文章编号:1671-4598(2018)03-0195-05 DOI:10.16526/j.cnki.11—4762/tp.2018.03.049 中图分类号:TP193 文献标识码:A

基于 CloudSim 的分类负载均衡调度模型

李荣荣，牛立栋，孙纪敏
（中国电子科技集团公司 第五十四研究所，石家庄 050081）

摘要：针对传统的集群调度模型效率低下不足以满足用户需求的问题，提出一种基于 CloudSim 的分类负载均衡调度模型；首先，构建任务请求的指标体系以完成数学模型的建立；接着，采用基于模糊 C 均值聚类算法的改进算法对请求分类，即用改进的最小支撑树算法获取初始中心，有效性测度获取其分类个数，BP 神经网络算法提高其学习能力；然后，采用两次分类的方法对服务器分类，预聚类对服务器进行功能预聚类，模糊关联聚类按处理负载能力对其分类；最后将分类调度模型在 CloudSim 下仿真实验，综合 3 种场景，相比于其他调度算法，分类调度模型的最大运行时间最低，资源利用率最高且最高至 99%，结果表明该模型更具适应性和高效性，具有工程指导意义。

关键词：负载均衡；模糊聚类；BP 神经网络；CloudSim；调度

Classified Load Balancing Scheduling Model Based on CloudSim

Li Rongrong，Niu Lidong，Sun Jimin
（The 54th Research Institute of CETC，Shijiazhuang 050081，China）

Abstract：Aiming at the problem that traditional cluster scheduling models are not efficient enough to meet the needs of users，now propose a classified load balancing scheduling model based on CloudSim. Firstly，construct the index system of task request to build the mathematical model；next，use the improved algorithm based on fuzzy C—means clustering algorithm to classify the request，obtaining the number of classification after determining its initial center and improving its learning ability by BP neural network；then，pre—cluster servers according to the function and use the fuzzy clustering algorithm to classify them according to their processing ability. Finally，the classification scheduling model is simulated under CloudSim. By comparing with other scheduling algorithms in three different scenarios，classified model has the shortest maximum running time and the resource utilization up to 99%. The results show that the proposed model is more adaptive and efficient and has engineering significance.

Keywords：load balancing；fuzzy clustering；BP neural network；CloudSim；scheduling

0 引言

负载均衡技术是集群工作的关键技术，负载均衡算法是影响集群效率的决定性因素，因此研究并提出一个性能良好的负载均衡调度算法对提高集群工作具有重要意义。

对于软件负载均衡调度技术，研究学者一般重点解决节点数据倾斜问题以及数据分类问题[1-3]，采用改进算法对资源进行分类以实现“合适的资源服务于合适的任务”这一原则[4-5]，通过反馈机制实现负载的实时监测以更精确表示节点负载情况[6-7]。

本论文将节点负载的研究和资源分配的研究结合起来，通过区分不同的任务请求，使得同一类型的请求分派到同一类型的服务器节点上处理，以此实现资源分配最优，效率最大化实现，系统也达到更好的均衡效果。

1 分类负载均衡模型

根据集群任务调度的过程，本文抽象出一种负载均衡模型，即分类负载均衡调度模型（CLBM）。该模型共分为三层：应用层、任务层和资源层。应用层由获取数据模块构成，包括提交请求和用户登录；任务层由请求分类模块和任务调度模块构成，请求分类模块包括接受用户请求和算法分类，任务调度模块包括任务指派和任务调度；资源层由资源分类模块和返回模块构成，资源分类模块包括服务器预分类（降维）和模糊关联聚类，返回模块包括服务器处理请求和返回用户请求。该模型各模块具体功能如下。

获取数据模块：该模块主要功能是统计提交的用户请求。用户登录系统并提交请求，用户提交的请求就作为下一步分类的数据来源。

请求分类模块：该模块的主要功能是请求任务的分类。服务器间隔固定的时间统一接收用户请求，分析任务请求的属性特征，构建指标体系，根据请求分类算法对请求分类。

任务调度模块：该模块的主要功能是请求任务的调度。将分好类别的任务请求分派到对应类别的任务指派器上，然后顺序选择服务器对任务请求进行任务调度。

资源分类模块：该模块主要功能是对服务器进行两次分类。首先，对服务器功能特性对服务器进行预分类，以减少冗余，降低后续工作的复杂度；然后根据服务器特征参数对其进行模糊关联聚类，使得服务器具有清晰的划分。

