**专利申请信息表**

|  |  |
| --- | --- |
| **请您核实以下申请信息是否正确** | |
| **申　请　人** | 南京赛宁信息技术有限公司  统一社会信用代码：91320115080271299G |
| **发明名称** | 一种网络靶场中计算节点网络关系变更的方法与系统 |
| **专利申请人**  **地址/邮编** | 南京市江宁区秣周东路12号 211100 |
| **发明人名单** | 詹志 |
| **第一发明人**  **身份证号** | 36250219990210463X |
| **联系人**  **联系方式** | 姓名：詹志  电话：13361875998  E-mail：zhanzhi@cyberpeace.cn |
| **邮寄地址** | 南京市江宁区秣周东路12号2号楼16楼 |
| **费减类型** |  按85%比例享受费减  按70%比例享受费减 不享受费减 |
| **专利申请选项** |  提前公开  指定公开时间   申请时请求实质审查  申请时不请求实质审查  基于相同技术方案的发明与实用新型同日申请（同日申请的发明不得提加快审查）   请求实质审查时申请延迟审查（可选1年、2年、3年），选定后无法改变与撤销 |

本发明公开了。

指定附图1为摘要附图

1.一种网络靶场中计算节点网络关系变更的方法，其特征在于，包括如下步骤：

对修改前的拓扑实例中的节点进行排序，并遍历节点集合以生成节点网络关系，包括节点的衍生网络信息和接入网络信息；其中衍生网络是遍历节点及其端口时节点的端口和对端端口均未接入网络时，产生的网络，接入网络是遍历节点及其端口时，对端端口已接入的网络；

比较拓扑实例修改前后节点和节点连线端口的变化，清理无效的节点网络关系；

基于修改后的拓扑实例中的节点端口的网络标识数据，以及修改前的拓扑实例生成的所有衍生网络集合，生成修改后拓扑实例中的节点的衍生网络信息和接入网络信息；

针对修改后的拓扑实例，重新执行节点网络关系生成过程，补充新增的节点网络关系；

通过对比拓扑实例修改前后的所有节点衍生网络集合，确定需要删除和新增的网络；

根据新的节点网络关系，对网络资源以及虚拟机资源进行增量修改。

2、根据权利要求1所述的一种网络靶场中计算节点网络关系变更的方法，其特征在于，所述对修改前的拓扑实例中的节点进行排序，并遍历节点集合以生成节点网络关系，包括：

根据节点类型对节点进行排序，相同类型的节点根据节点标识排序，得到节点集合；节点类型的排序按照交换机、路由器和终端顺序；

维护一个全局的端口和网络映射关系，用于记录每个端口对应的网络信息，遍历节点集合，对于每个节点执行：

初始化一个默认网络，用于XXX；遍历当前节点所有的连线端口，对于每个端口执行：判断端口是否存在于映射关系中：如果存在，则获取当前端口对应的目标网络，如果当前节点的接入网络和衍生网络都不包含所获取的目标网络，则将所获取的目标网络添加到当前节点的接入网络中，否则结束对当前端口的处理；如果不存在，则根据节点类型进行不同处理：对于终端或交换机节点，设置默认网络的节点标识为当前节点的标识，更新端口和网络映射关系，添加当前端口和当前端口所在连线的对端端口到默认网络的映射，并将两个端口加入默认网络的连接端口对；如果当前节点的接入网络和衍生网络都不包含默认网络，则将默认网络添加到当前节点的衍生网络中；对于路由器节点，初始化一个新的网络，用于XXX（默认网络和新的网络的区别在哪里）；设置新的网络的节点标识为当前节点的标识，更新端口和网络映射关系，添加当前端口和当前端口所在连线的对端端口到新的网络的映射，并将两个端口加入新的网络的连接端口对；如果当前节点的接入网络和衍生网络都不包含新的网络，则将新的网络添加到当前节点的衍生网络中。

3、根据权利要求1所述的一种网络靶场中计算节点网络关系变更的方法，其特征在于，所述比较拓扑实例修改前后节点和节点连线端口的变化，清理无效的节点网络关系，包括：

遍历修改后拓扑实例中的所有节点，对于每个节点检查修改前拓扑实例中是否存在相同节点标识的节点；

如果不存在，则清空其所有端口的网络标识信息；

如果存在，则遍历其所有端口；对于每个端口检查修改前拓扑实例中是否存在相同端口标识的端口，如果不存在，则清空当前端口的网络标识信息；如果存在，则分别获取修改前后的对端端口，根据不同的连接状态进行如下处理：

