(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107634895 B
(45)授权公告日 2020.09.22

(21)申请号 201610570067.8
(22)申请日 2016.07.19
(65)同一申请的已公布的文献号
申请公布号 CN 107634895 A
(43)申请公布日 2018.01.26
(73)专利权人 上海诺基亚贝尔股份有限公司
地址 201206 上海市浦东金桥宁桥路388号
专利权人 阿尔卡特朗讯公司
(72)发明人 朱杰 魏伟
(74)专利代理机构 北京市中咨律师事务所 11247
代理人 谭营营 杨晓光
(51)Int.Cl.
H04L 12/58(2006.01)
H04L 29/08(2006.01)

(56)对比文件
US 2015237502 A1,2015.08.20
CN 103023702 A,2013.04.03
CN 104486371 A,2015.04.01
CN 103997495 A,2014.08.20

审查员 文庆

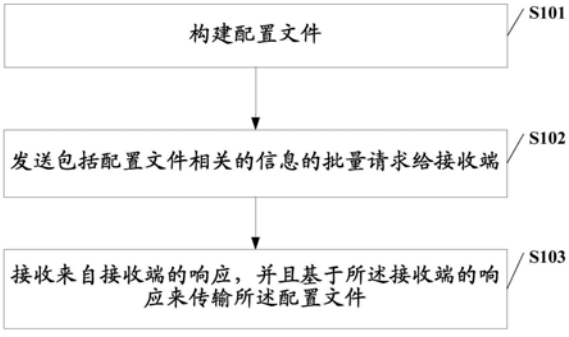
权利要求书2页 说明书13页 附图1页

(54)发明名称

用于基于文件或单个消息的批量操作处理方法和设备

(57)摘要

本发明提供一种用于基于文件或单个消息的批量操作处理方法，该方法包括以下步骤：构建配置文件，该配置文件至少包括头部信息和内容部分，所述头部信息至少包括片段索引，该片段索引至少包括每个要发送的命令的长度信息，所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件；发送包括配置文件相关的信息的批量请求给接收端；接收来自接收端的响应，并且基于所述接收端的响应来传输所述配置文件。



CN 107634895 B

1.一种用于基于单个文件或单个消息的批量操作处理方法,该方法包括以下步骤:

构建配置文件,该配置文件至少包括头部信息和内容部分,所述头部信息至少包括片段索引,该片段索引至少包括多个要发送的命令中的每个要发送的命令的长度信息,所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件;

发送包括配置文件相关的信息的批量请求给接收端,其中,所述批量请求封装与所述多个要发送的命令相关联的多个请求;

接收来自接收端的响应,并且基于所述接收端的响应来传输所述配置文件。

2.根据权利要求1所述的方法,其中所述配置文件还包括表明所述片段索引是否能包括所有命令的长度信息的字段,并且所述方法还包括:如果所述片段索引不能包括所有命令的长度信息,则增加片段索引以便包括所有命令的长度信息。

3.根据权利要求1所述的方法,其中所述批量请求被合并到一个或多个请求消息中,和/或批量响应被合并到一个或多个响应消息中。

4.根据权利要求3所述的方法,其中如果合并后的请求消息和/或合并后的响应消息的长度超过预定长度,则所述请求消息和/或所述响应消息被增加。

5.根据权利要求1-4中任一项所述的方法,其中所述批量请求包括要发送的配置文件的格式,所述响应包括所述接收端是否支持所述配置文件的格式的决定。

6.根据权利要求1-4中任一项所述的方法,其中在所述接收端不支持所述配置文件的格式的情况下,所述响应还包括所述接收端偏好的文件格式,并且所述方法还包括:与接收端进一步协商传输所述配置文件要使用的格式。

7.一种用于基于单个文件或单个消息的批量操作处理的设备,该设备包括:

配置装置,用于构建配置文件,该配置文件至少包括头部信息和内容部分,所述头部信息至少包括片段索引,该片段索引至少包括多个要发送的命令中的每个要发送的命令的长度信息,所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件;

发送装置,用于发送包括配置文件相关的信息的批量请求给接收端,其中,所述批量请求封装与所述多个要发送的命令相关联的多个请求;

接收装置,用于接收来自接收端的响应;并且

所述发送装置还用于基于所述接收端的响应来传输所述配置文件。

8.根据权利要求7所述的设备,其中所述配置文件还包括表明所述片段索引是否能包括所有命令的长度信息的字段,并且所述配置装置还用于:如果所述片段索引不能包括所有命令的长度信息,则增加片段索引以便包括所有命令的长度信息。