返回模块：该模块主要功能是处理请求任务。首先属于某一类别的服务器处理对应类别的任务请求，并将处理结果反馈给用户。

因此，在该模型下，负载均衡调度的全过程如下：首先登录系统并提交请求，作为下一步分类的数据来源；服务器间隔固定的时间统一接收用户请求，分析任务请求的属性特征，构

收稿日期:2018-01-02； **修回日期**:2018-01-05。
基金项目:国家自然科学基金(61671179,61504124)。
作者简介:李荣荣(1993-),女,河北石家庄人,硕士研究生,主要从事负载均衡,云计算方向的研究。

建指标体系，根据请求分类算法对请求分类；将分好类别的任务请求分派到对应类别的任务指派器上，使得访问同一类型的请求对应相应类型的服务器，然后顺序选择服务器对任务请求进行任务调度；最后，属于某一类别的服务器处理对应类别的任务请求，并将处理结果反馈给用户。分类负载均衡模型如图 1 所示。

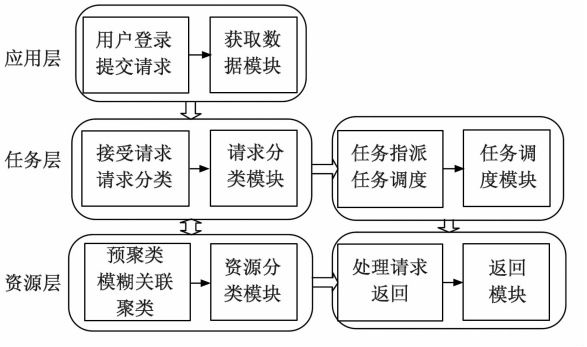

图 1 分类负载均衡模型

根据上述分类负载均衡模型各个模块的主要实现功能，该模型主要实现构建指标体系、对请求分类、对服务器分类三大功能。

构建指标体系：确定任务请求的主要特征属性，并根据这些属性确定任务请求的指标体系，作为下一步对请求分类的数据来源与数据依据。

对请求分类：按照由任务请求属性构建的指标体系，采用某种有效的数学算法，对请求进行分类。

对服务器分类：按照服务器的特征参数，对服务器进行分类，并与对请求的分类形成呼应。使得同一类大小的请求负载对应同一类处理能力的服务器。

因此，对分类负载均衡模型建模过程如下：

设 Web 集群服务器由一个分配器和 M 个后台服务器 RS_1，RS_2，...，RS_M 组成，服务器按照处理负载能力大小分为 c 类。我们将用户请求 $x=\{x_1, x_2, \ldots, x_n\}$ 按照任务量大小划分为 c 个不相交的区间 $\{[s_0=0, s_1], [s_1, s_2], \ldots, [s_{N-1}, s_C]=\infty\}$，并且让第 i $(1\leqslant i\leqslant c)$ 类服务器仅仅服务请求大小落在第 i 个区间 $[s_{i-1}, s_i]$ 的请求，即实现访问同一类型的请求分配到同一类型的服务器处理的初衷。

2 分类负载均衡模型的实现

2.1 构建指标体系

针对某项目系统，用户基本操作主要有以下 4 种：登录、增删改查、提交和 GPU 计算。一般操作对象为视频和文档，针对某系统现状，主要操作对象为文档。传统的调度算法只考虑了请求文档的大小，忽略了文档的其他属性。比如请求文档的紧急程度，文档的是否涉密等。而实际应用中，忽略这些属性往往会产生较大的误差，造成资源分配不合理，请求响应时间增加，不满足用户需求。这里我们定义文档属性为文档大小、紧急程度、涉密程度。因此本论文中我们定义 4 个指标：视频/文档、大小、紧急程度和涉密程度。

定义 4 个指标为请求数据的 4 个属性，为了便于分析，我们将指标量化数值表示优先级高低，如表 1 所示。将属性进行加权求和，请求的任务量大小用每个请求的属性加权和来表示，根据量化数据的加权和来划分分类范围。

表 1 数据指标的属性及量化值

文档/视频		文档大小		紧急程度		涉密程度	
文档	0	小文档	1	非紧急	1	非涉密	1
视频	1	中文档	2	紧急	2	涉密	2
—	—	大文档	3	加急	3	重要涉密	3
—	—	—	—	—	—	核心涉密	4