若两个连接端口都为空或者标识相同，则当前端口的网络标识信息不变；

若其中一个连接端口为空，则清空当前端口的网络标识信息；

若两个连接端口标识不同，则清空当前端口的网络标识信息，如果当前节点类型为交换机，遍历该节点下的所有端口，对每个端口及其对端端口，清空它们的网络标识信息。

4、根据权利要求1所述的一种网络靶场中计算节点网络关系变更的方法，其特征在于，所述生成修改后拓扑实例中的节点的衍生网络信息和接入网络信息，包括：

获取修改前的拓扑实例对应的所有衍生网络集合，维护一个全局的端口和网络映射关系，用于记录每个端口对应的网络信息；

对于衍生网络集合中每一个衍生网络，执行：遍历修改后拓扑实例中每一个节点的每一个端口，处理端口的网络标识信息，在网络标识信息不为空且与当前衍生网络的标识相同时，则获取对端端口，根据对端端口情况更新衍生网络的连接端口对以及端口和网络映射关系，在存在对端端口时，还将当前衍生网络加入到对端端口所属节点的接入网络中；

对于衍生网络集合中每一个衍生网络，执行：根据当前衍生网络中记录的节点标识，获取对应的节点信息，如果衍生网络的连接端口对不空，则将当前衍生网络加入到所获取的节点的衍生网络序列中。

5、根据权利要求1所述的一种网络靶场中计算节点网络关系变更的方法，其特征在于，所述通过对比拓扑实例修改前后的所有节点衍生网络集合，确定需要删除和新增的网络，包括：

将修改前的拓扑实例对应的所有衍生网络集合加载到第一集合；

遍历修改后拓扑实例中每一个节点，将每个节点的所有衍生网络添加到第二集合；

遍历第一集合，判断每个衍生网络是否存在于第二集合，若不存在，则标记当前衍生网络为需要删除的网络；

遍历第二集合，判断每个衍生网络是否存在于第一集合，若不存在，则标记当前衍生网络为需要新增的网络。

比较网络集合：遍历旧拓扑和新拓扑的所有节点衍生网络集合，确定需要删除和新增的网络；

删除和新增网络：根据比较结果，删除不需要的网络，新增需要的新网络。

6、根据权利要求1所述的一种网络靶场中计算节点网络关系变更的方法，其特征在于，所述对网络资源以及虚拟机资源进行增量修改，包括：

对于删除的网络，直接删除；

对于新增的节点，基于新的节点网络关系，直接创建；

对于删除的节点，直接删除；

对于网络发生变更的节点，基于修改前的衍生网络信息、接入网络信息以及修改后的衍生网络信息、接入网络信息，变更节点虚拟机的网卡接入信息。

7、根据权利要求1所述的一种网络靶场中计算节点网络关系变更的方法，其特征在于，存储节点信息的数据结构，包括节点标识、节点类型、节点的接入网络信息、衍生网络信息和端口列表；其中端口列表中每个端口包括端口标识和网络标识；节点的接入网络和衍生网络的数据结构相同，包括网络标识、节点标识和连接端口对列表。

8、一种网络靶场中计算节点网络关系变更的方法，其特征在于，包括：

旧拓扑节点网络关系生成模块，用于对修改前的拓扑实例中的节点进行排序，并遍历节点集合以生成节点网络关系，包括节点的衍生网络信息和接入网络信息；所述衍生网络信息XXX，所述接入网络信息XXX；

节点网络关系变更计算模块，用于比较拓扑实例修改前后节点和节点连线端口的变化，清理无效的节点网络关系；基于修改后的拓扑实例中的节点端口的网络标识数据，以及修改前的拓扑实例生成的所有衍生网络集合，生成修改后拓扑实例中的节点的衍生网络信息和接入网络信息；针修改后的拓扑实例，重新执行节点网络关系生成过程，补充新增的节点网络关系；以及通过对比拓扑实例修改前后的所有节点衍生网络集合，确定需要删除和新增的网络；

以及增量修改模块，用于根据新的节点网络关系，对网络资源以及虚拟机资源进行增量修改。

9、一种计算机系统，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，其特征在于，所述计算机程序被处理器执行时实现根据权利要求1-6任一项所述的一种网络靶场中计算节点网络关系变更的方法的步骤。

10、一种计算机程序产品，包括计算机程序，其特征在于，所述计算机程序被处理器执行时实现根据权利要求1-6任一项所述的一种网络靶场中计算节点网络关系变更的方法的步骤。

**一种网络靶场中计算节点网络关系变更的方法与系统**

**技术领域**

本发明涉及一种网络靶场中计算节点网络关系变更的方法与系统，属于计算机软件、网络安全技术领域。

**背景技术**

网络靶场是一种基于虚拟化技术，对真实网络环境的仿真，作为支撑网络空间安全技术验证，网络武器装备试验，攻防对抗演练和网络风险评估的平台。

在将靶场中某个拓扑实例构建完成后，有时还需要对此拓扑进行调整节点和连线关系。通常情况下需要将此拓扑实例关闭，会导致此拓扑实例创建出的虚拟机、网络、路由器等资源全部进行回收后，再编辑拓扑模板，去调整节点和连线关系。然后再根据拓扑模板重新构建生成拓扑实例。

