使用 P4 在软件定义网络上实现多个路由配置

Kouji Hirata和 Takuji Tachibana †
* 日本大阪关西大学工程科学学院 E-mail: hirata@kansai-u.ac.jp †日本福井大学工程研究生院 E-mail: takuji- t@u-fukui.ac.jp

摘要:为了保持通信网络的高可用性,我们应该在故障发生后立即恢复故障。作为实现这种快速故障恢复的技术之一,已经提出了多路由配置(MRC)。在 MRC 中,预先准备了多个备份路由配置,以快速修复单个链路/节点故障。

当在使用正常路由配置的数据传输过程中发生故障时,MRC 将路由配置更改为另一个不使用故障点的路由配置。因此,MRC 可以在几十毫秒内实现快速恢复并继续数据传输。在本文中,我们使用 P4(独立于编程协议的数据包处理器)在软件定义的网络上实现 MRC。 P4 是一种编程语言,它使我们能够定义网络设备数据平面的行为。我们使用 Mininet 检查 MRC 的实现。

一、引言

最近,软件定义网络(SDN)由于其灵活的特性而被广泛使用。 SDN 使控制网络变得更加灵活和高效[5]。

SDN的特点是将负责路由控制功能的控制平面与负责包转发功能的数据平面分离。这些平面之间的通信由诸如 OpenFlow [2] 之类的通信协议进行。作为控制平面,使用了一个名为 SDN 控制器的集中式控制器。它将网络管理策略转化为数据包转发规则,称为流表项,并安装在SDN交换机等网络设备(即数据平面)的流表上。网络设备根据其流表转发数据包。每个流条目由匹配字段、动作和计数器组成。匹配字段用于根据数据包头中的元素(如 IP 地址、MAC 地址和端口号)来识别数据包流。例如,在 OpenFlow 1.4 中,匹配字段可以使用 41 个元素来标识数据包流。我们可以通过调整匹配字段来改变流的粒度。这些动作定义了交换机如何处理流程,例如转发和丢弃。计数器记录了到达包数等统计信息。

在传统的 SDN 环境中,我们可以修改控制平面的行为,但不能修改数据平面的行为。具体来说,路由策略可以灵活定义,但不能灵活地做出相应的动作。为了解决这个问题,一种用于数据平面的编程语言命名为 P4(独立于编程协议的数据包处理器)已被提出[3]。 P4 可以表示数据平面(即 SDN 交换机)如何处理传入的数据包,但它不指定由 SDN 控制器管理的控制平面的行为。由于 P4 是一种开源且经过许可许可的语言,因此目前被广泛使用。在本文中,我们在具有 P4 的 SDN 环境中实现了多路由配置 (MRC) [4],它可以从单个节点/链路故障中快速恢复。

为了保持通信网络的高可用性,我们应该在故障发生后立即恢复故障。 MRC 是实现这种快速故障恢复的技术之一。在 MRC 中,预先准备了多个备份路由配置。当使用正常路由配置的数据传输过程中发生单个节点/链路故障时,MRC立即将路由配置更改为不使用故障点的另一个路由配置,从而继续数据传输。 MRC 可以在几十毫秒内实现快速恢复。

此外,MRC 的路由控制在故障恢复时是稳定的,因为每个节点都有共同的路由配置。在本文中,我们使用 Mininet [1] 来检查 MRC 的实现,Mininet [1] 是一个创建 SDN 环境的模拟器。

本文的其余部分安排如下。第二节简要解释了 MRC。在第三节中,我们讨论了 MRC 与 P4 的实现。此外,我们使用 Mininet 检查了 MRC 实现的行为。我们在第四节陈述了本文的结论。

二、多种路由配置

MRC提前准备K(K>1)个备份路由配置,实现快速故障恢复。即使在正常路由配置上发生链路/节点故障时,MRC 也可以通过使用不使用故障点的备份路由配置来继续数据传输。在K个备份路由配置中,每个节点分为正常节点和孤立节点。此外,每个链路被分类为正常链路、孤立链路和受限链路。正常链路/节点表示发生故障时可以直接用于数据传输的链路/节点。隔离的链路/节点表示发生故障时不能用于数据传输的链路/节点。受限链接代表一个链接