2.2 对任务请求分类

任务请求到达的方式有很多种，我们假定是在访问量稳定的情况下。以时间片为单位，将在某时间片内的所有请求进行分类。因此，请求的分类需要高频随时进行，需要选用相对简单复杂度低的算法，且算法运行时间要短，这里我们选择模糊 C 均值（FCM）聚类算法。

利用最小支撑树（MST）算法可以得到请求数据的初始聚类中心，算法原理是在数据密集区选择密度最大数据节点作为第一个类的初始中心，然后在距离第一个初始中心足够远的区域内选择数据密集区最大的数据节点作为第二个类初始中心，以此类推[8]。

我们对 MST 算法进行改进，得到改进的最小支撑树算法（IMST），该算法降低最小支撑树算法（MST）的计算复杂度，解决了 MST 算法初始点选取的冗余问题。现给出如下两个定义：

定义一：用加权属性和之差来定义两节点的距离。即：

$$d_{ik}=\sum_{q=1}^{m} w_{iq}c_{iq}-\sum_{q=1}^{m} w_{kq}c_{kq} \tag{1}$$

其中：d_{ik} 表示节点 x_k 到聚类中心 v_i 的距离，m 为属性的个数，w 属性值大小。

定义二：定义所在节点的度与分离度之积为该节点的稀疏度，分离度指两节点间的距离。即：

$$S_k=n_k\times d_{ik} \tag{2}$$

其中：S_k 为节点 x_k 相对于聚类中心 v_i 的稀疏度，n_k 为节点 x_k 的度，d_{ik} 为节点 x_k 到聚类中心 v_i 的距离。

实际计算中，我们无须将所有节点都作为初始聚类中心的候选点，为减少计算量，我们选择节点的度大于平均值的点作为候选点[9]。

这里采用类间最小距离原则：首先选取节点中度最大的值作为第一个初始聚类中心，然后更新候选点集合，去掉距离第一个聚类初始化中心为 0 的候选点，计算新候选点集合到第一个初始聚类中心的分离度，选择分离度最大的点作为第二个初始聚类中心，继续更新候选点集合，去掉距离第一、二初始聚类中心为 0 的候选点，然后从新候选点集合中选取分离度最大的点作为第三个初始聚类中心，如此反复直至候选点集合为空。此时得到准初始聚类中心集合。根据有效测度指标确定最佳聚类数，由此确定最终的初始聚类中心点。

算法 1：IMST 算法

输入：请求数据 Data

输出：初始聚类中心

Step1：利用请求数据，以数据为节点，两节点间距离作权值，构造最小支撑树；

Step2：求各请求节点的度 n_k 及平均度 n_{avg}，若 $n_k>n_{avg}$，

则 $x_k \in N$(N 为候选中心集合)；

Step3：利用类间最小距离原则：取 $n_{\max}$ 的节点作为 p_{01}(p_{0i} 表示第 i 个初始聚类中心)，然后更新 N(为避免重复，去掉 $d_{1k}=0$ 的节点)，计算更新后 N 中元素到 p_{01} 的稀疏度 S_1，取 $S_{\max 1}$ 对应的节点作为第二个初始中心 p_{02}；

Step4：重复 Step3 直至 N 为空集，输出初始聚类中心；

Step5：算法停止。

针对 FCM 算法提出了很多有效性指标，使用以下有效性测度指标来确定其聚类个数，定义如下：

$$S_c=\frac{\sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m d_{ik}^2}{n\cdot \min\limits_{i,k} d_{ik}^2}=\frac{J_m}{n\cdot \min\limits_{i,k} d_{ik}^2} \tag{3}$$

其中，$m>1$ 是模糊系数；u_{ik} 是第 k 个请求 x_k 属于第 i 类的隶属度值；v_i 为第 i 类的聚类中心，距离 d_{ik} 表示节点 x_k 到聚类中心 v_i 的距离，$J_m=\sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m d_{ik}^2$ 为 FCM 聚类算法的目标函数。

BP 神经网络算法比较实际输出与理想输出的差值反向修改神经元之间的权值来实现反馈迭代，因此算法具有较强的适应性和学习能力，这弥补了 FCM 聚类算法因其模糊性造成的适应能力差的缺陷，将 BP 算法应用于 FCM 聚类算法可以增加其学习能力并提高准确识别率。在 FCM 的输出数据中，选取距离最佳聚类中心稀疏程度最小的部分数据作为 BP 算法的训练输入数据，使网络具有对输入请求分类的能力，然后用训练好的 BP 神经网络预测请求所属类别。其算法流程图如图 2 所示。