现有的网络关系变更处理方案存在如下问题：1、性能差。若调整一个已经构建完成的拓扑实例，对创建的所有虚机、网络、路由器等资源全部回收，然后再重新完成全局构建，这个过程会消耗大量CPU、内存、磁盘，涉及到大量磁盘IO操作，消耗系统大量资源的同时，也非常耗时。2、拓扑实例修改，如果涉及到重建（回收然后重新创建）虚机，因为重建会回收磁盘资源，那么重建之前产生的虚机数据都会消失。

**发明内容**

发明目的：针对上述现有技术存在的问题，本发明目的在于提一种网络靶场中计算节点网络关系变更的方法与系统，能够计算已构建的拓扑修改后，其节点网络关系的变化，根据计算的节点网络关系变化结果，可以对已构建的拓扑进行增量修改，提升重建性能。

技术方案：为实现上述发明目的，本发明采用如下技术方案：

有益效果：与现有技术相比，本发明提出的方法能够计算已构建的拓扑修改后，其节点网络关系的变化，根据计算的节点网络关系变化结果，可以对已构建的拓扑进行增量修改，而不需要先回收全部资源，然后重新执行全构建流程。增量修改相对于先全局回收然后重新构建优势如下：1、性能极大提高。拓扑的构建涉及到网络、CPU、内存、磁盘的大量操作，支持对已构建的拓扑进行增量修改，可以极大优化修改过程中不必要的网络、CPU、内存、磁盘资源占用和操作。2、增量修改对于拓扑修改前后任然存在的节点，不会回收其磁盘资源，在拓扑修改前产生的业务数据，在修改后可以保留。

**附图说明**

图1为本发明实施例的流程示意图。

图2为本发明实施例中的示例的拓扑变化示意图。

**具体实施方式**

下面将结合附图和具体实施例，对本发明的技术方案进行清楚、完整的描述。

针对上述修改拓扑运行实例存在的问题。本专利提出了一种网络靶场中增量计算网络关系变更的方法和系统。通过此系统可以知道在拓扑实例动态变更前后，节点之间网络关系的变更情况。由此，对于1.网络关系没有变更的节点，不需要处理；2.网络关系产生变更的节点，只需要修改其接入的网络信息，其CPU、内存、磁盘资源都无须额外处理；3.被删除的节点，需要释放其网络、CPU、内存、磁盘资源；4.新增的节点，需要申请网络、CPU、内存、磁盘资源。

为了更好地描述实现过程。以上图中描述的拓扑模板为例。图一左边是最初的拓扑模板，虚拟核心路由器r1通过端口r1-p1和交换机s1端口s1-p2连接，通过端口r1-p2和交换机s2端口s2-p2连接。交换机s1通过端口s1-p1和虚拟机终端t1端口t1-p1连接。交换机s2通过端口s2-p1和虚拟机终端t2端口t2-p1连接，同时通过端口s2-p3和虚拟机终端t3端口t3-p1连接。

经过用户修改后，交换机s2的端口s2-p2连接新的路由器r2的端口r2-p2，终端t3被删除，交换机s2端口s2-p3不连接任何节点了。新路由器r2的端口r2-p1连接着路由器r1端口r1-p3。路由器r1端口r1-p2不连接任何节点了。

下面详细介绍如何增量计算网络关系变更。

1. 涉及的数据结构

1.1节点网络关系数据结构核心内容

{

"network\_id": "2343252345", //网络id

"node\_id": "s1", //衍生网络的节点id

"connect\_ports": [ // 网络连接关系

{

"from": "s1-p1", // 出发端口

"to": "t1-p1" // 目的端口

},

{

"from": "s1-p2", // 出发端口

"to": "r1-p1" // 目的端口

}

]

}

如上结构，为Network节点网络关系的存储结构。network\_id为网络的唯一标识id，使用标准的uuid生成器生成。node\_id为衍生该网络的节点id，本系统将网络和节点之间的关系分为节点衍生网络和节点接入网络。两者的定义见下面步骤2。connect\_ports为当前网络中接入的端口，有多组，一组是一条连线，连线的一端是from出发端口，另外一段是to目的端口。from出发端口都是node\_id对应的节点的端口。

1.2节点信息数据主要内容

{

"node\_id": "s1",

"node\_type": "1", //节点类型分为0-终端 1-交换机 2-路由器

"access\_networks": [

],

"derive\_networks": [

{

"network\_id": "2343252345",

"node\_id": "s1",

"connect\_ports": [

{

"from": "s1-p1",

"to": "t1-p1"