9.根据权利要求7所述的设备,其中所述批量请求被合并到一个或多个请求消息中,和/或批量响应被合并到一个或多个响应消息中。

10.根据权利要求9所述的设备,其中如果合并后的请求消息和/或合并后的响应消息的长度超过预定长度,则所述请求消息和/或所述响应消息被增加。

11.根据权利要求7-10中任一项所述的设备,其中所述批量请求包括要发送的配置文件的格式,所述响应包括所述接收端是否支持所述配置文件的格式的决定。

12.根据权利要求7-10中任一项所述的设备,其中在所述接收端不支持所述配置文件的格式的情况下,所述响应还包括所述接收端偏好的文件格式,所述发送装置还用于与接

收端进一步协商传输所述配置文件要使用的格式。

13.根据权利要求7-10中任一项所述的设备,其中所述设备为控制器或交换机。

用于基于文件或单个消息的批量操作处理方法和设备

技术领域

[0001] 本发明涉及通信领域，具体涉及一种用于基于文件或单个消息的批量操作处理方法和设备。

背景技术

[0002] 当前，交换机和处理器针对基于软件定义网络(Software Defined Network，SDN)的开放流(openflow)具有若干种消息处理逻辑，但是它们不能用于解决批量数据传输或业务开销问题。

[0003] 以下是当前规范(2015年3月26日最后更新的1.5.1版)中用于消息处理的现有行为的概要。

[0004] 1、单个原子消息

[0005] 这是用于交换机和控制器的基本工作模式的消息，每个消息包括仅一个请求或回复，这种模式简单明白，因此不再赘述。

[0006] 2、多部件(multipart)消息

[0007] 有时单个消息不能容纳较大的数据包，通常大于64KB，消息被分成小的片段。多部件消息用于一次传递每个小的部分，并且具有额外的标志来指示哪个小的部分是完成发送或接收较大分组前的最后一个消息。这种分离的处理方法通过用额外的标志分离和汇聚多个消息来识别出这种情况并解决这个问题。

[0008] 3、屏障(barrier)消息

[0009] 专用于确保请求是按顺序进行的，屏障请求之前的所有请求都应当在发送屏障响应之前进行，其通常用于清除交换机上的所有现有请求。然后在发送屏障回复之后开始新的请求。

[0010] 4、绑定(bundle)消息

[0011] 用于将多个请求作为一个事务处理来执行，所有请求被成功执行，否则不执行任何请求，其用于将一组请求绑定为单个动作。

[0012] 5、其他

[0013] 基于互联网研究，已经有一些建议或专利来将分组压缩成数据包后特定的编码/解码方法来用于交换消息。

[0014] 根据基于当前规范和研究的以上概要，没有特定的请求类型或逻辑来在短时间处理批量请求和响应消息。

发明内容

[0015] 针对以上问题，本发明提供用于基于文件或单个消息的批量操作处理方法和设备，以便实现在短时间内处理批量请求和响应消息。

[0016] 根据本发明的一个方面，提供一种用于基于文件或单个消息的批量操作处理方法，该方法包括以下步骤：

[0017] 构建配置文件，该配置文件至少包括头部信息和内容部分，所述头部信息至少包括片段索引，该片段索引至少包括每个要发送的命令的长度信息，所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件；

[0018] 发送包括所述配置文件相关的信息(例如文件名字、格式、长度等等)的批量请求给接收端；

[0019] 接收来自接收端的响应，并且基于所述接收端的响应来传输所述配置文件。

[0020] 根据本发明的另一方面，提供一种用于基于文件或单个消息的批量操作处理的设备，该设备包括：

[0021] 配置装置，用于构建配置文件，该配置文件至少包括头部信息和内容部分，所述头部信息至少包括片段索引，该片段索引至少包括每个要发送的命令的长度信息，所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件；

[0022] 发送装置，用于发送包括配置文件相关的信息的批量请求给接收端；

[0023] 接收装置，用于接收来自接收端的响应；并且

[0024] 所述发送装置还用于基于所述接收端的响应来传输所述配置文件。

[0025] 由此，本发明提出了针对基于SDN网络的开放流发出批量请求或响应的模式，使用批量请求或响应的一个单个消息或单个文件可以减轻网络业务开销，并且改进交互效率，特别是针对大量配置请求或查询响应、或小的交互消息(例如交换机配置、异步事件配置等)。

附图说明

[0026] 通过阅读下面结合附图对本发明具体实施例的说明，本发明的上述及其他特征和优点将变得更加明显。其中：

[0027] 图1显示了根据本发明提供的用于基于文件或单个消息的批量操作处理方法的一种实施方式的流程图；以及

[0028] 图2显示了根据本发明提供的用于基于文件或单个消息的批量操作处理的设备的一种实施方式的结构框图。

具体实施方式

[0029] 本发明的实施例将会参照附图进行详细的描述。说明书中涉及的特征、优点、或者类似语言并非暗示与本发明一起实现的所有特征以及优点均应出现在本发明的一个实施例中。相反，涉及特征以及优点的语言被理解为表示与实施例相关描述的特定特征、优点、或者特性被包含在本发明的至少一个实施例中。进一步地，本发明描述的特征、优点以及特性可以任意合适方式合并在一个或者多个实施例中。相关领域技术人员将会理解无需特定实施例的一个或者多个特定特征或者优点，本发明仍可被实施。在其它情形下，附加特征以及优点可出现在本发明所有实施例中均未出现的特定实施例中。