算法 2：模糊神经网络算法（MBFCM 算法）

输入：待分类请求数据 Data，FCM 停止阈值 $emps$，模糊指数 m，BP 停止阈值 e，BP 算法学习速率 η。

输出：最佳聚类数 c，正确识别率 h。

Step1：输入待分类请求数据 Data，根据 MSTZ 算法求得初始聚类中心集合；

Step2：根据有效性测度确定初始分类中心 v_0 和最佳分类个数 c；

Step3：运行从指定中心开始的模糊 C 均值（MFCM）聚类算法，得到最佳聚类中心 v；

Step4：计算个体到聚类中心的分离度并选择稀疏度程度最小的部分数据进行网络训练；

Step5：网络预测与结果统计，得到正确识别率 h。

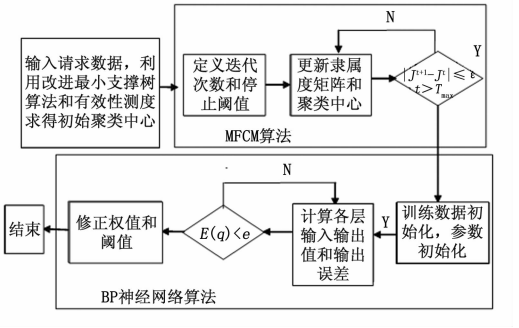

图 2　MBFCM 算法流程图

2.3　对服务器分类

根据服务器结构描述，现对服务器进行两次分类。首先对服务器进行预聚类（功能分类），从而避免因服务器属性过于复杂导致算法计算复杂度过高，并且有效提高任务调度的精准性；紧接着运用模糊关联聚类算法对服务器进行性能分类，然后 F 统计量确定服务器分类个数，并用编网法获取最佳聚类划分。使同一类别的请求分派到后台对应类别的服务器上处理，避免小任务排在大任务后面受到巨大的响应延时。

因服务器特征参数复杂繁琐，且变化幅度较大，其分类个数用 F 统计量来获取。F 统计量根据样本空间中数据的特征来划分类别，通过类间距离和类内距离比值量化分类效果的好坏，显然，类与类之间的距离越大，类内样本数据之间的距离越小，同一类别的数据分布越稠密，不同类别的数据越分散，分类效果就越好[10]。

F 统计量公式如下：

$$F=\frac{\sum_{j=1}^{r} n_j \left\| \overline{y}^{(j)}-\overline{y}\right\|/(r-1)}{\sum_{j=1}^{r}\sum_{i=1}^{nj}\left\| y_i^{(j)}-\overline{y}^{(j)}\right\|/(n-r)} \tag{4}$$

其中：$\left\| \overline{y}^{(j)}-\overline{y}\right\|=\sqrt{\sum_{k=1}^{m}(\overline{y_k}^{j}-\overline{y_k})^2}$ 表示不同类中心之间的距离，$\left\| y_i^{(j)}-\overline{y}^{(j)}\right\|$ 表示每一类中数据到该类中心的距离，F 统计量遵从自由度为 $r-1$，$n-r$ 的 F 分布。

基于模糊等价关系的聚类算法基本原理是：首先对请求数据构建数据矩阵并利用某种方法对数据矩阵建立模糊相似矩阵，然后对模糊相似矩阵作水平截集，通过改变截距的大小获得不同精度的聚类划分。

3　云仿真调度模型验证与结果分析

CloudSim 是一款可以模拟实际仿真调度的工具，本章将使用 CloudSim 工具对分类负载均衡模型做实验验证。开发人员通过扩展或者修改 CloudSim 中的基础类，来优化或者实现不同的调度算法，比如顺序分配策略和贪心策略，将算法原理加入 DataCenterBroker 类中，然后调度分配算法，实现算法仿真。

用 CloudSim 模拟用户数量为 3，服务器数量为 5 的集群调度过程，如图 3 所示，用户列表为 User1，User2，User3；虚拟机列表为 Vm0，Vm1，Vm2，Vm3，Vm4；请求的任务列表为 Task1，Task2，…Taskn；实际模拟过程中，数据中心并不直接参与调度，而是由数据代理管理和调度任务列表中的请求任务，分发给虚拟机处理，完成数据中心和虚拟机之间的交互。本文的仿真环境设置和实验参数设置表 1 所示。