},{

"from": "s1-p2",

"to": "r1-p1"

}

]

}

],

"ports": [

{

"port\_id": "s1\_p1",

"network\_id": "2343252345",

"src\_element": "指向当前节点的引用"

},

{

"port\_id": "s1-p2",

"network\_id": "2343252345",

"src\_element": "指向当前节点的引用"

}

]

}

如上数据结构为节点Node的结构，每个节点均持有derive\_networks节点衍生网络信息和access\_networks节点接入网络信息。这两个字段的类型都是List<Network>。同时节点Node还持有其拥有的端口的信息，相关信息存放在其持有的List<Port> ports中。每个节点的端口数据ports是根据拓扑就确定的，其衍生网络和接入网络数据，是本系统特殊定义的概念，数据计算过程见下面步骤2。

步骤2：拓扑第一次构建时，生成节点网络关系。节点网络关系的生成过程，其实就是每个节点的derive\_networks节点衍生网络信息和access\_networks节点接入网络信息的计算过程。具体步骤包括：

步骤2.1 节点排序，得到排序后的节点的集合List<Node>。计算节点网络关系前，要先根据节点的类型，对节点进行排序。节点类型分为终端、路由器、交换机，类型相同，则根据节点id排序。节点类型中交换机排在最前面，然后是路由器类型，最后是终端类型。

步骤2.2 遍历节点集合List<Node>，生成节点网络关系。具体过程包括：

步骤2.2.1维护一个全局的端口和网络的映射关系Map<Port, Network> portToNetworkMap。

步骤2.2.2遍历节点集合List<Node>，对于其中的每个节点，执行下面步骤。

步骤2.2.3 初始化一个默认网络（初始化一个网络具体是使用标准uuid生成器设置上网络id，其它都设置为空，下同） Network defaultNetwork，处理当前节点，遍历该节点所有的连线端口List<Port> ports，对于其中的每个端口，执行下面步骤。

步骤2.2.4 对于当前端口Port port，执行：

步骤2.2.4.1 判断portToNetworkMap是否包含当前端口port，如果包含，执行步骤2.2.4.1下面步骤。

步骤2.2.4.1.1 使用portToNetworkMap.get(port)获取到当前port对应网络Network target\_network，如果当前节点node的access\_networks和derive\_networks都不包含当前target\_network，则将target\_network加入到当前节点node的access\_networks中，否则当前端口port的处理结束。

步骤2.2.4.2 判断portToNetworkMap是否包含当前端口port，如果不包含，执行步骤2.2.4.2下面步骤。

步骤2.2.4.2.1 如果当前节点类型node\_type为0或者1，即终端类型或者交换机。defaultNetwork的node\_id为当前遍历节点的node\_id。

步骤2.2.4.2.1.1 当前端口port所在连线的另外一侧端口为Port connect\_port，设置portToNetworkMap.put(port, defaultNetwork), portToNetworkMap.put(connect\_port, defaultNetwork)。同时defaultNetwork.getConnectPorts.add(port, connect\_port)。即portToNetworkMap中添加key为port，value为defaultNetwork的映射；添加key为connect\_port，value为defaultNetwork的映射。以及defaultNetwork的connect\_ports添加from为port，to为connect\_port的端口对。同时port和connect\_port的network\_id值都设置为defaultNetwork的network\_id。

步骤2.2.4.2.1.2 如果当前节点node的access\_networks和derive\_networks都不包含defaultNetwork, 则将defaultNetwork加入到当前节点的derive\_networks中。

步骤2.2.4.2.2 如果当前节点类型node\_type为2，路由器。

步骤2.2.4.2.2.1 初始化一个网络Network currentNetwork。currentNetwork的node\_id为当前遍历节点node的node\_id。

步骤2.2.4.2.2.2 当前端口port所在连线的另外一侧端口为Port connect\_port，设置portToNetworkMap.put(port, currentNetwork), portToNetworkMap.put(connect\_port, currentNetwork)。同时defaultNetwork.getConnectPorts.add(port, connect\_port)。即portToNetworkMap中添加key为port，value为currentNetwork的映射；添加key为connect\_port，value为currentNetwork的映射。以及currentNetwork的connect\_ports添加from为port，to为connect\_port的端口对。同时port和connect\_port的network\_id值都设置为currentNetwork的network\_id。

步骤2.2.4.2.2.3 如果当前节点node的access\_networks和derive\_networks都不包含currentNetwork, 则将currentNetwork加入到当前节点的derive\_networks中。