[0030] 为什么批量请求/响应消息处理对于SDN网络来说是很重要的?下面结合一个具体情况进行解释。

[0031] 1、交换机和控制器之间的初始阶段

[0032] 当交换机和控制器首次或在角色改变(例如从从机到主机)后建立连接或同步信

息时，需要向彼此了解批量的交互消息，例如Hello消息、交换机配置、数据流表、群组表等。

[0033] 如果这是交换机的首次建立，则控制器希望应用批量的默认配置，特别是针对数据流表、群组表、度量表等。当然，这种消息需要逐个传递。还需要按照如下逻辑进行处理：

[0034] ■控制器将请求消息构件成配置文件。

[0035] ■文件格式是灵活的，并且可以在控制器与客户之间协商。例如，文件可以是压缩文件(例如使用zip、gzip、7zip标准或协商的压缩参数压缩的文件)，也可以是普通文件。

[0036] ■控制器发送文件信息给交换机。该信息至少包括文件名称、格式、长度、SHA摘要等。

[0037] ■当协商完成时，文件能够通过使用建立的连接逐个传递消息来传递、或者使用单独的协议来传递(TFTP、FTP、SFP、SCP)。

[0038] ■当文件被传递给交换机时，交换机开始转换文件(例如解压文件)、解析文件(使用预定义的方案)、并且逐个处理文件中的每个消息。

[0039] ■交换机发回针对每个请求消息的任意响应，大多数时间是错误情况或任意反常通知。

[0040] ■如果请求消息需要发送响应，其使用单个消息单独发送、也可以使用多部件消息或者其他可接受的消息来发送。

[0041] ■在交换机处理整个批量消息之后，它发送最终状态给控制器。

[0042] 2、控制器对来自交换机的现有配置执行快照(dump)

[0043] 如果控制器不具有交换机的数据，则它倾向于在做出任何改变之前对来自交换机的所有数据执行快照以便避免或减轻负面影响。

[0044] 对于这种情况，当然，它能够逐个查询每个项。实际上，效率很低并且所有查询和响应消息的网络延迟严重影响服务恢复，这是因为控制器在知道该交换机之前不能进行任何必要的动作。

[0045] 与上述情况类似，但是这时，控制器构建批量请求并且将其推送给交换机。交换机准备相关数据、构建文件、与控制器协商参数、最后将文件上传到控制器。然后控制器解析相关数据。

[0046] 根据以上情况，一个控制器能够管理几百个交换机，这种基于批量请求或响应的文件处理能够帮助解决或减轻以上问题。

[0047] 基于以上情况，本发明提供一种用于基于文件或单个消息的批量操作处理方法，如图1所示，该方法可以包括以下步骤：在步骤S101，构建配置文件，该配置文件至少包括头部信息和内容部分，所述头部信息至少包括片段索引，该片段索引至少包括每个要发送的命令的长度信息，所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件；在步骤S102，发送包括配置文件相关的信息的批量请求给接收端；在步骤S103，接收来自接收端的响应，并且基于所述接收端的响应来传输所述配置文件。从而能够实现在短时间内处理批量请求和响应消息，减轻网络业务开销，并且改进交互效率。

[0048] 其中，所述配置文件相关的信息可以包括例如文件名字、格式、长度等等，该方法可以由控制器或交换机执行。

[0049] 根据一个实施方式，所述配置文件还包括表明所述片段索引是否能包括所有命令的长度信息的字段，并且所述方法还包括：如果所述片段索引不能包括所有命令的长度信

息,则增加片段索引以便包括所有命令的长度信息。

[0050] 根据一个实施方式,所述批量请求包括要发送的配置文件的格式,所述响应包括所述接收端是否支持所述配置文件的格式的决定。

[0051] 根据一个实施方式,如果接收端支持所述配置文件的格式,则可以发送给发送端接受消息,表明接受该配置文件的格式,该接受消息可以具有文件摘要信息;然后发送端可以在接收到该接受消息时按照以该格式将该配置文件传送给接收端,并且可以使用所述文件摘要信息来识别正在传送的批量请求。

[0052] 根据一个实施方式,在所述接收端不支持所述配置文件的格式的情况下,所述响应还包括所述接收端偏好的文件格式,并且所述方法还包括:与接收端进一步协商传输所述配置文件要使用的格式。具体来说,如果所述发送端支持所述接收端偏好的文件格式,则将所述配置文件转换为所述接收端偏好的文件格式并传送给所述接收端;如果所述发送端不支持所述接收端偏好的文件格式,则向所述接收端发送新的文件格式以便与接收端进一步协商。

[0053] 根据一个实施方式,如果在传输过程中有错误发生,则接收端可以向发送端发送错误状态消息,终止当前传输过程,并且可选地还可以向发送端发送请求相关的错误以便发送端进行相应的处理;如果传输过程中没有错误,则可以在传输完成时向发送端发送完成状态消息。