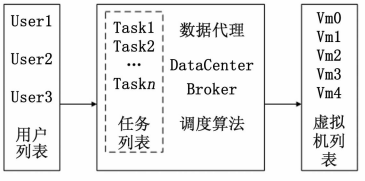

图 3　CloudSim 模拟集群调度过程

在相同实验环境及实验参数下分别运行顺序选择策略、贪心策略和分类负载均衡策略，比较 3 种调度策略的性能优劣。

在 DataCenterBroker 类中加入本文算法：

表 1 实验软硬件仿真环境

	软件环境	硬件环境
软硬件环境	Window 7，Eclipse2015 Cl，jdk1.8.0_101，CloudSim－3.0，ApacheCommons－math3－3.2.jar	内存：8G，硬盘：1000 G，处理器：Inter (R)Core(TM)i5－6500 CPU 3.20GHz
	JDK 配置	CloudSim 配置
环境配置	新建系统变量 JAVA_HOME，变量值：C:\Program Files\java\jdk1.7.0_u45；Path 中加入路径：%JAVA_HOME%\bin；ClassPath 中加入路径%JAVA_HOME%\lib\dt.jar；%JAVA_HOME%\lib\tools.jar	在 ClassPath 中加入路径 C:\cloudsim－3.0\jars\cloudsim－3.0.jar；C:\cloudsim－3.0\jars\cloudsim－examples－3.0.jar
	MBFCM 算法参数配置	任务请求参数配置
实验参数配置	聚类数 c＝3，模糊指数 m＝2，停止阈值 emps＝0.01	任务数量 Task＝20－200；计算资源节点数（虚拟机数）为 5

broker.bindCloudletsToVmsMBFCM－CLBM；//分类负载均衡调度算法

通过逐步增加任务请求个数来验证分类负载均衡调度策略的有效性。现设计 3 种场景对 3 种算法进行仿真调度实验，对比其最大运行时间和资源利用率的变化情况。

1）针对高负载任务的实验仿真，任务的指令长度远大于服务器的处理速度。

对于场景（1），针对只有高负载任务的请求场景，如图 4～5 所示，顺序策略的最大运行时间最长，贪心策略和 CLBM 策略的最大运行时间相接近；顺序选择策略的资源利用率最低，在任务量较小时，贪心策略的资源利用率要高于 CLBM 策略，随着任务量的增加，两者资源利用率逐渐接近。因此，针对高负载任务场景，CLBM 策略的性能和贪心策略的性能基本接近，且远大于顺序选择策略的性能。

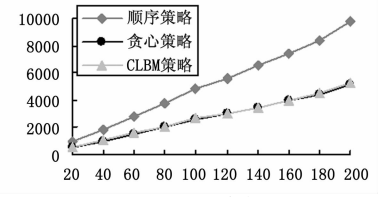

图 4 最大运行时间随请求个数变化图

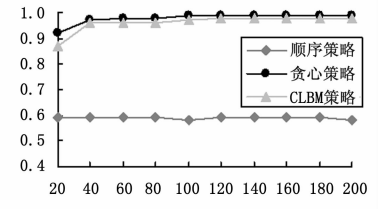

图 5 资源利用率随请求个数变化图

2）针对低负载任务，服务器在单位时间内就可以处理完成，任务的指令长度小于服务器单位时间处理速度。

对于场景（2），针对低负载的任务请求场景，如图 6～7 所示，顺序策略的最大运行时间大于贪心策略最大运行时间，贪心策略最大运行时间稍高于 CLBM 策略的最大运行时间；顺序选择策略的资源利用率最低，随着任务量增加，CLBM 策略的资源利用率远大于贪心策略的资源利用率。说明，针对低负载的请求场景，CLBM 策略的性能高于贪心策略的性能。