在示例拓扑图中，按照步骤2.1~2.2描述，我们可以得到这样一个节点网络关系：

交换机s1衍生网络derive\_networks中只有一个元素network1，接入网络access\_networks为空。network1的node\_id为s1的node\_id。包含的端口连接对connect\_ports为(s1-p2,r1-p1)、(s1-p1,t1-p1)两个， 连接对第一个元素为from，第二个元素为to。

交换机s2衍生网络derive\_networks中也只有一个元素network2，接入网络access\_networks为空。network2的node\_id为s2的node\_id。包含的端口连接对connect\_ports为(s2-p2,r1-p2)、(s2-p1,t2-p1)、(s2-p3,t3-p1)。

终端t1的衍生网络derive\_networks为空，access\_networks有一个元素为network1。

终端t2的衍生网络derive\_networks为空，access\_networks有一个元素为network2。

终端t3的衍生网络derive\_networks为空，access\_networks有一个元素为network2。

路由器r1的衍生网络derive\_networks为空，access\_networks有两个元素是network1和network2。

步骤2.3得到上述关系后，我们遍历所有节点的derive\_networks，得到所有衍生网络集合List<Network> all\_networks，并将其持久化。同时将节点信息和端口信息持久化。

步骤3：对拓扑实例进行修改后，清理无效的节点网络关系。

此步骤比较拓扑修改前后，节点和节点连线端口的变化，将由于节点网络关系变化导致节点端口中network\_id字段记录的无效网络信息清空。便于步骤4能正确计算出拓扑变更前后有效的节点网络关系。下面详细介绍如何清理无效的节点网络关系。

步骤3.1比较拓扑实例修改前后拓扑，修改前拓扑记为old\_topo，修改后拓扑记为new\_topo。

步骤3.2遍历new\_topo中所有节点，对于其中每个节点new\_topo\_node做如下操作。

步骤3.2.1 当前节点id为new\_topo\_node.node\_id。检查old\_topo中是否存在相同节点id的节点。

步骤3.2.2 如果old\_topo中不存在相同节点id的节点，此new\_topo\_node为新增节点，此节点所有的端口的network\_id信息置空。

步骤3.2.3 如果old\_topo中存在相同节点id的节点，记为old\_topo\_node。遍历new\_topo\_node的所有端口，对于每个端口new\_topo\_node\_port做如下操作

步骤3.2.3.1 当前端口id为new\_topo\_node\_port.port\_id，检查old\_topo\_node中是否存在端口id和new\_topo\_node\_port.port\_id相同。

步骤3.2.3.2 如果不存在相同id的端口，则说明new\_topo\_node\_port为新端口，端口的network\_id信息应该为空。

步骤3.2.3.3如果存在相同id的端口old\_topo\_node\_port，在new\_topo中获取new\_topo\_node\_port所在连线另外一端的端口new\_topo\_node\_connect\_port。在old\_topo中获取old\_topo\_node\_port所在连线的另外一端的端口old\_topo\_node\_connect\_port。

步骤3.2.3.4 如果new\_topo\_node\_connect\_port为空，old\_topo\_node\_connect\_port为空，则说明拓扑修改前后new\_topo\_node\_port对应的端口都是没有接入任何连线的，new\_topo\_node\_port.network\_id信息保持不变。

步骤3.2.3.5如果new\_topo\_node\_connect\_port不为空，old\_topo\_node\_connect\_port为空，则说明拓扑修改前new\_topo\_node\_port端口是没有连线的，修改后增加了这个端口的连线。new\_topo\_node\_port.network\_id置空。

步骤3.2.3.6 如果new\_topo\_node\_connect\_port为空，old\_topo\_node\_connect\_port不为空，则说明拓扑修改前new\_topo\_node\_port端口有连线，修改后删除了连线。将new\_topo\_node\_port.network\_id置空。

步骤3.2.3.7 如果new\_topo\_node\_connect\_port不为空，且old\_topo\_node\_connect\_port也不为空。

步骤3.2.3.7.1 如果new\_topo\_node\_connect\_port.port\_id和old\_topo\_node\_connect\_port.port\_id相同，则说明拓扑修改前后连线没有变化，不需要处理。

步骤3.2.3.7.2 如果new\_topo\_node\_connect\_port.port\_id和old\_topo\_node\_connect\_port.port\_id不同，说明拓扑修改前后new\_topo\_node\_port连接的端口发生了变化，将new\_topo\_node\_port.network\_id置空。

步骤3.2.3.7.2.1如果当前节点类型new\_topo\_node.node\_type为交换机，遍历new\_topo\_node下面所有端口。

步骤3.2.3.7.2.1记当前端口为sw-p，记当前端口所在连线对端端口为sw-connect-p。将sw-p.network\_id置空，同时如果sw-connect-p不为空，将sw-connect-p.network\_id置空。