[0054] 本发明中定义新的OFPT_MULTIPART_BULK_REQUEST(OFPT多部件批量请求)和OFPT_MULTIPART_BULK_REPLY(OFPT多部件批量回复)以及相关的数据结构来承载内容;还定义了针对OFPT_MULTIPART_BULK_REQUEST和OFPT_MULTIPART_BULK_REPLY的控制器行为和交换机行为。

[0055] 下面给出一个具体例子来进行说明。

[0056] 基于SDN网络开放流1.5.1版本(2015年3月26日最后更新)给出了如下示例性实现程序。

[0057] 增加了如下OFPT_MULTIPART_BULK_REQUEST和OFPT_MULTIPART_BULK_REQUEST动作和相关结构:

[0058]
```
enum ofp_type {
    /* Immutable messages. */
    OFPT_HELLO              = 0,  /* Symmetric message */
    OFPT_ERROR              = 1,  /* Symmetric message */
    OFPT_ECHO_REQUEST       = 2,  /* Symmetric message */
    OFPT_ECHO_REPLY         = 3,  /* Symmetric message */
    OFPT_EXPERIMENTER       = 4,  /* Symmetric message */
……
    /* Bundle operations (multiple messages as a single operation). */
    OFPT_BUNDLE_CONTROL     = 33, /* Controller/switch message */
```

```
    OFPT_BUNDLE_ADD_MESSAGE = 34, /* Controller/switch message */

    /* Controller Status async message. */
    OFPT_CONTROLLER_STATUS  = 35, /* Async message */

    OFPT_MULTIPART_BULK_REQUEST      = 36, /* Symmetric message */
    OFPT_MULTIPART_BULK_REPLY        = 37, /* Symmetric message */
};


#define MAX_BULK_COMMANDS 15

struct ofp_multipart_bulk_file_section_index {
    uint8_t index;                              /* Start from 0, content index for
section  */
    uint8_t has_more;                     /* 0 means no more index, else has more
section appended */
    uint8_t length[MAX_BULK_COMMANDS]; /* Each command' s length, appened on body
part*/
    uint8_t pad[1];
    uint8_t body[0];                     /* Body of the request. 0 or more bytes.
*/
};
OFP_ASSERT(sizeof(struct ofp_multipart_one_request) == 4);

#define MAX_FILE_NAME_LENGTH 32

enum ofp_multipart_bulk_file_type {
    OFPMP_BULK_FILE_TYPE_PLAIN = 0,       /* Plain file */
    OFPMP_BULK_FILE_TYPE_ZIP = 1,       /* zip file. */
    OFPMP_BULK_FILE_TYPE_GZIP = 2,       /* gzip file. */
    OFPMP_BULK_FILE_TYPE_7ZIP = 3,       /* 7zip file. */
    OFPMP_BULK_FILE_TYPE_RAR = 4,       /* rar file. */
};

struct ofp_multipart_bulk_file_info_header {
uint16_t length;                                    /* Length of this header */
uint8_t name[MAX_FILE_NAME_LENGTH]; /* File name */
uint16_t file_length;                                    /* File length  */
uint8_t sha_digest[20];                                 /* File SHA digest */
uint8_t format;                                     /* File format, one of
OFPMP_BULK_FILE_TYPE_* */

    uint8_t pad[7];
```

[0059]

```
    uint8_t body[0];                                    /* Extra info of this
header. 0 or more bytes. */
};
OFP_ASSERT(sizeof(struct ofp_multipart_bulk_file_info_header) == 64);

struct ofp_multipart_bulk_file_content_header {
uint8_t sha_digest[20];                    /* File SHA digest, to identify message
and avoid heavy overhead */
    uint8_t body[4];                              /* Extra info of this header.
0 or more bytes. */
};
OFP_ASSERT(sizeof(struct ofp_multipart_bulk_file_content_header) == 24);

enum ofp_multipart_bulk_message_type {
    OFPMP_BULK_MESSAGE_TYPE_FILE_HEADER = 0,       /* File header */
    OFPMP_BULK_MESSAGE_TYPE_FILE_CONTENT = 1,       /* File content */
};

struct ofp_multipart_bulk_request {
    struct ofp_header header;
    uint8_t message_type;                       /* One of
OFPMP_BULK_MESSAGE_TYPE_FILE_*  */
[0060]  uint8_t length;                        /* Length of this message header  */
uint8_t has_more;                       /* 0 means no more content, else has more
data appended */
    uint8_t pad[5];
/* depents on message type, the following content is either
ofp_multipart_bulk_file_info_header or
ofp_multipart_bulk_file_content_header */
    uint8_t body[0];                        /* Body of the response. 0 or more bytes.
*/
};
OFP_ASSERT(sizeof(struct ofp_bulk_request) == 16);

enum ofp_multipart_bulk_reply_status {
    OFPMP_BULK_REPLY_STATUS_DONE = 0,        /* Success */
    OFPMP_BULK_REPLY_STATUS_ERROR = 1,        /* Failed */
    OFPMP_BULK_REPLY_STATUS_IN_PROGRESS = 2, /* In progress */
    OFPMP_BULK_REPLY_STATUS_NEGOTIATION = 3, /* Negotiation */
    OFPMP_BULK_REPLY_STATUS_ACCEPT = 4, /* Accept */
};

struct ofp_multipart_bulk_reply {
    struct ofp_header header;
uint8_t status;   /* One of OFPMP_BULK_REPLY_STATUS_XXX */
```