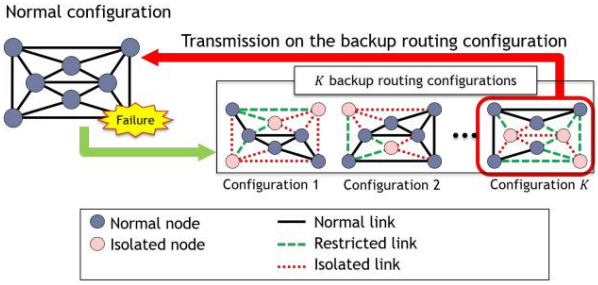

图 1. 路由配置。

仅用于数据传输的第一跳或最后一跳,考虑到连接到受限链路的孤立节点不发生故障的可能性。

图 1 显示了一个正常配置和 K 个备份配置从正常配置生成的示例。在 MRC 中,一个正常配置和 K 个备份路由配置应满足以下条件。

1) 各个节点和链路在至少一个备份路由配置中成为孤立节点和孤立链路。

2)在每个备份路由配置中,都存在一个由所有正常节点和正常链路组成的连通图。

3) 受限链路或隔离链路连接到隔离节点,而正常链路不连接到隔离节点。

4) 连接到孤立节点的至少一条链路是受限链路。

5)在两个节点通过受限链路连接的情况下,一个是正常节点,另一个是孤立节点。

6) 在两个节点通过隔离连接的情况下
链接,一个是孤立节点。

通过使用 K 个备份路由配置之一,MRC 保持高可用性。具体来说,当单个节点发生故障时,MRC会选择该节点为孤立节点的备份路由配置。类似地,当单个链路发生故障时,MRC 会选择一个备份路由配置,其中该链路是隔离链路。请注意,在选定的

备份路由配置,在某些情况下,非故障链路/节点被视为孤立的链路/节点。例如,在图1中,当正常配置发生单节点故障,使用备份路由配置1时,2个孤立节点和7个孤立链路均无故障。

这些没有故障的孤立节点可以使用受限链路传输和接收数据。因此,MRC 可以在不识别节点故障或链路故障的情况下继续数据传输。

图 2 显示了发生单个链路/节点故障时的 MRC 过程。当数据传输过程中路由路径发生故障时,故障点的前一个节点检测到故障。节点选择适当的备份路由配置,其中故障点被选为隔离

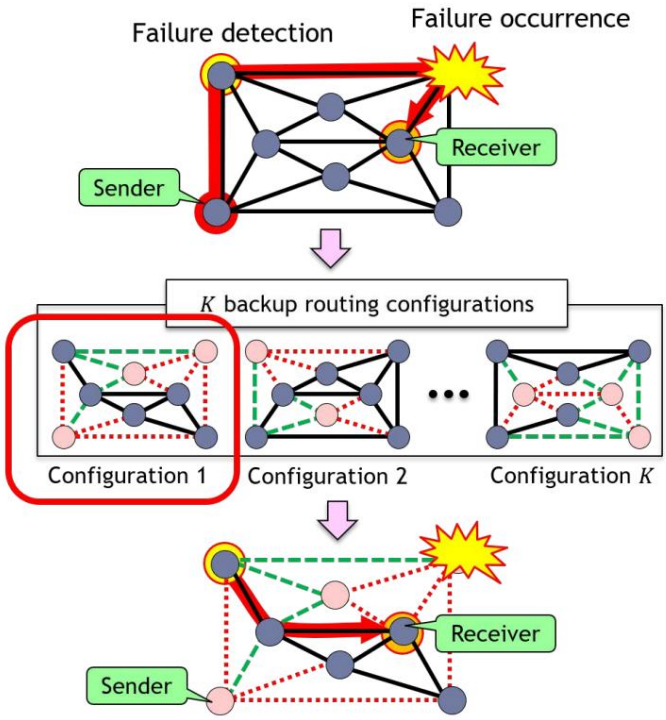

图 2. 故障恢复程序。

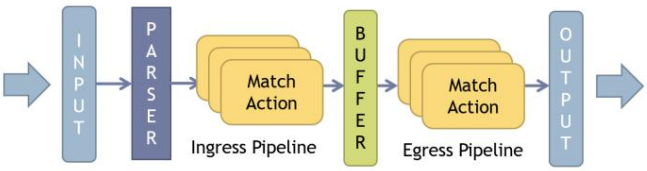

图 3 SDN 交换机中的数据包转发流程。

链接/节点。然后该节点使用选定的备份路由配置继续数据传输。此外,该节点将有关所选备份路由配置的信息放入数据包中,并将该信息通知其他节点。因此,MRC 仅通过简单的路由修改即可实现快速故障恢复。

三、多路由的实现
配置

A. P4的概念

P4 使用匹配动作管道的概念。 SDN 交换机中的数据包转发是通过查表和相应的操作来完成的。图 3 显示了 SDN 交换机中数据包转发的过程。传入的数据包首先由解析器处理。解析器仅处理数据包头,并且假设数据包中的数据被缓冲。解析器根据编程的解析图识别并从标题中提取字段。

