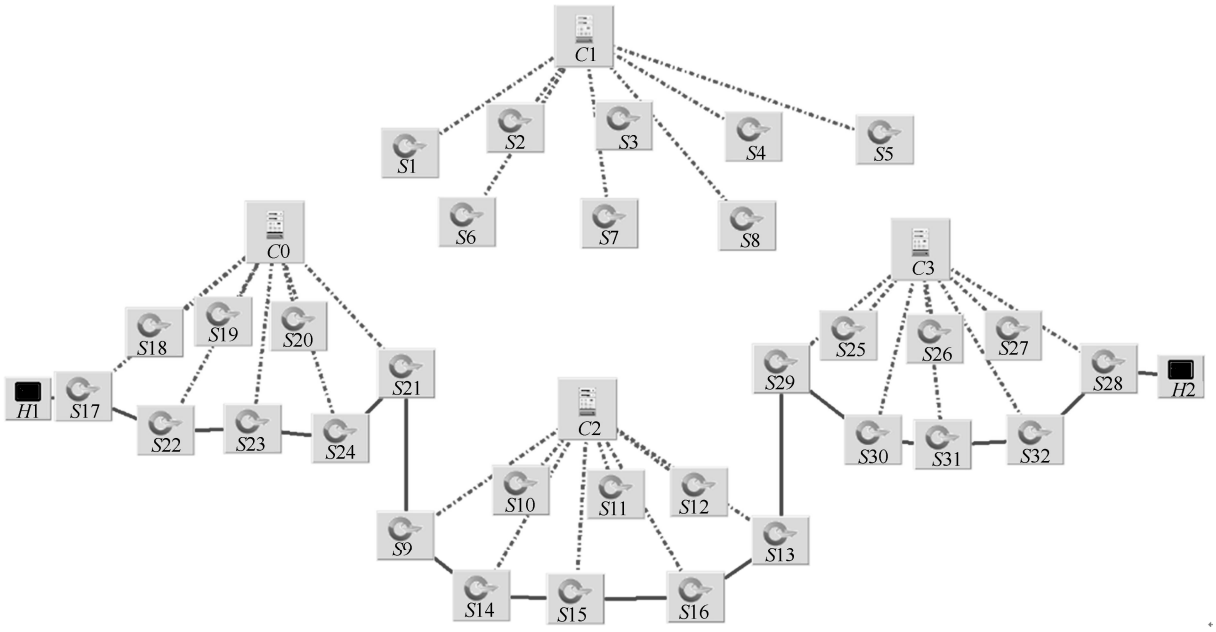


图 11　SSRU 切换之后的转发路径

个往返时延（round-trip time，RTT）内可能会出现数据包失序的情况。幸运的是 TCP 具有重传机制可以有效地缓解这个问题，用户数据报协议（user datagram protocol，UDP）也有相关的机制解决少量的数据包失序。

从 H1 到 H2 跨越 3 个控制域发送数据包，发送速率从 800 Mbit/s 到 1 200 Mbit/s 的丢包数量实验结果见表 4。

发送速率逐步提高的实验结果见表 5，可以看出来 SSRU 算法在进行路径切换时引起的平均时延非常少，且这种时延几乎不影响文件的实际传输，在文件的传输过程中既保证了业务

```
root@shaoyu-virtual-machine: /home/shaoyu/green_dragon
1
00:00:00:00:00:04
OFPMatch(oxm_fields={'eth_dst': '33:33:00:00:00:02', 'in_port': 2})
others_flag33
6
1
00:00:00:00:00:04
OFPMatch(oxm_fields={'eth_dst': '33:33:00:00:00:02', 'in_port': 2})
5_flag34
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->LinksNomissChange EventOFPPacketIn
```

