基于SDN-SFC的服务功能负载均衡

张 钊 李海龙 胡 磊 董思岐
(火箭军工程大学 西安 710025)

摘 要 随着互联网技术的飞速发展,网络终端设备趋向于小型化、方便化,而随着移动终端的普及,其被使用的频率越来越高,人们对网络带宽的需求与日俱增,同时对网络数据传输时间的要求也愈加苛刻。为满足这一需求,文中提出了基于SDN-SFC的负载均衡机制。首先,其对各个终端所需服务的类型和优先级进行分类;然后,采用启发式算法来规划SFC之间的传输路径,以减少每个SF的负载,从而提高整体网络性能。仿真结果表明,提出的方法可以缩短数据传输的时间,实现负载均衡。

关键词 软件定义网络,负载均衡,数据带宽,服务功能链
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.09.018

Service Function Load Balancing Based on SDN-SFC

ZHANG Zhao LI Hai-long HU Lei DONG Si-qi
(Rocket Force University of Engineering, Xi'an 710025, China)

Abstract With the rapid development of Internet technology, network terminal devices tend to be smaller and more convenient, and with the popularity of mobile terminals, their frequency of utilization is higher, people's demand for network bandwidth is increasing sharply, at the same time, people's requirement for network data transmission time is becoming more and more stringent. To meet this demand, this paper proposed a load balancing mechanism based on SDN-SFC. It considers and categorizes the types and priorities of services required by each terminal. Then, a heuristic algorithm is used to plan the transmission paths between SFCs to reduce the load of each SF and improve the overall network performance. Simulation results show that the proposed method can shorten the time of data transmission and achieve load balancing.

Keywords Software-defined network, Load balancing, Data bandwidth, Service function chain

1 引言

随着信息化的飞速发展,人类对互联网的需求越来越大,面向用户的服务功能部署规模也随之扩大,因此,传输的数据量也将增加。通常可以通过提高网络设备的性能或者更新网络硬件设备来解决网络负载越来越大的问题,但是,这些解决方案往往会大幅提高成本投入。因此,亟需研究一种不仅可以增加网络负载,同时可以减少数据传输时间的方法。

软件定义网络(Software Defined Networking, SDN)是一种新兴的可编程的网络架构[1]。SDN的OpenFlow[2]技术能够分离传统网络中的控制与数据转发的统一层面,形成控制层面和数据层面[3]。数据层面的交换机只负责数据转发工作,并不管理网络流表;控制层面从宏观上掌握全局的网络视图,通过对网络全局的掌握来下发数据转发的流表,提高网络利用率。SDN数控分离的结构实现了网络设备的集中化、精细化管理。SDN的优势在于控制器具有网络的全局视图,因此它可以在这个过程中以更全面的方式建立转发规则,并根据网络状态实时调整转发规则,最终达到平衡负载的目标。网络功能虚拟化(Network Function Virtualization, NFV)与SDN同时被提出。NFV从硬件和网络中提取网络功能并用软件替换这些功能,因此,所有的网络服务可以按照逻辑集中进行管理。通过这些软件,网络可以受控。这两个方法已经共同使用,简称服务功能链(Services Function Chains, SFC)[4]。用户的请求将转发到SFC,以优化路线分配。

在真实环境中,网络资源分布不均匀会导致网络传输拥塞。因此,必须保证网络负载均衡。

为了实现优化解决方案,本文采用了模拟退火算法(Simulated Annealing, SA)[5],通过接受相对较差的解决方案来选择数据包传输路径,从而避免算法陷入局部最优,同时减少数据包的传输时间。另外,算法基于数据包进入SDN控制器的顺序对其进行了分配,较早进入SDN控制器的数据包将具有较高优先级。因此,稍后的分组服务请求应根据SF(Services

到稿日期:2018-08-24 返修日期:2018-11-27 本文受国家自然科学基金项目(61374054)资助。
张 钊(1993-),男,硕士,主要研究领域为计算机网络,E-mail:657455664@qq.com;李海龙(1978-),男,博士,副教授,主要研究领域为计算机网络,E-mail:35244637@qq.com(通信作者);胡 磊(1995-),男,硕士,主要研究领域为计算机网络;董思岐(1995-),女,硕士,主要研究领域为计算机网络。

Function)的负载增加传输路径的总距离以访问更远的网络服务服务器(Network Services Service,NSS)。为了高效地分配分组传输路径,本文引入了面向服务的算法[6]。它将数据包分类并规划各自的路径,避免由 SFC 的直接规划导致的拥塞。