[0061]
```
uint8_t length;   /* Length of this reponse */
    uint8_t pad[6];
/* depents on status type, the following content is either
ofp_multipart_bulk_file_info_header or
ofp_multipart_bulk_file_content_header */

};
OFP_ASSERT(sizeof(struct ofp_bulk_request) == 16);
```

[0062] 下面举个例子来定义针对OFPT_MULTIPART_BULK_REQUEST和OFPT_MULTIPART_BULK_REQUEST的交换机行为和控制器行为。

[0063] 其中在设置与交换机的连接之后,控制器希望将默认配置推送到交换机。

[0064] A)控制器将默认配置构建为独立的文件

[0065] 该文件以如上定义的ofp_multipart_bulk_file_section_index为开始;

[0066] 然后将每个配置的内容作为正常的单个请求附加在片段索引之后;

[0067] 同时,控制器会更新片段索引区域中的每个独立的命令的相对应的长度信息;

[0068] 当前,片段索引对命令的数量具有默认限制。如果单个片段索引不能容纳所有命令,则将字段has_more设置为非零,然后将另一片段索引附加到文件结尾。然后重复该步骤,直到所有命令都在文件中。

[0069] 该文件可以独立构建,并且在大多数交换机之间可以公用,

[0070] B)控制器发送批量请求给客户端

[0071] 用OFPT_MULTIPART_BULK_REQUEST来填充字段ofp_multipart_bulk_request;

[0072] 将message_type设置为OFPMP_BULK_MESSAGE_TYPE_FILE_HEADER;

[0073] 用文件信息和偏好的格式来填充ofp_multipart_bulk_file_info_header;

[0074] 发出该请求。

[0075] C)交换机接收批量请求

[0076] 解析请求内容;

[0077] 提取文件信息;

[0078] 如果不支持所述文件格式,则与控制器交互OFPMP_BULK_REPLY_STATUS_NEGOTIATION状态类型。响应包含偏好的文件格式。如果控制器接受该文件格式,则协商完成。否则,控制器再次发出批量请求,该批量请求具有新的文件格式,并且重复协商过程。当前,它仅支持格式协商。

[0079] 如果交换机接受该文件格式,则发送具有摘要信息的OFPMP_BULK_REPLY_STATUS_ACCEPT。控制器使用摘要作为关键字来识别正在进行的批量请求。

[0080] D)控制器接收ACCEPT消息,将文件转换为合适的格式并且开始发送

[0081] 可能能够支持选择文件的传输方法。当前,控制信道用于翻译原始消息;

[0082] 将原始文件转换为协商的文件格式,例如从普通文件到压缩文件;

[0083] 用OFPT_MULTIPART_BULK_REQUEST来填充ofp_multipart_bulk_request;

[0084] 将message_type设置为OFPMP_BULK_MESSAGE_TYPE_FILE_CONTENT;

[0085] 用文件信息来填充ofp_multipart_bulk_file_content_header;

[0086] 将文件内容附加到请求中;

[0087] 发出请求,它不请求来自交换机的显性回复;

[0088] 重复上述步骤,直到到达文件结尾;

[0089] 如果存在来自交换机的任意显性响应,则用正常逻辑进行处理。

[0090] E)交换机接收批量消息

[0091] 摘要是针对每个批量请求和响应的关键字;

[0092] 交换机接收所有消息,并将其组装到一个文件中;

[0093] 将文件转换为普通文件;

[0094] 在片段索引的帮助下解析文件;

[0095] 相应地处理每个正常请求;

[0096] 如果有任何错误发生,则它发送OFPMP_BULK_REPLY_STATUS_ERROR状态给控制器,然后终止当前操作,它还可以返回与错误相关的请求以便促进问题调试;

[0097] 如果没有错误,则发送OFPMP_BULK_REPLY_STATUS_DONE给控制器;

[0098] 批量请求和响应会话到此完成。

[0099] OFPT_MULTIPART_BULK_REQUEST和OFPT_MULTIPART_BULK_REPLY还可以扩展为具有更多功能。

[0100] 根据一个实施方式,所述批量请求被合并到一个或多个请求消息中,和/或批量响应被合并到一个或多个响应消息中。

[0101] 根据一个实施方式,如果合并后的请求消息和/或合并后的响应消息的长度超过预定长度,则所述请求消息和/或所述响应消息被增加。该预定长度可以根据需要来设定,例如为1500字节。