由此，完成清理无效的节点网络关系。按照步骤3表述规则，示例拓扑图中清理无效节点网络关系情况如下：

终端t1所有的端口连线都没有改变，不做任何处理。

交换机s1所有的端口连线都没有改变，不做任何处理。

路由器r1的端口r1-p1连线没有发生变化，不需要处理；端口r1-p2连线删除，r1-p2.network\_id置空；端口r1-p3连线新增，r1-p3.network\_id应该为空。

路由器r2为新增节点所有端口的network\_id都应该是空。

交换机s2端口s2-p2的连线发生变化，其端口s2-p2、s2-p1以及端口所在连线的对端端口t2-p1、r2-p2的network\_id都需要置空。

终端t2端口t2-p1的network\_id被置空。

终端t3在拓扑修改前后被删除，在步骤3中不涉及处理。

步骤4：计算不变的节点网络关系。

步骤2完成了修改前拓扑的网络关系初始化，用户修改拓扑后，我们需要根据老拓扑的记录的节点网络关系和新拓扑的节点和节点的连线关系变化，增量计算出节点网络关系。步骤3清理了由于拓扑变化而无效的节点网络关系。步骤4开始计算有效的节点网络关系。即根据新拓扑中节点端口的network\_id数据和步骤2生成的all\_networks，加载节点的衍生网络derive\_networks和接入网络access\_networks。因为经过步骤3处理，所以步骤4获得的节点网络关系是拓扑修改前后没有发生变化的节点网络关系。此关系需要经过后面步骤处理新拓扑新增的节点网络关系才能最终获取到新拓扑完整的节点网络关系。

下面详细介绍如何计算不变的有效节点网络关系。

步骤4.1 获取所有衍生网络集合all\_networks。步骤2最后持久化了所有衍生网络集合。这里我们重新将其加载到内存中，获取到上次拓扑构建的所有衍生网络集合all\_networks。

步骤4.2 加载节点衍生网络derive\_networks，使用Map<Port, Network> portToNetworkMap记录节点端口和网络之间的映射关系。执行以下步骤：

步骤4.2.1遍历编辑后拓扑中所有节点，获取节点node\_id到节点信息node的映射管理，存放到Map<String, Node> nodeIdToNodeMap中。

步骤4.2.2遍历all\_networks，对于其中每一个网络network，做如下操作。

步骤4.2.2.1遍历拓扑中所有节点，对于每个节点node做如下操作。

步骤4.2.2.1.1遍历当前节点端口ports，对于每一个端口port，做如下处理。

步骤4.2.2.1.1.1如果port.network\_id为空则略过当前端口处理。

步骤4.2.2.1.1.2 如果port.network\_id不为空，且port.network\_id和network.network\_id不相同，略过当前端口处理。

步骤4.2.2.1.1.3如果port.network\_id不为空，且port.network\_id和network.network\_id相同，获取到当前port所在连线的对端端口connect\_port。

步骤4.2.2.1.1.3.1如果connect\_port为空，则将(port,null)加入到network.connect\_ports中，同时portToNetworkMap添加key为port，value为network的映射关系。

步骤4.2.2.1.1.3.2如果connect\_port不为空。

步骤4.2.2.1.1.3.2.1如果connect\_port.network\_id和network.network\_id相同。

步骤4.2.2.1.1.3.2.1.1将(port,connect\_port)加入到network.connect\_ports中。

步骤4.2.2.1.1.3.2.1.2同时portToNetworkMap添加key为port，value为network的映射关系；添加key为connect\_port，value为network映射。

步骤4.2.2.1.1.3.2.1.3connect\_port.node\_id为连线对端端口归属的节点，通过nodeIdToNodeMap.get(connect\_port.node\_id)获取到对端节点Node connect\_node，将当前网络network加入到对端节点的接入网络中，connect\_node.add(network)。

步骤4.2.2.1.1.3.2如果connect\_port.network\_id和network.network\_id不相同，出现拓扑数据异常，中止当前所有流程。

步骤4.2.2.2衍生当前网络network的节点node\_id为network.node\_id。nodeIdToNodeMap.get(node\_id)获取到对应的节点信息node。

步骤4.2.2.2.1如果当前网络network.connect\_ports不为空，则将当前network加入到节点node的衍生网络序列中，node.derive\_networks.add(network)。

步骤4.2.2.2.2如果当前网络network.connect\_ports为空，则不做任何处理。

至此，完成计算不变的节点网络关系。按照步骤4所述规则，示例拓扑图中计算不变的节点网络关系结果如下：

交换机s1衍生网络derive\_networks中只有一个元素network1，接入网络access\_networks为空。network1的node\_id为s1的node\_id。包含的端口连接对connect\_ports为(s1-p2,r1-p1)、(s1-p1,t1-p1)两个，连接对第一个元素为from，第二个元素为to。