然后将提取的报头字段传递到入口匹配+动作表,该表由查找键(例如,IP 地址和 MAC 地址)、相应的动作(即,转发和丢弃)和参数组成。比赛+行动表

```
标头 ipv4_t { bit<4> 版本;位
        <4> ihl;位<8> 差异服务;位<16> 总
        长度;

        ...
        ...
}
```

图 4. IP 包头。

```
状态 parse_ipv4
        { packet.extract(hdr.ipv4);过渡接受;

}
```

图 5. 解析器中的标头提取。

根据查找键和动作处理数据包头,我们可以通过编程修改。此外,它还确定了一个出口端口和一个放置数据包的队列。然后数据包被传递到出口匹配+动作表。我们还可以定义出口匹配+动作表。

最后,数据包被转发到输出端口。

B. 假设

在本文中,我们在 OpenFlow 和 P4 环境中实现 MRC,旨在执行快速恢复,在发生单个链路/节点故障时选择备份配置。

本文着重于进行备份路由配置,并确认我们可以将 MRC 与 P4 一起使用。请注意,我们不考虑故障检测。我们将故障检测留作未来的工作。

我们在以下假设下检查 MRC 的行为。我们准备了一个正常的路由配置和一些备份的路由配置。备份路由配置的使用以离线方式确定。

具体地,每个节点都有备份路由配置,并为其设置节点间通用的标识号(即,正常路由配置为0,备份路由配置为1～K)。每个发送者将一个标识号放入数据包中,以便使用相应的路由配置。每个节点根据数据包中的信息选择路由配置。

在本文中,我们通过根据包头中的服务类型(TOS)字段的值选择路由配置来实现MRC,假设每个发送者在故障发生后立即检测到故障发生并改变TOS的值字段根据故障点。下面,我们将重点介绍 MRC 在 SDN 交换机中的实现。

C. 实施

我们首先定义 IP 包头,如图 4 所示。
TOS 字段是标头定义中的“bit <8> diffserv”。
我们根据故障更改此值以使用备份路由配置。当一个数据包到达一个

```
表 ipv4_lpm { key =
        { hdr.ipv4.dstAddr:
                lpm;

        } 行动 =
                { ipv4_forward;降低;

                无动作;

        } 大小 = 1024;
        default_action = drop();
}
```

图 6. 匹配+动作表。

```
动作 drop() { mark_to_drop();

}

操作 ipv4_forward(macAddr_t dstAddr,egressSpec_t 端口)
                                {standard_metadata.egress_spec
        = 端口; hdr.ethernet.srcAddr =


                                hdr.ethernet.dstAddr;
        hdr.ethernet.dstAddr = dstAddr; hdr.ipv4.ttl =
        hdr.ipv4.ttl - 1;
}
```

图 7. 动作定义。

SDN交换机,解析器提取数据包的IP头,在P4程序中描述如图5所示。提取的头根据头中TOS字段的值传递给入口匹配+动作表.

图6表示P4程序中描述的入口匹配+动作表的定义。该表由键和相应的操作组成。在这种情况下,关键是目标 IP 地址,操作是转发和丢弃。

动作的定义如图 7 所示。丢弃动作丢弃与流表不匹配的传入数据包。

另一方面,转发动作重写包头,例如MAC地址和TTL的值,然后将包传递到出口+匹配动作表。

在我们的实现中,我们准备了一些 match+action 表,并根据 TOS 字段的值选择一个。
该实现如图 8 所示,其中有三个匹配+动作表:ipv4 lpm、ipv4 lpm2 和 ipv4 lpm3。在本例中,如果 TOS 字段的值等于 0,则选择表 ipv4 lpm。此外,如果 TOS 字段的值等于 1,则选择表 ipv4 lpm2;否则,选择表 ipv4 lpm3。请注意,表 ipv4 lpm2 和 ipv4 lpm3 以及图 6 中所示的表 ipv4 lpm 都已定义。在实现中,我们不准备出口匹配+动作表。

在入口和出口处理中,数据包是根据流表进行处理的,流表的一部分如图9所示。流表决定了对数据包应用哪个动作以及将数据包发送到哪个出口端口。为了

```
应用 { if
        (hdr.ipv4.isValid()) { if(hdr.ipv4.diffserv == 0)
                { ipv4_lpm.apply();


                } 否则 if(hdr.ipv4.diffserv == 4){ ipv4_lpm2.apply();


                } 其他
                    { ipv4_lpm3.apply();
                }
        }
}
```