2　相关研究

为了解决网络负载的问题,很多学者已对 SDN 负载均衡进行了研究。文献[7]假定负载均衡器使用 SDN 执行软件功能,设计并实现了所提出的负载均衡算法,该算法减少了数据传输的延迟和传统负载所需的额外负载平衡器;此外,他们还提出了基于循环法均匀分配负载的方法,模拟结果证明了该方案的可行性及有效性。文献[8]考虑了带宽的标准偏差,用每条链路的使用率作为负载的标准平衡,标准偏差越小则每个带宽利用率的差异越小,负载也更平衡。文献[9]提出了基于蚁群优化算法的链路负载均衡,该算法将链接负载、延迟和包丢失添加到蚂蚁算法中,以识别最短路由路径和减少节点之间的最小延迟;但是,蚂蚁算法的迭代时间较长,并且计算成本较高,成功率较低。文献[10]使用排队模型来实现负载均衡,但没有考虑传输包的时间,这在实际中会导致错误的负载平衡。文献[11]通过简短的计算规则实现流量的目标分配,同时自动适应负载平衡策略的变化。但是,这个解决方案具有可扩展性的限制。文献[12]考虑负载崩溃流量的中间传输,提出了一个 Load-Balanced 路由与 OpenFlow(LABERIO)相结合的方法来解决传输过程中生成负载不平衡的问题。文献[13]提出了一种基于服务器响应时间的负载平衡(Load Balancing scheme Based on Server Response Times,LBBSRT)方法,其使用控制器获取响应,服务器选择服务器的最小时间或最稳定的响应时间。文献[14]采用模糊综合评估机制(Fuzzy Synthetic Evaluation Mechanism,FSEM)来评估分组传输路径。路径可以借助 FSEM 进行动态调整以实现负载平衡。文献[15]在网络功能中提出了一个两阶段的算法,算法虚拟化(NFV)找到最低的网络等待时间,并且满足必须的条件,即服务通过数据包传递,如防火墙和视频转码。该算法的第一阶段采用了贪婪策略来选择网络等待时间延迟最低的服务器,并建立服务链。基于第一阶段的服务链,第二阶段试图与之交换服务对序列的要求。换句话说,服务链中的序列发生了变化,因此算法须评估新服务链的质量。如果新的服务链可以减少网络等待时间,则该服务链将被采纳;否则,服务链保持不变。这种方法可以减少网络延迟并提供网络服务。但是,上述方法都没有说明数据包处理是连续的并且按照分组可以被分类。尽管用户要求数据包最靠近所需的 NSS,但因为持续传输数据包将导致 SF 过载,一些数据包必须发送服务请求给更远的 NSS。当通过传输网络的数据量不断增长时,网络规划的复杂性则逐渐增加。若数据包不能有效地分配给其他网络服务,则可能会导致网络因过载而瘫痪。

3　问题的定义

本文旨在通过设计 SFC 来解决负载均衡问题。在设定的场景中,用户设备(User Equipment,UE)将服务请求发送给控制器,控制器分配 SFC 并提供服务路径。因此,服务功能链模型被定义为 $Set\_SFC=\{sfc_1,sfc_2,\cdots,sfc_k\}$,$Set_SF=\{sf_1,sf_2,\cdots,sf_n\}$和集合 $Set\_UE=\{ue_1,ue_2,\cdots,ue_m\}$。其中 k 是 SFC 的数量,n 是 SF 的总数,m 是 UE 的总数。sfc_i 代表第 i 个 SFC,sf_f 代表第 f 个 SF,ue_q 代表第 q 个 UE。每个 SF 都有自己的带宽容量 SF_BW_j 并运行一种服务类型,SF_Type_f 代表 sf_f 的服务类型,如防火墙和入侵检测。每个用户都有不同的数据包大小需求、带宽、需传递的服务和目的地。

本文设计 SFC 连接每个 SF,采用 SFC 来控制流量路径,并减少流量整体的平均数据传输时间来实现负载均衡。sfc_i 的定义如式(1)所示:

$$sfc_i=\begin{bmatrix} sf\_s_{(1,1)},sf\_o_{(1,1)} & \cdots & sf\_s_{(1,j)},sf\_o_{(1,j)} \\ \cdots & \cdots & \cdots \\ sf\_s_{(i,1)},sf\_o_{(i,1)} & \cdots & sf\_s_{(i,j)},sf\_o_{(i,j)} \end{bmatrix} \tag{1}$$

每个 SFC 表示一个矩阵。sf_s 是布尔值,表示 sf_f 在 sfc_i 中的使用状态,$sf\_s=1$ 意味着 sf_f 将用于 sfc_i,反之亦然。sf_o 表示将被使用的 sf_f 的序列。

SFC 的时间成本如下:

$$SFC\_Cost_j=T\_Cost_i+S\_Cost_i+W\_Cost_i \tag{2}$$

SFC_Cost_j 是每个 sfc_i 的传输时间,包括传输成本(T_Cost_i)、服务时间成本(S_Cost_i)和服务等待时间成本(W_Cost_i)。所有 SFC 的传输时间的总和 $Total\_Cost$ 如下式所示。