图 12　控制器连接 OVS

```
*** Starting CLI:
mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
From 10.0.0.1 icmp_seq=7 Destination Host Unreachable
From 10.0.0.1 icmp_seq=8 Destination Host Unreachable
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
From 10.0.0.1 icmp_seq=15 Destination Host Unreachable
From 10.0.0.1 icmp_seq=16 Destination Host Unreachable
From 10.0.0.1 icmp_seq=17 Destination Host Unreachable
From 10.0.0.1 icmp_seq=18 Destination Host Unreachable
From 10.0.0.1 icmp_seq=19 Destination Host Unreachable
64 bytes from 10.0.0.4: icmp_seq=29 ttl=64 time=1.50 ms
64 bytes from 10.0.0.4: icmp_seq=30 ttl=64 time=0.068 ms
64 bytes from 10.0.0.4: icmp_seq=31 ttl=64 time=0.065 ms
64 bytes from 10.0.0.4: icmp_seq=32 ttl=64 time=0.078 ms
64 bytes from 10.0.0.4: icmp_seq=33 ttl=64 time=0.063 ms
64 bytes from 10.0.0.4: icmp_seq=34 ttl=64 time=0.065 ms
64 bytes from 10.0.0.4: icmp_seq=35 ttl=64 time=0.071 ms
64 bytes from 10.0.0.4: icmp_seq=36 ttl=64 time=0.064 ms
64 bytes from 10.0.0.4: icmp_seq=37 ttl=64 time=0.063 ms
^C
--- 10.0.0.4 ping statistics ---
37 packets transmitted, 9 received, +19 errors, 75% packet loss, time 36823ms
rtt min/avg/max/mdev = 0.063/0.226/1.501/0.450 ms, pipe 4
mininet>
```

图 13　二层交换交换机丢包现象

```
mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.087 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.062 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.107 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.058 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.085 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.057 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.153 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=0.071 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=0.063 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=0.086 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=0.063 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=0.067 ms
64 bytes from 10.0.0.4: icmp_seq=14 ttl=64 time=0.114 ms
64 bytes from 10.0.0.4: icmp_seq=15 ttl=64 time=0.060 ms
64 bytes from 10.0.0.4: icmp_seq=16 ttl=64 time=0.056 ms
64 bytes from 10.0.0.4: icmp_seq=17 ttl=64 time=0.060 ms
64 bytes from 10.0.0.4: icmp_seq=18 ttl=64 time=0.060 ms
64 bytes from 10.0.0.4: icmp_seq=19 ttl=64 time=0.071 ms
64 bytes from 10.0.0.4: icmp_seq=20 ttl=64 time=0.061 ms
64 bytes from 10.0.0.4: icmp_seq=21 ttl=64 time=0.055 ms
64 bytes from 10.0.0.4: icmp_seq=22 ttl=64 time=0.052 ms
64 bytes from 10.0.0.4: icmp_seq=23 ttl=64 time=0.061 ms
64 bytes from 10.0.0.4: icmp_seq=24 ttl=64 time=0.067 ms
64 bytes from 10.0.0.4: icmp_seq=25 ttl=64 time=0.091 ms
64 bytes from 10.0.0.4: icmp_seq=26 ttl=64 time=0.094 ms
64 bytes from 10.0.0.4: icmp_seq=27 ttl=64 time=0.056 ms
64 bytes from 10.0.0.4: icmp_seq=28 ttl=64 time=0.058 ms
64 bytes from 10.0.0.4: icmp_seq=29 ttl=64 time=0.055 ms
^C
--- 10.0.0.4 ping statistics ---
29 packets transmitted, 29 received, 0% packet loss, time 28662ms
rtt min/avg/max/mdev = 0.052/0.071/0.153/0.025 ms
mininet>
```

图 14　SSRU 切换丢包现象

表 3　发送速率从 100 Mbit/s 到 500 Mbit/s 的丢包数量实验结果

评价指数	发送速率				
	100 Mbit/s	200 Mbit/s	300 Mbit/s	400 Mbit/s	500 Mbit/s
收到的数据包	8 550	18 542	25 825	34 882	45 852
丢失的数据包	0	0	0	0	0
丢包率	0%	0%	0%	0%	0%
数据包记录	0	1	5	14	24
重新排序率	0%	0.005 39%	0.019%	0.040%	0.052%