[0102] 为什么具有多个请求/响应的一个消息对于SDN网络来说是很重要的?下面是结合一个具体情况进行解释。

[0103] 关于正常请求和响应大小,根据当前规范,大多请求/响应的大小小于100字节,因特网的正常MTU(最大传输单元)是1500字节。一般来说,字节越小花费的传输时间越小,但是考虑到从源节点到目的节点的网络路径,路径上的延迟可能比不同长度(例如100字节对比1500字节)的数据包花费的时间大得多,这就意味着如果单个数据包/消息能够容纳源节点与目的节点之间的更多消息,则对于群组请求和响应消息来说,可以减小网络延迟,并且改进传输效率。

[0104] 基于以上情况,本发明中定义了新的OFPT_MULTIPART_ONE_REQUEST(OFPT多部件一个请求)和OFPT_MULTIPART_ONE_REPLY(OFPT多部件一个回复)以及相关的数据结构来承载内容;针对OFPT_MULTIPART_ONE_REQUEST和OFPT_MULTIPART_ONE_REPLY的控制器行为和交换机行为。

[0105] 下面给出一个具体例子。

[0106] 基于SDN网络开放流1.5.1版本(2015年3月26日最后更新)给出了如下示例性实现程序。

[0107] 增加了如下OFPT_MULTIPART_ONE_REQUEST和OFPT_MULTIPART_ONE_REQUEST动作和相关结构:

```
enum ofp_type {
    /* Immutable messages. */
    OFPT_HELLO              = 0,  /* Symmetric message */
    OFPT_ERROR              = 1,  /* Symmetric message */
    OFPT_ECHO_REQUEST       = 2,  /* Symmetric message */
    OFPT_ECHO_REPLY         = 3,  /* Symmetric message */
    OFPT_EXPERIMENTER       = 4,  /* Symmetric message */
......

    /* Bundle operations (multiple messages as a single operation). */
    OFPT_BUNDLE_CONTROL     = 33, /* Controller/switch message */
    OFPT_BUNDLE_ADD_MESSAGE = 34, /* Controller/switch message */

    /* Controller Status async message. */
    OFPT_CONTROLLER_STATUS  = 35, /* Async message */
```

[0108]

```
    OFPT_MULTIPART_ONE_REQUEST      = 36, /* Controller/switch message */
    OFPT_MULTIPART_ONE_REPLY        = 37, /* Controller/switch message */
};


#define MAX_REQUEST_COMMANDS 15
#define MAX_RESPONSE_COMMANDS 15

struct ofp_multipart_one_request {
    struct ofp_header header;
    uint8_t index;                          /* Start from 0, content index for section
*/
    uint8_t has_more;                        /* 0 means no more index, else has more
section appended */
    uint8_t length[MAX_REQUEST_COMMANDS]; /* Each request's length, appened on body
part*/
    uint8_t pad[3];
```

```
    uint8_t body[0];                    /* Body of the request. 0 or more bytes.
*/
};
OFP_ASSERT(sizeof(struct ofp_multipart_one_request) == 20);

enum ofp_multipart_reply_flags {
    OFPMPF_REPLY_MORE  = 1 << 0  /* More replies to follow. */
};


struct ofp_multipart_one_reply {
    struct ofp_header header;
    uint8_t index;                      /* Start from 0, content index for
section  */
    uint8_t has_more;                   /* 0 means no more index, else has more
section appended */
    uint8_t length[MAX_RESPONSE_COMMANDS]; /* Each response's length, appened on
body part*/
    uint8_t pad[3];
    uint8_t body[0];                    /* Body of the response. 0 or more bytes.
*/
};
OFP_ASSERT(sizeof(struct ofp_multipart_one_reply) == 20);
```

[0109]

[0110] 对于OFPT_MULTIPART_ONE_REQUEST,将头部和请求填入ofp_multipart_one_request中:

[0111] 字段ofp_header(头部)是头部的实例,其将请求设置为OFPT_MULTIPART_ONE_REQUEST;

[0112] 字段索引、片段索引从0开始;

[0113] 关于字段has_more,如果在当前请求中有多于一个片段,则将其设置为非零,否则将其设置为零;

[0114] 关于字段length[MAX_REQUEST_COMMANDS],其定义了一个片段中的MAX_REQUEST_COMMANDS(当前其为15)个请求命令,在队列中,该值应该是连续的,并且默认值是0,如果实际上存在附加的命令,则队列的相对应成员应当设置为具有该请求的长度,队列成员在uint8_t中定义,其取值范围为0到255,它对于大多数请求来说足够长,如有需要,还可以将其扩展到更大范围;

[0115] pad(填充)字段在这里用于64比特的填充;

[0116] body(主体)字段是每个请求的真实内容;

[0117] 如果has_more为非零,则重复上述构建或解析原理以便构建/解析消息。

[0118] 控制器可以使用上述结构将批量请求合并到一个消息中。

[0119] 该请求的默认限制是MTU(默认为1500),如果请求的总长度超过MTU,则需要分离的OFPT_MULTIPART_ONE_REQUEST。

[0120] 当交换机接收到OFPT_MULTIPART_ONE_REQUEST消息时,它使用相反的逻辑来解析该消息。交换机的默认行为是将所有响应使用以下结构和逻辑组装到OFPT_MULTIPART_ONE_REPLY中。但是交换机也可以发送独立地对应于每个请求的正常回复。