$$Total\_Cost=\sum_{j=1}^{k}SFC\_Cost_j \tag{3}$$

为了实现高性能路由分配,我们计算了平均传输时间来评估网络整体的传输质量。

$$Avg\_Cost=\frac{Total\_Cost}{k} \tag{4}$$

在一段时间 T 内,每个 SF 的带宽负载可以通过添加 sf_j 的带宽需求获得。$UserBW\_Request_q$ 是 ue_q 的需求带宽。

$$SF\_BWLoad_j=\sum_{i=1}^{k}sf\_s_{(i,j)}\times UserBW\_Request_q \tag{5}$$

通过式(6)计算 sf_j 的带宽负载,其主要用于评估 sf_j 的负载情况:

$$SF\_BWoverload_j=SF\_BWLoad_j-SF\_BW_j \tag{6}$$

sf_j 的剩余带宽为:

$$SF\_BWSurplus_j=SF\_BW_j-SF\_BWload_j \tag{7}$$

我们将数据包分类为固定的和在时间 T 内的非固定分组数据包。如果数据包使用相同的服务,则可以称为固定分组,否则被称为非固定数据包。T 是一个独立变量,需要根据环境进行调整。环境是不可预测的。T 越短,则固定数据包的数量越多,反之亦然。如果分组传输不稳定,为获得更多的非固定数据包来分配,可以将 T 值调小。$StatusPacket_z$ 表示第 z 个数据包的状态。$StatusPacket_z=1$ 表示固定分组,否则表示非固定的数据包。根据 SDN 框架可以知道在一段时间内的数据包数量和投入使用的 SFC 数量。SFC 的负载量可以通过下式计算:

$$SF\_BWoverload=\sum_{j=1}^{n}SF\_BWoverload_j \tag{8}$$

本文使用线性规划模型来定义目标,旨在尽量减少平均时间成本。在传输时间最小化的过程中,所有的数据包不仅

可以快速传输到 NSS,同时可以在最短的等待时间内回应服务请求。这种方式可以间接实现负载均衡。此外,为了遵守真实环境,这项研究定义了一些限制条件:首先,至少应该有一种 SF 服务类型(如防火墙、数据过滤或数据分析)来满足用户需求。SF_TypeNB 代表场景中 SF 服务类型的数量。因此,至少应该有一种类型的服务。其次,用户可以在已知的服务中提出请求类型,不应该请求没有存在的服务。$UserSF\_Request_q$ 代表 SF 类型的数量,另外,还需要考虑传输成本的价值,服务时间成本和服务等待时间成本不可以是负值。最后,通过计算负载获得 SF 的过载带宽。SF 的最小超载应该为零,而不应该是负值。

约束条件:

$SF\_TypeNB>1$

$UserSF\_Request_q<SF\_TypeNB$

$T\_Cost_i, S\_Cost_i, W\_Cost_i>0$

$SF\_BWoverload_j\geqslant 0$

4 服务导向方法

一些物联网设备在特定的时间需要特定的服务。例如,感应节点将收集到的数据发送到云端进行处理时,将通过防火墙服务,而这样的行为感应一直在进行。如果固定业务的分组被转发到负载相对较低的 SF,SFC 则可以提供更多的非固定分组。为了避免陷入网络路径局部最优解,本文采用模拟退火算法优化 SFC。如果用户数量过高,则使用模拟退火算法将耗费更多的时间。因此,本文也使用了常见的贪婪算法,以迅速建立 SFC 来应对大量的用户并减少计算时间。同时,我们引入了服务导向的概念算法。

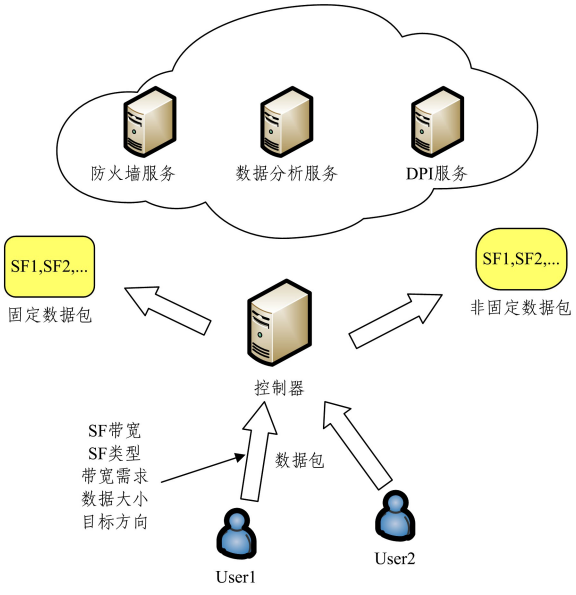


图 1 网络框架

Fig. 1 Network framework

图 1 为本文研究的网络框架。我们将 SFC 的生成方式分为固定分组规则和非固定分组规则。当用户发送 $UserBW\_Request_q$ 和 $UserSF\_Request_q$ 时,SDN 控制器将根据数据包的类型使用不同的 SFC 配置规则。

图 2 为本文方法的流程。方法第一步是确定数据包是否为固定,如果 SF 由一个固定的数据包加载,数据包将以最大值安排到 SF 的剩余负载中,使得固定数据包的服务状态较为稳定。如果将数据包转换为 SF,在更低的负载下,可以将服务组合分配至非固定的数据包,从而更好地实现负载平衡效果。相反,若一个数据包是固定的,则使 SFC 通过不同的算法规则。下面介绍负载平衡规则的贪心算法和 SA 算法。