表 4　发送速率从 800 Mbit/s 到 1 200 Mbit/s 的丢包数量实验结果

评价指数	发送速率				
	800 Mbit/s	900 Mbit/s	1 000 Mbit/s	1 100 Mbit/s	1 200 Mbit/s
收到的数据包	64 589	75 628	85 421	88 642	108 541
丢失的数据包	0	0	0	0	0
丢包率	0%	0%	0%	0%	0%
数据包记录	25	24	22	41	45
重新排序率	0.039%	0.032%	0.026%	0.046%	0.041%

表 5　发送速率逐步提高的实验结果

SR	ATS						
	0.525 Gbit/s	0.685 Gbit/s	0.728 Gbit/s	0.846 Gbit/s	0.985 Gbit/s	1.258 Gbit/s	1.532 Gbit/s
NR	15	14	14	14	15	14	15
TTR/s	70.87	72.51	70.87	70.53	71.87	72.17	71.10
NTT/s	70.85	72.41	70.73	70.28	71.52	71.76	70.58
TDIR/s	0.02	0.10	0.14	0.25	0.35	0.41	0.52
ADIER/s	0.001 3	0.007 1	0.010	0.018	0.023	0.029	0.034

以及会话的稳定运行，又防止流量只走最短路径导致网络局部拥塞，对网络整体进行了负载均衡。

对表 5 出现的缩写的简单介绍见表 6。

表 6　缩写介绍

缩写	全称
ATS	average transmission rate
SR	simulation result
NR	number of reconfiguration
NTT	normal transmission time
TTR	transmission time with reconfiguration
TDIR	total delay increased by reconfiguration
ADIER	average delay increased by each reconfiguration

传输速率对平均时延的影响如图 15 所示，可以看出随着传输速率 ATR 的增长，路径切换产生的平均时延呈线性增长，这表示 SSRU 算法在高速传输中依然可以保持高可用性。验证了即使在高的传输速率 SSRU 也可以实现无损的路径切换。

4　结束语

随着近年网络技术的不断发展和演进，用户对网络资源的需求体量越来越大，内容越来越复杂，传统的网络架构很难满足目前灵活的网络需求。SDN 的革命性技术思想赋予了网络可编程性和可演进性，实现网络资源更灵活的管理。为了提高网络资源利用率，提出了基于 LSTM 模型的流量预测，为路由规划提供理论依据。通过提前

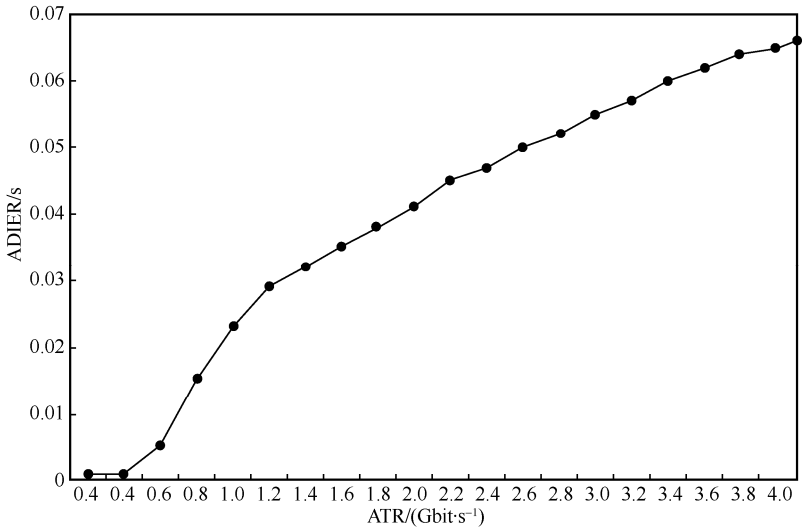


图 15　传输速率对平均时延的影响

预测网络流量，在流量激增之前进行防御措施保证网络的安全性和稳定性，实现基于流量预测的路由规划、无损路径切换等功能。

参考文献：

[1] 张朝昆, 崔勇, 唐骞骞, 等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1): 62-81.
ZHANG C K, CUI Y, ANGH H. State-of-the-art survey on software-defined networking(SDN)[J]. Journal of Software, 2015, 26(1): 62-81.