[0121] 对于OFPT_MULTIPART_ONE_REPLY,将头部和响应使用上面定义的结构填充到into ofp_multipart_one_response中:

[0122] 字段ofp_header是头部的实例,其将响应设置为OFPT_MULTIPART_ONE_REPLY;

[0123] 字段索引、片段索引从0开始;

[0124] 关于字段Has_more,如果在当前响应中有多于一个片段,则将其设置为非零,否则将其设置为零;

[0125] 关于字段length[MAX_REQUEST_COMMANDS],其定义了一个片段中的MAX_REQUEST_COMMANDS(当前其为15)个响应命令,在队列中,该值应该是连续的,并且默认值是0,如果实际上存在附加的命令,则队列的相对应成员应当设置为具有该响应的长度,队列成员在uint8_t中定义,其取值范围为0到255,它对于大多数响应来说足够长,如有需要,还可以将其扩展到更大范围;

[0126] pad字段在这里用于64比特的填充;

[0127] body是每个响应的真实内容;

[0128] 如果has_more为非零,则重复上述构建或解析原理以便构建/解析消息。

[0129] 如上所述,交换机发送OFPT_MULTIPART_ONE_REPLY消息、后者与之前一样发送独立的响应消息。

[0130] OFPT_MULTIPART_ONE_REPLY的默认限制是MTU(默认为1500),如果响应的总长度超过MTU,则需要分离的OFPT_MULTIPART_ONE_REPLY。

[0131] 如果从消息解析出的请求或响应无效,则交换机或控制器报告相对应的错误消息,交换机或控制器能够解决错误、终止请求/响应或再次进行重试。

[0132] 对于响应部分,如果总的回复的长度超过MTU,则其应当发回单独的回复消息,这意味着一个请求可能触发多个回复消息,接收机可以考虑到这种情况并且相应地做出动作。

[0133] 根据本发明的另一方面,提供了一种用于基于文件或单个消息的批量操作处理的设备,如图2所示,该设备包括:

[0134] 配置装置110,用于构建配置文件,该配置文件至少包括头部信息和内容部分,所述头部信息至少包括片段索引,该片段索引至少包括每个要发送的命令的长度信息,所述内容部分包括按照片段索引的顺序将对应的要发送的命令组织形成的普通文件;

[0135] 发送装置120,用于发送包括配置文件相关的信息的批量请求给接收端;

[0136] 接收装置130,用于接收来自接收端的响应;并且

[0137] 所述发送装置120还用于基于所述接收端的响应来传输所述配置文件。

[0138] 根据一个实施方式,所述配置文件还包括表明所述片段索引是否能包括所有命令的长度信息的字段,并且所述配置装置110还用于:如果所述片段索引不能包括所有命令的长度信息,则增加片段索引以便包括所有命令的长度信息。

[0139] 根据一个实施方式,所述批量请求被合并到一个或多个请求消息中,和/或批量响应被合并到一个或多个响应消息中。

[0140] 根据一个实施方式,如果合并后的请求消息和/或合并后的响应消息的长度超过预定长度,则所述请求消息和/或所述响应消息被增加。

[0141] 根据一个实施方式,所述批量请求包括要发送的配置文件的格式,所述响应包括

所述接收端是否支持所述配置文件的格式的决定。

[0142] 根据一个实施方式，如果接收端支持所述配置文件的格式，则可以发送给设备接受消息，表明接受该配置文件的格式，该接受消息可以具有文件摘要信息；然后所述设备可以在接收到该接受消息时按照以该格式将该配置文件传送给接收端，并且可以使用所述文件摘要信息来识别正在传送的批量请求。

[0143] 根据一个实施方式，在所述接收端不支持所述配置文件的格式的情况下，所述响应还包括所述接收端偏好的文件格式，所述发送装置120还用于与接收端进一步协商传输所述配置文件要使用的格式，具体来说，如果所述设备支持所述接收端偏好的文件格式，则将所述配置文件转换为所述接收端偏好的文件格式并传送给所述接收端；如果所述设备不支持所述接收端偏好的文件格式，则向所述接收端发送新的文件格式以便与接收端进一步协商。

[0144] 根据一个实施方式，如果在传输过程中有错误发生，则接收端可以向所述设备发送错误状态消息，终止当前传输过程，并且可选地还可以向所述设备发送请求相关的错误以便所述设备进行相应的处理；如果传输过程中没有错误，则可以在传输完成时向所述设备发送完成状态消息。