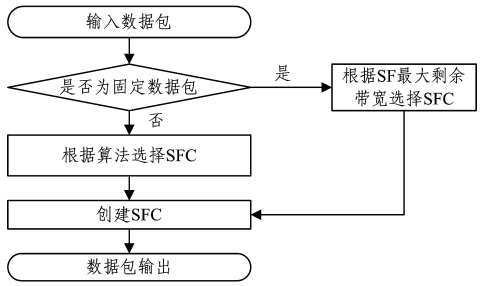


图 2 服务导向方法的流程

Fig. 2 Flow of Service-oriented method

一般来说,建立数据传输的最常见的策略是处理最近目标的贪婪算法。本文考虑了距离并结合了服务导向的概念,提出了基于贪婪的服务导向算法(Greedy-based Service-orientation Algorithm,GSOA)。

4.1 基于贪婪的服务导向方法

GSOA 的主要策略是选择最近的 SF。第一步是确定数据包是否为固定。如果这个数据包是固定的,则所有 SF 提供的服务会根据各自的服务请求进行识别,并选择负载最低的 SF,通过 SFC 的 SF 将被标记为 $sf\_s_{(i,j)}$,它们的序列将被记录为 $sf\_o_{(i,j)}$。如果数据包不是固定的,则所有 SF 提供的服务会根据各自的服务请求进行识别,并选择最接近的 SF。将通过 SFC 的 SF 标记为 $sf\_s_{(i,j)}$。它们的顺序记录为 $sf\_o_{(i,j)}$。GSOA 的基本流程如算法 1 所示。

算法 1 GSOA

Input:$UserBW\_Request_q$, $UserSF\_Request_q$

Output:sfc_i

1. For each $UserBW\_Request_q$
2. If ($SF\_BWoverload_j>0$ && $StatusPacket_z=1$)
3. For each sf_j
4. If SF_Type_j meet the service requests
5. If has maximum $SF\_BWSurplus_j$
6. Replace the current choice;
7. Update the $sf\_s_{(i,j)}$ and $sf\_o_{(i,j)}$;
8. End If
9. End If
10. End For
11. Else
12. For each sf_j
13. If SF_Type_j meet the request
14. If sf_j is closer $User_i$ than SF_{best}
15. Replace the current choice;
16. Update the $sf\_s_{(i,j)}$ and $sf\_o_{(i,j)}$;
17. End If
18. End If
19. End For
20. End If
21. End For

GSOA 可以快速生成 SFC，但它有明显的缺点，即在生成 SFC 时，只考虑了区域最优解，也就是说每个 SFC 的最佳解决方案。这种解决方案可能对整个网络来说不是最好的。因此，有必要使用启发式算法找到全局最优解。本文基于退火算法建立 SFC，并提出了基于 SA 的服务导向算法（SASOA）。

4.2　基于 SA 的服务导向算法

首先，基于 SA 的服务导向算法（SA-based Service-orientation Algorithm，SASOA）根据用户的服务需求随机生成 SFC，如图 3 所示。字段（$sf\_s_{(4,2)}$，$sf\_o_{(4,2)}$）的值是（1，3）。1 和 3 意味着第 3 个数据将通过 sf_2。如果值是（0，0），则意味着数据不会通过 SF 且没有顺序。然后设置 SASOA 的迭代。本文设定迭代规则重复执行，直到解决方案不再改变（如算法 1 第 1—2 行和算法 2 的第 12 行终止条件，其中 T_{end} 为终止温度，TP 为初始值温度，TR 为指随机变量，TU 表示下降速度）。接着，SASOA 将根据产生的 SFC 用随机的相同类型服务替换 SF。假设 sf_4 和 sf_5 服务类型相同，而使用这项服务的顺序是第二。如果用 sf_5 取代 sf_4，那么可将（$sf\_s_{(4,4)}$，$sf\_s_{(4,4)}$）的（1，2）改变为（0，0），并且将（$sf\_s_{(4,5)}$，$sf\_s_{(4,5)}$）的（0，0）改变为（1，2）。尽管 SFC 是随机产生的，每套 SFC 都必须满足用户的服务需求服务顺序。我们采用了面向服务这个概念的 SA 算法。