交换机s2衍生网络derive\_networks为空，接入网络access\_networks也为空。

终端t1的衍生网络derive\_networks为空，access\_networks有一个元素为network1。

终端t2的衍生网络derive\_networks为空，access\_networks也为空。

终端t3是拓扑修改后被删除节点，在步骤4无须处理。

路由器r1的衍生网络derive\_networks为空，access\_networks有一个元素是network1。

路由器r2是新节点，其衍生网络derive\_networks和access\_networks都为空。

步骤5：计算新增的节点网络关系。

计算新增的节点网络关系，即是对每个节点重新执行步骤2。步骤3和步骤4已经将拓扑修改后没有变化的节点网络关系计算完毕。每个计算完毕的端口都尤其对应的网络，这个关系记录在步骤4.2的portToNetworkMap中。如果当前节点的某个端口没有在这个Map中获取到映射的网络，则说明这个端口的网络关系需要重新进行计算，因此在portToNetworkMap中没有获取到映射网络的端口，需要重新执行步骤2.2来补充新增的节点网络关系。

步骤5.1针对编辑后的拓扑new\_topo执行步骤2.1对其节点进行排序。遍历排序后节点，记当前节点为new\_topo\_node。

步骤5.2 对当前的每个节点new\_topo\_node执行上面步骤2.2下面的所有步骤，其中步骤2.2.1声明的portToNetworkMap值为步骤4.2的portToNetworkMap的值。

按照步骤5.1~5.2所述规则，示例拓扑图中新拓扑最终的节点网络关系如下：

交换机s1衍生网络derive\_networks中只有一个元素network1，接入网络access\_networks为空。network1的node\_id为s1的node\_id。包含的端口连接对connect\_ports为(s1-p2,r1-p1)、(s1-p1,t1-p1)两个，连接对第一个元素为from，第二个元素为to。

交换机s2衍生网络derive\_networks中有一个元素network3，接入网络access\_networks为空。network3的node\_id为s2的node\_id。包含的端口连接对connect\_ports为(s2-p1,t2-p1)、(s2-p2,r2-p2)。

终端t1的衍生网络derive\_networks为空，access\_networks有一个元素为network1。

终端t2的衍生网络derive\_networks为空，access\_networks有一个元素为network3。

终端t3是拓扑修改后被删除节点，不在编辑后拓扑new\_topo中，在步骤5无须处理。

路由器r1的衍生网络derive\_networks有一个元素为network4。access\_networks有一个元素是network1。network4的node\_id为r1的node\_id。包含的端口连接对connect\_ports为(r1-p3,r2-p1)一个。

路由器r2是新节点，衍生网络derive\_networks为空，access\_networks有两个元素为network3、network4。

步骤6：计算需要删除和新增的网络。

记拓扑被修改前构建时所有节点的衍生网络的集合为A，记拓扑被修改后，经过步骤3~5处理，新拓扑所有节点的衍生网络的集合为B。需要删除的网络即为A-B，新增的网络即为B-A。具体执行步骤如下：

步骤6.1步骤2.4持久化了拓扑被修改前构建时所有节点的衍生网络的集合List<Network> allNetworks。

步骤6.2 取得执行完步骤5后的编辑后的拓扑new\_topo。用List<Network> allNetworksForNewTopo记录新拓扑的所有衍生网络。

步骤6.3遍历new\_topo所有节点，对于每个节点current\_node。

步骤6.3.1 将current\_node的衍生网络添加进allNetworksForNewTopo。allNetwoksForNewTopo.addAll(current\_node.derive\_networks)。

步骤6.4遍历allNetworks，对于allNetworks下面的元素network执行下面操作。

步骤6.4.1如果network不存在于allNetworksForNewTopo则network是需要被删除的网络。

步骤6.5遍历allNetworkForNewTopo，对于allNetworkForNewTopo下面元素new\_topo\_network执行下面操作

步骤6.5.1如果new\_topo\_network不存在于allNetworks则new\_topo\_network是需要新增的网络。

按照步骤6所述规则，示例拓扑图中，需要被删除的网络为network2，新增网络为network3、network4。

步骤7：根据计算出的新的节点网络关系，操作新增删除变更网络、虚拟机资源。

经过上述步骤1~6，我们可以得到编辑后拓扑的节点网络关系。对于新增的网络，直接创建即可。对于删除的网络，直接删除即可。对于新增的节点，已经生成了其新的节点网络关系，直接创建即可。对于删除的节点，直接删除即可。对于网络发生变更的节点，拥有其修改前的衍生网络、接入网络关系和其修改后的衍生网络、接入网络关系，直接根据两者变化，变更此节点虚拟机的网卡接入的网络即可。