[0145] 根据一个实施方式，其中所述设备可以为控制器，或者为交换机。

[0146] 应当理解，虽然本发明的上述具体实例是针对SDN网络给出的，但是还可以应用于其他合适的网络，而不限于仅SDN网络。虽然本发明给出的示例中用于基于文件或单个消息的批量操作处理方法是由控制器执行的，但是还可以由其他设备例如交换机、处理器、CPU等具有控制和处理功能的设备来执行，而不限于仅控制器。

[0147] 根据本发明提供的设备中的配置装置110、发送装置120、接收装置130中的至少一个可以包括程序指令，当执行程序指令时，使得所述装置根据如上所述的示例性实施例运行。上述配置装置110、发送装置120、接收装置130中的任意一个可被集成在一起或者在分开的部件中执行，并且可以是适合本地技术环境的任意类型，并作为非限制性实施例可包括通用计算机，专用计算机，微处理器，数字信号处理器(DSP)以及基于多核处理器架构的处理器中的一个或者多个。上述ROM可以是适合本地技术环境的任意环境并且可利用任意适合的数据存储技术而被执行，诸如基于半导体的存储设备，闪存，磁存储设备以及系统，光存储设备以及系统，固定存储器以及可移动存储器。

[0148] 总之，不同示例性实施例可在硬件或者专用电路，软件、逻辑或者其任意组合中实施。例如，一些方面可在硬件中实施，而其它方面可在通过控制器，微处理器或者其它计算设备执行的固件或者软件中执行，尽管本发明不限于此。虽然本发明示例性实施例的不同方面可按照框图，流程图，或者使用一些其它图形表示的方式被阐释以及描述，本领域技术人员将会理解此处描述的这些块，部件，系统，技术或者方法可在如非限定实施例的硬件，软件，固件，专用电路或者逻辑，通用硬件或者控制器或者其它计算设备，或者其一些组合中实现。

[0149] 应当理解本发明示例性实施例的至少一些方面可体现为计算机可执行指令，例如在通过一个或者多个计算机或者其它设备执行的一个或者多个程序模块中。总得来说，当被计算机或者其它设备中的处理器执行时，程序模块包括执行特定任务或者实施特定概要数据类型的例行程序，程序，客体，部件，数据结构等。计算机可执行指令可存储在计算机可

读介质中，例如硬盘，光盘，可移动存储介质，固态存储器，随机存取存储器(RAM)等。本领域技术人员将会意识到，程序模块的功能如在不同实施例中期望的合并或者分散。此外，功能可全部或者部分实现在固件或者硬件中，例如集成电路，现场可编程门阵列(FPGA)等。

[0150] 尽管本发明特定实施例已被公开，本领域技术人员将会理解在不脱离本发明精神和范围的前提下可对特定实施例做出变化。本发明的范围并不局限于特定实施例，并且附随权利要求涵盖了本发明范围中此类应用，变型以及实施例的任意以及全部。

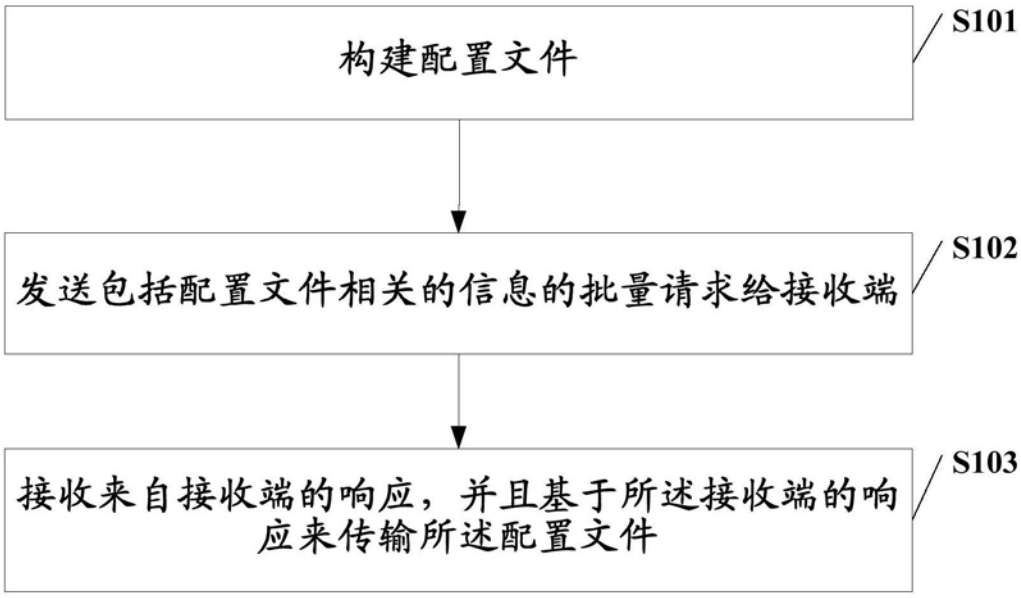

构建配置文件
S101
发送包括配置文件相关的信息的批量请求给接收端
S102
接收来自接收端的响应，并且基于所述接收端的响应来传输所述配置文件
S103

图1


配置装置
110
发送装置
120
接收装置
130

图2