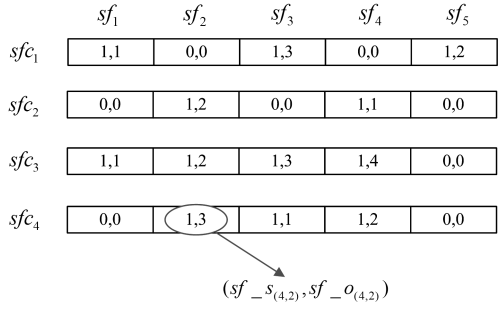


图 3　SFC 编码

Fig. 3　SFC code

然后，我们必须确定数据包是否是固定的。如果数据包固定，则可以使用式（8）来确定这个数据包通常使用的 SFC 的负载情况。如果 SFC 经常加载，则意味着固定的数据包可以转移到另一个 SF，还有更多非固定分组的 SFC 可以实现负载均衡。与 GSOA 相比，SASOA 考虑了负载状态和平均时间，以确定是否有必要访问新的 SFC。如果所有的条件都满足，那么就用新的 SFC 代替。同时如果 TP 大于 TR，则 TP 也进行新老代替（具体参见算法 2 第 5—10 行）。

如果数据包是非固定数据包，则下一步将根据本研究提出的目标类型确定新 SFC 的平均传输时间是否比旧 SFC 的平均传输时间长。如果满足模拟退火的标准或规则，则进行更换（具体参见算法 2 的第 12—17 行）。对于每次迭代，通过降低 TP 来使解决方案收敛，其目的是经过几次迭代来减小接受较差的解决方案的可能性。TU 将根据环境情况进行调整，其主要影响解决方案收敛的速度，过高的 TU 更容易使算法陷入局部最优并花费更多的传输时间。

算法 2　SASOA

Input：$UserBW\_Request_q$，$UserSF\_Request_q$

Output：sfc_i

1. Generate each sfc_i randomly;
2. Repeat
3. For each sfc_i
4.　Create new sfc_i according to the service requests of each user;
5.　if ($SF\_BWoverload_j$＞0 && $StatusPacket_z$＝1)
6.　if (new Avg_Cost＞old Avg_Cost && new SFC_OverLoad＜old SFC_OverLoad)
7.　sfc_i＝new sfc_i;
8.　Else if(TP＞TR)
9.　sfc_i＝new sfc_i;
10.　End if
11.　Else
12.　if (new Avg_Cost ＞ old Avg_Cost)
13.　sfc_i＝new sfc_i;
14.　Else if(TP＞TR)
15.　sfc_i＝new sfc_i;
16.　End If
17.　End If
18. End For

5　仿真结果

5.1　环境设置

在模拟实验中，假设特定号码和 SF 的用户的位置在 500 m * 500 m 范围内随机分布。SF 提供了 3 种类型的服务，每种服务都有至少有一个 SF。数据的大小设置为 1～10 MB。每个用户所需的服务数量为 1～3 个。80％用户的服务需求将随着时间的推移发生变化，其余的 20％是固定的。其他的参数设置如表 1 所列。

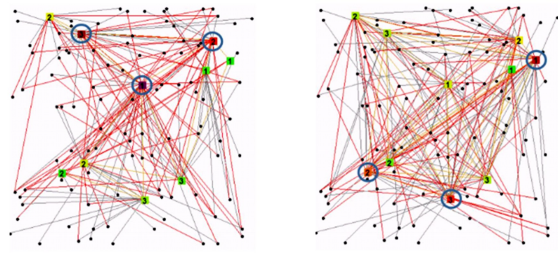
表 1　参数设置

Table 1　Parameter settings

参数	值
用户数量	25，50，100，150，200
用户带宽/ Mbps	1～100
SF 带宽/ Mbps	100～1000
数据包大小 / MB	1～10
用户服务需求数量	1～3

5.2　面向服务的方法和非面向服务的方法之间的比较

图 4 和图 5 显示了由 C＃执行生成的网络拓扑结构。其中黑色的点代表用户；浅色方块（绿色和黄色）代表正常的 SF；深色方块（红色、棕色和紫色）代表过载 SF；正方形中的数字代表服务的类型；灰线代表从源端到 SF 的链接；红线代表从 SF 到目的地的链接；黄线代表 SF 之间的联系。数据包将根据路径实施服务，路径的序列是灰线、黄线、红线。



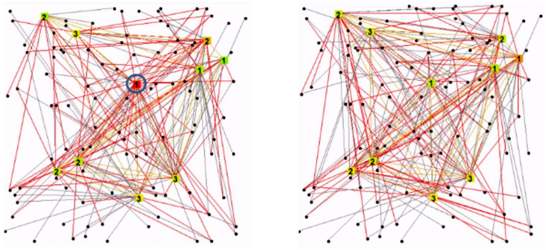
（a）贪婪算法拓扑　　（b）SA 算法拓扑

图 4　贪婪算法和 SA 算法的拓扑（电子版为彩色）

Fig. 4　Topology of Greedy algorithm and SA algorithm

SF 从 0 到 50%的负载用从绿色变化到黄色来表示;SF 50%至 100%的负载用从黄色变化到红色来表示;SF 从 100%到 150%的负载用从红色变化到棕色来表示;SF 从 150%到 200%的负载用棕色变化到紫色来表示。过载的 SF 以蓝色标记。