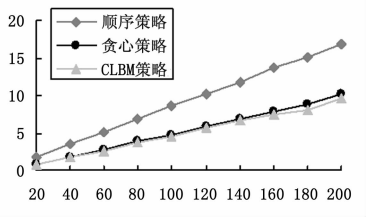

图 6 最大运行时间随请求个数变化图

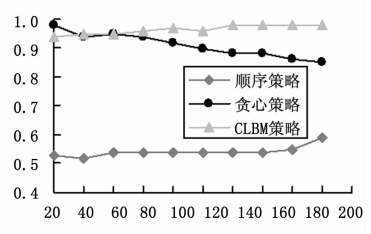

图 7 资源利用率随请求个数变化图

3）针对高负载任务和低负载任务混合调度，其中，高负载任务请求个数：低负载任务请求个数＝1：1 。

对于场景（3），针对既有高负载又有低负载的请求场景，根据图 8～9 所示，随着任务量的增加，顺序选择策略的最大运行时间远大于其他两种策略，贪心策略的最大运行时间稍比 CLBM 策略的最大运行时间稍高；顺序选择策略的资源利用率最低，CLBM 策略的资源利用率高于贪心策略的资源利用率，说明 CLBM 策略的调度性能高于贪心策略的调度性能。

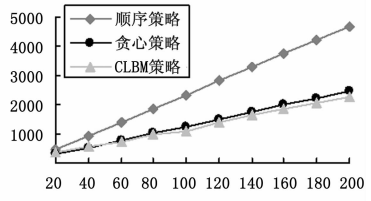

图 8 最大运行时间随请求个数变化图

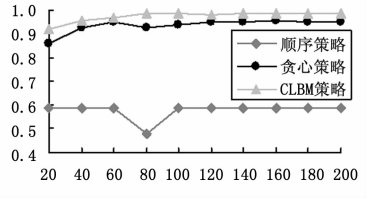

图 9 资源利用率随请求个数变化图

观察并分析上述实验结果：顺序分配策略不考虑任务的大小以及虚拟机的处理能力，仅仅以每个虚拟机上运行任务数量作为衡量标准，为实现负载均衡，尽量满足每个虚拟机上运行任务数量相同，原理简单，易于实现，但实际上不同的任务请求具有不同的负载大小，并且服务器的处理能力也不尽相同，因此顺序选择策略的调度性能很差。

相比于顺序选择策略，贪心策略加入任务的指令长度（MI）和虚拟机的执行速度（MIPS）这两个性能参数，将某任务在某虚拟机上的执行时间（T）定义为指令长度与执行速

度的比值，即 $T=MI/MIPS$[11]，该算法思想为指令长度越大的任务需要执行速度越快的虚拟机处理，其目的是令所有任务的完成时间接近最短。

但是贪心策略仅仅以所有任务的完成时间作为衡量标准，没有考虑资源的使用情况。本文提出的分类负载均衡策略同时考虑执行时间和资源使用情况两大性能，即将任务的完成时间和资源利用率作为衡量调度性能的标准，客观全面地反映调度的性能，算法的最大运行时间最短，资源利用率最高，最高可达 99%。

综上以上分析，理论和实验皆证明本文提出的分类负载均衡调度算法适用性更强，性能更好。

4　结束语

本文基于模糊聚类算法，提出一种分类负载均衡调度模型，分别对用户请求和后台服务器分类。用改进的最小支撑树算法获取初始中心，F 测度确定最佳聚类数，通过 BP 神经网络算法提高 FCM 算法的学习能力，预聚类以及模糊关联聚类算法对资源有效分类。最后 CloudSim 下的仿真环境模拟，在 3 种场景下对比传统算法和本文算法，具有更强的适应性和更好的调度性能。实验表明该模型在作业调度和资源分配上有实际的指导意义。

参考文献：

[1] 李明阳，严　华. 改进粒子群算法在云计算负载均衡中的应用研究 [J]. 计算机测量与控制，2016，24 (10)：219－221.

[2] I Konstantinou，D Tsoumakos，N Koziris. Fast and cost－effective online load balancing in diatributed range－queriable systems [J]. IEEE Transactions on Parallel and Distributed Syst. 2011，22 (8)：1350－1364.

[3] Wang X J，Wang Y，Hao Z，et al. The Research on Resource Scheduling Based on Fuzzy Clustering in Cloud Computing [A]. ICICTA 2015：2015 8th International Conference on Intelligent Computation Technology and Automation [C]. 2015：1025－1028.

[4] 曹鸿强，卢锡城. 多机服务器任务调度的经济学方法 [J]. 计算机工程与科学，2001，23 (2)：4－9.

[5] 李春鸽，刘欣然，张哲宇，等. 虚拟计算环境下一种基于模糊聚类的资源匹配模型 [C]. VARA 2013：The sixth Session of Information Security Vulnerability Analysis and Risk Assessment，2013：365－378.