图 8. 基于 TOS 字段的表格选择。

```
{
        “表”：“MyIngress.ipv4_lpm”，“匹配”：
        { “hdr.ipv4.dstAddr”：[ “10.0.1.1”，32]


        },
        “action_name”：
                “MyIngress.ipv4_forward”，
        “action_params”：{ “dstAddr”：“00:00:00:00:01:01”，
                “端口”：1

        }
}
```

图 9. 流表。

每个 SDN 交换机,我们安装一个流表,其中包含每个匹配+动作表的条目（即，ipv4 lpm、ipv4 lpm2 和 ipv4 lpm3）。 – –

–

D. 评估

为了确认 MRC 的实现,我们使用 Mininet [1] 进行了实际实验。图 10 显示了本实验中使用的路由配置。每条链路的带宽等于 1 Mbps。我们准备了一个正常的路由配置（A)和两个备份路由配置（B)和（C）。在每个路由配置中,三个 UDP 流（节点 1 到节点 6、节点 2 到节点 4、节点 3 到节点 5)由 Iperf 传输 180 秒。每个发送者根据路由配置设置 TOS 字段的值。

具体来说,当我们使用路由配置 (A)、(B) 和 (C) 时,每个发送方的 TOS 字段的值分别设置为 0、4 和 8。在这种情况下,我们使用备份路由配置 (B) (resp. (C)) 来应对节点 2 和 5（分别是节点 2 和 3)之间的链路故障。因此,节点 2 和 3 可以使用受限链路传输数据包。

表 I 显示了从实验中获得的每个流的吞吐量。从该表中可以看出,在路由配置 (A) 中,流 2 和 3 共享链路 2-5。因此,流 2 和流 3 的吞吐量小于流 1 的吞吐量。由于在备份路由配置 (B) 中所有流共享链路 3-6,它们具有相似的吞吐量。在备份路由配置（C）中,流3的吞吐量

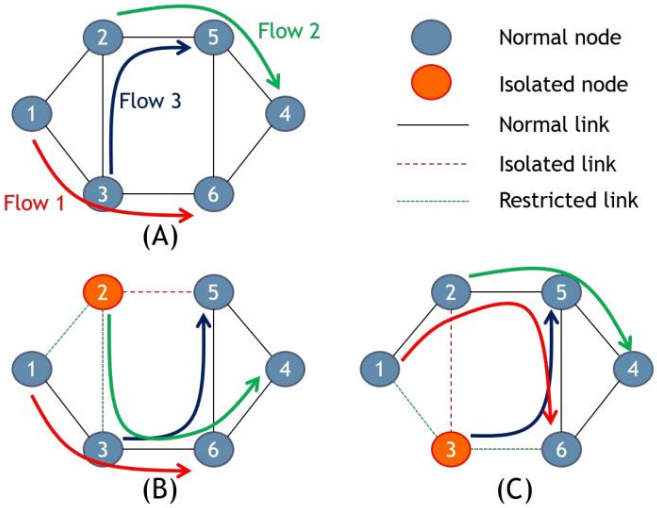

图 10. 模型。

表一
吞吐量评估。

配置	流程 1	流程 2	流程 3	全部的
(一个)	880 kbps	500 kbps	475 kbps	1,855 kbps 320 kbps
(乙)	318 kbps	338 kbps	976 kbps	477 kbps 503 kbps
(C)	965 kbps	1,945 kbps		

是最大的,因为流 1 和 2 共享链路 2-5。这些观察表明 MRC 在实验中运行良好。

此外,在实验中,我们确认这些流正确地使用了基于传输数据包中 TOS 字段值的路由配置。

四。结论

在本文中,我们使用 P4 在软件定义网络上实现了 MRC。我们使用 Mininet 检查了 MRC 的实现。作为未来的工作,我们将实现故障检测机制和根据故障检测重写 TOS 字段值的机制。

致谢

这项研究得到了日本总务省 SCOPE 的部分支持,资助号为 191605004。

参考

[1] Mininet:http://mininet.org/ [2]
开放网络基金会，
https://www.opennetworking.org
[3] P. Bosshart 等人，“P4:独立于编程协议的数据包处理器”,ACM SIGCOMM 计算机通信评论,第一卷。 44,没有。 3,页。 88-95,2014。

[4] A. Kvalbein、A. Hansen、T. Cicic、S. Gjessing 和 O. Lysne，“用于快速 IP 网络恢复的多路由配置”,IEEE/ACM Transactions on Networking,第一卷。 17,没有。 2,第 473-486 页,2009 年。
[5] B. Nunes,M. Mendonca,X.-N。 Nguyen、K. Obraczka 和 T. Turletti，“软件定义网络调查:可编程网络的过去、现在和未来”,IEEE 通信调查与教程,卷。 16,没有。 3,第 1617-1634 页,2014 年。