由图 4 的结果可知:以前的研究采用了贪婪的方法并通过模拟退火算法来实现负载平衡,但是没有考虑服务导向的因素。图 5 显示了 GSOA 和 SASOA 与面向服务的方法。通过服务导向后的方法,SF 的过载明显减少,因为这种方法可以使用较小的负载为固定服务的数据包进行 SF 转发。所有未被分类到固定和非固定的数据包将按照当前网络服务的状态进行分配,如图 4 所示。对于贪婪方法(见图 4(a)),分组将选择离用户最近的 SF。当用户集中时,很容易造成 SF 超载。相反,如果分组被分类并且固定分组被转移到 SF 低负载,被多个用户访问的 SFs 就会减少负载(见图 5(a))。



(a)GSOA 网络拓扑 (b)SASOA 网络拓扑

图 5 GSOA 和 SASOA 的网络拓扑(电子版为彩色)

Fig. 5 Topology of GSOA algorithm and SASOA algorithm

对于 SA 方法(见图 4(b)),虽然未分类的分组将根据平均时间成本最低分配,但是它的传输路径却变得更复杂。具有固定传输路径的分组将根据当前的网络服务状态改变它们的路径。因此,采用服务导向可以将固定数据包传输到低负载、平均时间成本低的 SF。当平均 SF 负载减少时,非固定数据包会有更多的 SFC 分配组合。因此,整个网络的负载平衡增强,如图 5(b)所示。

由上述结果可以得知,服务导向的方法能够将固定服务的数据包分配至负载较低的传输路径,从而增强网络的负载平衡性。以上结果在显示了服务导向方法优越性的同时也凸显了将数据包进行分类的必要性。本文通过提高固定数据包在整体网络中的占比进行对比实验,以验证将数据包按照固定数据包与非固定数据包进行分类是否会对整体网络的负载性能有所影响。

5.3 采用 20%的固定数据包进行性能评估

图 6 显示了不同类型的 20%固定数据包时总路径距离的比较,总路由距离即由所有数据包执行 SFC 距离的总和。SA 算法的总路由距离比其他 3 种算法长,SA 算法根据当前最低平均时耗的 SF 安排 SFC,然而这样很容易增加不必要的路径。SASOA 可以将固定数据包传送给具有更低时间成本和负载的 SF,虽然这会稍微增加分组的传输距离,但可以为不固定数据包的 SFC 提供更多的选择并减少总路由距离。当用户数量增加时,总路径距离的差距会越来越明显。首先生成传输时间短的路径并且忽略路径距离的 SA 和 SASOA 算法将会选择传输路径长但是时间短的路径。对于贪婪和 GSOA 这类首选距离用户最近路径的方法,总路由距离更短。由于采用了面向服务的引入方法,GSOA 会将固定数据包以最低的负载传送给 SF,使其总路由距离比 Greedy 方法稍长。

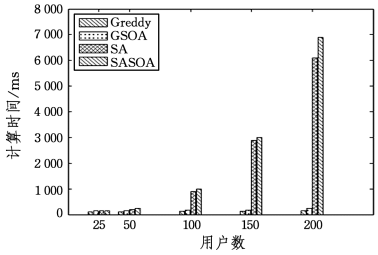


图 6 20%固定数据包时消耗的计算时间

Fig. 6 Calculation time under 20% fixed data packet

就平均传输时间而言,GSOA 和 SASOA 比 Greedy 和 SA 算法更好。其原因在于面向服务的方法(SASOA 和 GSOA)将固定分组转移到了更远的具有相同的服务和低负载的 SF。由图 7 可以看出,虽然这样可能导致整个路由距离增加,但每个数据包的服务等待时间可以大幅减少,资源可以均匀分配,因此平均传输时间有所减少。

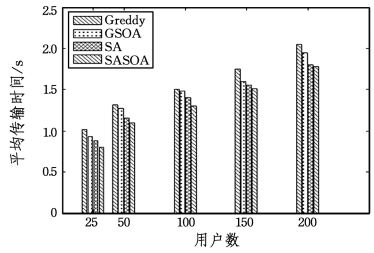


图 7 20%固定数据包时的平均传输时间

Fig. 7 Average transmission time under 20% fixed packets

图 8 比较了有 20%的固定数据包时 SF 的总过载。从中可知,SASOA 明显优于其他 4 种方法。尽管 SASOA 和 SA 都分配了一些流量超载的 SF 到轻载的 SF 以减少超载 SF 的负载和等待时间,但 SASOA 的优势在于它可以将 20%的固定数据包使用的服务转移到低负载的 SF,通过集中用户降低 SF 的负载。当用户数量增加,面向服务的 GSOA 和 SASOA 可以更明显地减少 SF 的负载。当用户数量减少时,SF 负载过大的问题也会得到缓解。

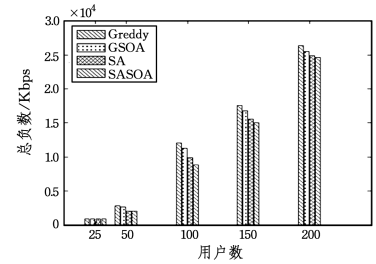


图 8 20%固定数据包时的总负载

Fig. 8 Total load under 20% fixed packets

图 9 比较了 4 种算法的计算时间。当用户增加时,SASOA 和 SA 的计算时间也将增加。SA 方法需要多次迭代来识别最佳方案,每次迭代都会生成一个新的 SFC 并确定时间