具体按示例拓扑图展示的拓扑修改情况，需要执行如下操作:

步骤7.1 删除网络network2。

步骤7.2 新增网络network3、network4。

步骤7.3 终端t3拓扑修改后节点被删除，虚拟机资源进行回收。

步骤7.4 终端t1拓扑修改后节点存在，且修改前衍生网络derive\_networks为空，access\_networks有一个元素为network1，修改后衍生网络derive\_networks为空，access\_networks有一个元素为network1。其拓扑修改前后其衍生网络和接入网络没有任何变化，因此无须任何处理。

步骤7.5 交换机s1拓扑修改后节点存在，且修改前衍生网络derive\_networks中只有一个元素network1，接入网络access\_networks为空，修改后衍生网络derive\_networks中只有一个元素network1，接入网络access\_networks为空。其拓扑修改前后其衍生网络和接入网络没有任何变化，对应的网络资源network1不需要任何操作。

步骤7.6 路由器r1拓扑修改前衍生网络derive\_networks为空，access\_networks有两个元素是network1和network2。根据network1.connect\_ports，r1的端口r1-p1对应网卡接入网络network1，根据network2.connect\_ports，r1的端口r1-p2对应网卡接入网络network2。拓扑修改后，路由器r1的衍生网络derive\_networks有一个元素为network4。access\_networks有一个元素是network1。端口r1-p2对应的网卡不再接入网络network2；端口r1-p3对应网卡接入网络network4；端口r1-p1对应网卡接入网卡不需要变更。

步骤7.7 路由器r2是新增节点，其衍生网络derive\_networks为空，access\_networks有两个元素为network3、network4。端口r2-p1对应网卡接入网络network4；端口r2-p2对应网卡接入网络network3。

步骤7.8 交换机s2拓扑修改前衍生网络derive\_networks中只有一个元素network2，接入网络access\_networks为空。拓扑修改后衍生网络derive\_networks中只有一个元素network3，接入网络access\_networks为空。其拓扑修改前后衍生网络由network2变更到network3。

步骤7.9 终端t2拓扑修改前衍生网络derive\_networks为空，access\_networks有一个元素为network2。拓扑修改后衍生网络derive\_networks为空，access\_networks有一个元素为network3。端口t2-p1对应网卡接入网络从network2变更到network3。

基于本发明实施例的方法，能够增量计算拓扑变更前后的网络关系变更情况，对于网络接入关系没有发生变化的虚拟节点，在拓扑实例进行动态调整的过程中，不需要消耗任何资源；对于网络接入关系即使发生变化的虚拟节点，在拓扑实例进行动态调整的过程中，也只是调整网络接入关系，分配给虚拟节点的CPU、内存、磁盘等资源也不需要进行回收，只需要调整接入的网络。由此可以极大优化修改拓扑实例的性能。通过增量计算网络关系变更，只有拓扑实例变更前后，被删除的虚拟才会涉及到磁盘资源回收，其它虚机磁盘数据都不影响。

最后我画个流程图

图1

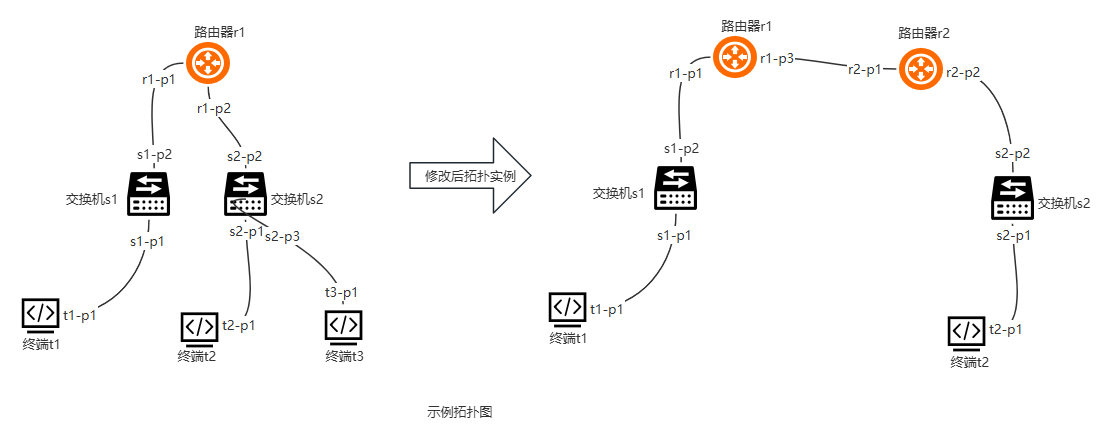


图2