[2] UHLIG S, QUOITIN B, LEPROPRE J, et al. Providing public intradomain traffic matrices to the research community[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(1): 83-86.

[3] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[4] BENGIO Y, COURVILLE A, VINCENT P. Representation learning: a review and new perspectives[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(8): 1798-1828.

[5] 贺小伟, 徐靖杰, 王宾, 等. 基于 GRU-LSTM 组合模型的云计算资源负载预测研究[J]. 计算机工程, 2022, 48(5): 11-17, 34.
HE X W, XU J J, WANG B, et al. Research on cloud computing resource load forecasting based on GRU-LSTM combination model[J]. Computer Engineering, 2022, 48(5): 11-17, 34.

[6] 杜秀丽, 陶帆, 范志宇, 等. 面向骨干网流量的非线性组合预测方法[J].小型微型计算机系统, 2021(4): 1-7.
DU X L, TAO F, FAN Z Y, et al. Nonlinear Combined Prediction Method for Backbone Network Traffic[J]. Small Microcomputer System, 2021(4): 1-7.

[7] 刘振鹏, 张庆文, 李明, 等. 基于蚁群优化算法的 SDN 路由策略[J]. 昆明理工大学学报(自然科学版), 2022, 47(3): 60-66.
LIU Z P, ZHANG Q W, LI M, et al. SDN routing strategy based on ant colony optimization algorithm[J]. Journal of Kunming University of science and Technology (Natural Science edition), 2022, 47(3): 60-66.

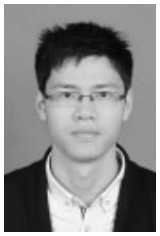
[8] 宋永辉. 认知无线网络中基于人工智能算法的流量预测和数据传输机制研究[D]. 重庆: 重庆邮电大学, 2017.
SONG Y H. Research on mechanism of traffic prediction and data transmission based on artificial intelligence algorithm in cognitive wireless network[D]. Chongqing: Chongqing University of Posts and Telecommunications, 2017.

[9] 汪尧, 黄宁, 武润升, 等. 基于改进自回归差分移动平均模型的网络流量预测[J]. 通信技术, 2021, 54(12): 2626-2631.
WANG Y, HUANG N, WU R S, et al. An improved autoregressive integrated moving average model for network traffic prediction[J]. Communications Technology, 2021, 54(12): 2626-2631.

[10] 刘庆杰, 王小英, 王茂发. 基于 BP 算法和 ACO 算法的故障诊断推理研究[J]. 计算机测量与控制, 2012, 20(6): 1460-1462, 1466.
LIU Q J, WANG X Y, WANG M F. Research on reasoning for fault diagnosis based on BP and ACO algorisms[J]. Computer Measurement & Control, 2012, 20(6): 1460-1462, 1466.

[作者简介]

诸葛斌（1976– ），男，博士，浙江工商大学信息与电子工程学院教授，浙江省新世纪151 人才工程第三层次培养人员，主要研究方向为网络与通信技术、互联网技术和网络安全。

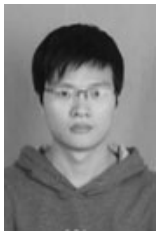


邵瑜（1994– ），男，浙江工商大学硕士生，主要研究方向为软件定义网络。

王林超（1996– ），男，浙江工商大学硕士生，主要研究方向为软件定义网络。

董黎刚（1973– ），男，博士，浙江工商大学信息与电子工程学院党委书记、教授、硕士生导师，主要研究方向为新一代网络和分布式系统。

宋杨（1996– ），男，浙江工商大学硕士生，主要研究方向为软件定义网络。



蒋献（1988－ ），男，浙江工商大学实验师，主要从事数字电路实验和模拟电路实验工作。