是否优于原来的 SFC,耗时较长。GSOA 和贪婪算法没有这一过程,因此计算时间较少。由模拟结果可以看出,当用户数量少于 50 时,SASOA 能够在更短的计算时间内实现更佳的平均传输时间。

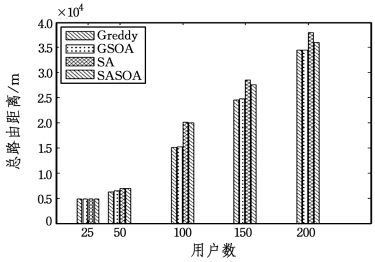


图 9　20%固定数据包时的总路由距离

Fig. 9　Total routing distance under 20% fixed packets

5.4　40%固定数据包的性能评估

为了更好地比较面向服务的方法的负载平衡,我们将整个网络的固定数据包从 20%提高到 40%,并比较 4 种算法。图 10 显示了固定 40%数据包时的总路由距离。与图 6 相比,我们可以看到,如果用户和 SF 的数量保持不变,贪婪算法和 SA 的总路由距离没有太大变化。当用户数低于 150 时,由于固定分组的增加,固定分组转移到低负载 SF 的速率增加,因此,GSOA 的总路由距离与 SASOA 相近。但是,当用户数量超过 150 时,由于用户的增加,SF 负载的数据转移速率增加,因此,可降低固定分组的转移速率,所有数据包都根据离用户最近的 SF 分配,使得 GOSA 的总布线距离比 SA 低。因为 SASOA 不会将固定数据包传输到最小负载的 SF,所以 SASOA 的整体传输路径得到减少。当整体网络的固定数据包比例上升时,固定的分组可以被转移到接近用户或有更少的负载的 SF,这样可以减少总路由距离。

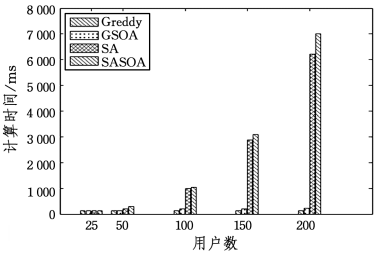


图 10　40%固定数据包时的计算时间

Fig. 10　Calculation time under 40% fixed data packet

图 11 比较了 4 种算法在 40%是固定数据包的情况下的平均传输时间。传输时间主要影响传输成本、服务时间成本和服务等待的时间。一旦 SF 的负载增加了服务等待时间,则整体 SFC 的传输时间也将增加。虽然 GSOA 的总传输距离的增加将导致传输成本有所增加,但随着固定数据包的传输被转移给负载较低的 SF,整体服务等待时间将会减少。整体 SFC 的传输成本远低于有 20%的固定数据包的情况,甚至接近 SA 的水平。与图 7 相比,由于固定的数据包比例从 20%变为了 40%,增加了传输给距离用户更近或负载更低的 SF 的固定数据包,GSPA 和 SASPA 的平均传输时间显著减少。使用远离用户 SF 的非固定数据包的比例减少,传输时间和服务等待时间降低,整体 SFC 的传输时间也相应减少。

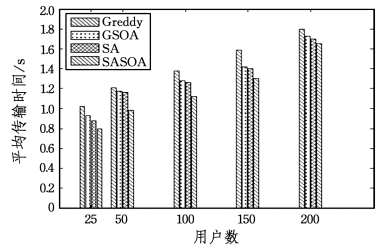


图 11　40%固定数据包时的平均传输时间

Fig. 11　Average transmission time under 40% fixed packets

图 12 比较了有 40%的固定数据包时 SF 的总超载。比较图 12 和图 8 可以看出,GSOA 和 SASOA 的负载明显减少。GSOA 的负载甚至接近 SA 算法,这种趋势随着用户数量的增加愈加明显。固定数据包为 20%时,虽然服务的概念被引入到 GSOA 中,但贪婪算法容易陷入区域最优解的缺陷仍然存在。即使有些数据包转被移到其他 SF,80%的非固定数据包仍然选择离用户最近的 SF。然而,当固定数据包的数量增加到 40%时,转移到 SF 的固定分组数量得到了增加,从而解决了贪婪算法造成的非固定数据包的问题,并且随着用户数量的增加,这个趋势将更加明显。