[6] Wang Y，Wang J K，Wang G R，et al. Utility optimization strategy of resoutce scheduling in cloud computing [A]. CCC 2016：Control Conference [C]. 2016 35th Chinese：5235－5238.

[7] 刘　健，徐　磊，张维明. 基于动态反馈的负载均衡算法 [J]. 计算机工程与科学. 2003，25 (5)：65－68.

[8] 李春生. 模糊聚类的组合方法及其应用研究 [D]. 长春：吉林大学，2010，8.

[9] 李荣荣，孙纪敏. A Fuzzy Clustering Algorithm Based on Complex Synaptic Neural Network [A]. 2017 17th IEEE International Conference on Communication Technology (ICCT 2017) [C]. China，2017，1291－1295.

[10] 曲福恒，崔广才，李岩芳，等. 模糊聚类算法及应用 [M]. 北京：国防工业出版社，2011.

[11] 刘　鹏. 云计算（第二版） [M]. 北京：电子工业出版社，2012.

(上接第 194 页)

图 9 为通信过程中设置的监听节点（monitor）的侦听包情况。由图可知，监听节点侦听数据包数量分别为 10 包和 26 包。由表 2 可知，监听节点截获的数据包数量远远不足以恢复出全部原始信息，同时由于源节点处采用了冗余编码措施，此时截获的数据包只是部分编码包，不具有任何原始数据的直接信息，验证了多路由冗余传输算法的抗截获性能。

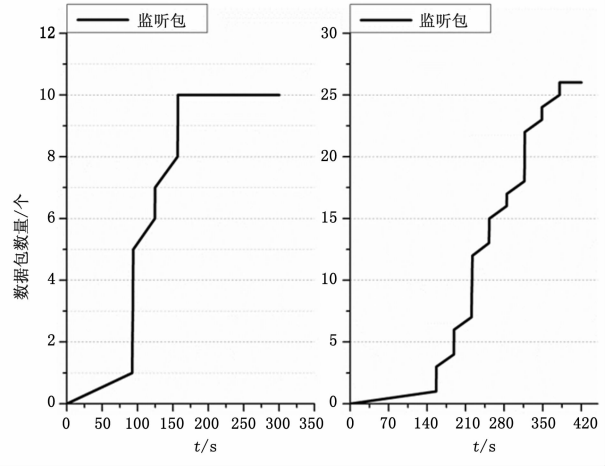

图 9　监听节点侦听包情况

5　结论

本文针对自组网通信系统中抗干扰、抗截获性能需求提出了一种多路由冗余传输算法，该算法有效融合了多径路由协议和冗余编码的特点，具有无需重传的优点，增加了网络数据传输的容错率。经试验证明，与传统的单径路由协议相比，该算法能有效提高网络的抗干扰、抗截获性能，保证通信传输的可靠性。另外，有关冗余度的与网络节点数的关系以及多路由冗余传输算法下 MAC 层协议机制等问题，需要进一步研究分析。

参考文献：

[1] 杨同茂. 军事通信抗干扰技术的发展现状及趋势 [J]. 通信技术，2014，47 (7)：704－712.

[2] 秦　茜，宋志群，刘玉涛. 一种固定分配与动态竞争结合的 MAC 层协议算法 [J]. 无线电工程，2017，47 (2)：15－19.

[3] 王嘉欣. 移动 Ad Hoc 网络抗干扰可靠路由技术 [D]. 成都：电子科技大学，2013.

[4] Yi J，Adnane A，David S，et al. Multipath optimized link state routing for mobile ad hoc networks [J]. Ad Hoc Networks. 2011 (9)：28－47.

[5] 文　静. MP－OLSR 在船舶自组网中的优化研究 [D]. 广州：华南理工大学，2014.

[6] 姜　博，晏　坚，蒋卫东. 喷泉码及其在通信网络中的应用 [J]. 数字通信世界. 2007，10：64－67.

[7] 时琳川，徐松毅，杜文举. 喷泉码在窄带通信中的研究 [J]. 无线电通信技术. 2017，43 (1)：19－22.

[8] 钱晋希. LT 码编译码算法优化及应用研究 [D]. 哈尔滨：哈尔滨工程大学，2013.

[9] Mackay D J C. Fountain Codes [C]. IEEE Proceeding－Communications，2005：1062－1068.

[10] 胡志伟，梁加红，陈　凌，等. 移动自组网仿真技术研究综述 [J]. 系统仿真学报，2011，23 (1)：1－6.