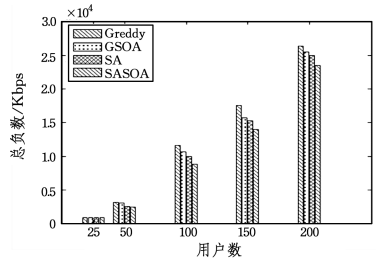


图 12　40%固定数据包时的总负载

Fig. 12　Total load under 40% fixed packet

图 13 比较了 4 种算法的计算时间。虽然服务导向方法可以降低平均 SF 负载,这是因为其增加了固定和不固定数据包的判断和不同数据包规则的选择,但是 SASOA 和 GSOA 的计算时间都会比 20%的固定数据包时的计算时间更长。因为 GSOA 对固定和非固定数据包的路径选择方法都基于贪婪算法,所以当固定数据包的比例增加时不会产生较大的影响。因此 SASOA 考虑了最小传输时间和 SF 的最低负载,所以迭代数量比 SA 更多。

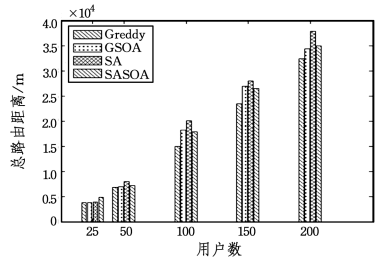


图 13　40%固定数据包时的总路由距离

Fig. 13　Total routing distance under 40% fixed packets

以上结果表明,面向服务的 SASOA 方法可以有效地通过转移固定数据包至其他较少负载的 SF 来实现负载均衡,

但计算时间较长、成本较高。当用户较多时,建议使用GSOA。虽然GSOA不能达到与SASOA相同的效果,但是它可以及时处理数据包的传输方式,与SASOA相比降低了计算时间。如果用户数量很少或者硬件计算能力很强,则优先选择SASOA。与GSOA相比,虽然SASOA需要更多的计算资源,但是它在减少数据传输时间和平衡SF负载方面更有效。

结束语 为了满足人们对网络的需求服务和解决由于用户过多引发的网络超负载甚至网络瘫痪的问题,本文在数控分离的SDN的基础上提出了GSOA和SASOA来减少物联网终端的数据传输时间,并平衡SF的负载。实验结果表明:本文提出的面向服务概念能够通过将固定数据包转发到较少负载的SF上来有效地实现不同SF之间的负载均衡。

参 考 文 献

[1] Stanford University. Clean slate program[OL]. http://cleanslate.stanford.edu/.

[2] MCKEOWN N,ANDERSON T,BALAKRISHNAN H,et al. OpenFlow:Enablinginnovation in campus networks[J]. ACM SIGCOMM Computer Communication Review,2008,38(2):69-74.

[3] Open Networking Foundation. Software-Defined networking:The new norm for networks[M]. ONF White Paper,2012.

[4] XUE W,FU G. Research on Key Technology of Service Chaining Based on SDN/NFV[J]. Designing Techniques of Posts and Telecommunications,2015(2):1-6. (in Chinese)
薛淼,符刚. 基于SDN/NFV的Service Chaining关键技术研究[J]. 邮电设计技术,2015(2):1-6.

[5] BERTSIMAS D,TSITSIKLIS J. Simulated Annealing[J]. Statistical Science,1993,8(1):10-15.

[6] WANG S,LIU Z,SUN Q,et al. Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing[J]. Journal of Intelligent Manufacturing,2014,25(2):283-291.

[7] KAUR S,KUMAR K,SINGH J,et al. Round-robin based load balancing in Software Defined Networking[C]// International Conference on Computing for Sustainable Global Development. IEEE,2015.

[8] ZHU L,TANG R,TAO Y,et al. Multi-objective Ant Colony Optimization Algorithm Based on Load Balance[C]// Cloud Computing and Security(ICCCS 2016). Springer,2016:193-205.

[9] CHEN L,WANG Y,WANG H,et al. Multiple-combinational-channel:A network architecture for workload balance and deadlock free [J]. Future Generation Computer Systems,2016,56:238-246.

[10] BANAIE F,YAGHMAEE M H,HOSSEINI S A. SDN-based scheduling strategy on load balancing of virtual sensor resources in sensor-cloud[C]// International Symposium on Telecommunications. IEEE,2017:666-671.

[11] WANG R,BUTNARIU D,REXFORD J. OpenFlow-based server load balancing gone wild[C]// Usenix Conference on Hot Topics in Management of Internet,Cloud,and Enterprise Networks and Services. USENIX Association,2011:12.

[12] LONG H,SHEN Y,GUO M,et al. LABERIO:Dynamic load-balanced Routing in OpenFlow-enabled Networks[C]// International Conference on Advanced Information Networking and Applications. IEEE Computer Society,2013:290-297.

[13] ZHONG H,FANG Y,CUI J. LBBSRT:An efficient SDN load balancing scheme based on server response time[J]. Future Generation Computer Systems,2017,68:183-190.

[14] LI J,CHANG X,REN Y,et al. An Effective Path Load Balancing Mechanism Based on SDN[C]// IEEE,International Conference on Trust,Security and Privacy in Computing and Communications. IEEE Computer Society,2014:527-533.

[15] THAI M T,LIN Y D,LAI Y C. A joint network and server load balancing algorithm for chaining virtualized network functions [C]// IEEE International Conference on Communications. IEEE,2016:1-6.