1、 OpenNF：在网络功能控制方面实现创新

2、网络功能：对数据包/流执行复杂的状态操作

3、 NF趋势：(网络功能虚拟化（NFV）->动态分配NF实例)+(软件定义网络->动态重新路由流)=动态重新分配数据包处理

4、弹性NF缩放

->满足性能SLA（可能数据包丢失）

->最大限度地降低运营成本

->准确监控流量

5、问题：NFV + SDN不足：无法同时上边三条

6、 SLA +成本+准确度：我们需要什么？快速移动，复制或共享内部NF状态以及更新网络转发状态：无损，保续，同时适用于其他方案

7、挑战

->以最小的变化支持许多NF

->处理竞争条件路由更新数据包状态

->限制开销

8、现有方法

->虚拟机复制：无法组合--有限的重新平衡

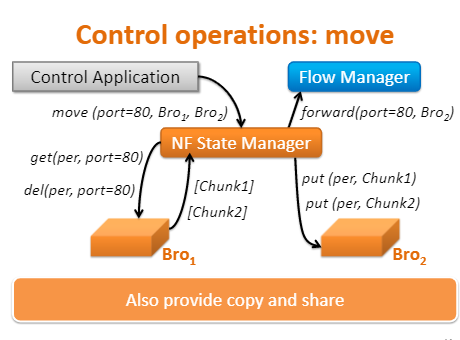
->拆分/合并：状态分配和访问通过库；发生地址特定问题--限制适用性；数据包可能被丢弃或重新排序错误--NF行为

9、OpenNF概述：控制应用程序->（OpenNF控制器移动/复制/共享状态）->NF状态管理器->导出/导入状态流管理器

10、NF状态分类：由NF创建或更新的状态适用于单个流或流的集合

11、NF API：导出/导入状态功能：获取，放置，删除--无需公开/更改内部状态组织！

12、控制操作：移动->移动期间丢失的更新



->无损失：所有状态更新应反映在传输状态，并且所有数据包应该是已处理的

->只需要更改NF的接收数据包功能！

->使用事件进行无损移动

1. 在Bro 1上启用事件（红色，丢弃）
2. 在Bro 1上获取/删除
3. 控制器上的缓冲事件。
4. 放置Bro 2
5. 将事件中的数据包刷新到Bro
6. 更新转发

->保序移动

1. 将事件中的数据包刷新到Inst 2
2. Inst2上启用事件（红色，缓冲区）
3. 转发更新：发送到Inst 1和控制器
4. 等待来自交换机的数据包（记得最后）转发更新：
5. 发送到Inst 2 Drop等待事件来自Inst 2的最后一个数据包
6. 来自Inst 2的数据包释放缓冲区

->限制开销--应用程序决定（基于NF和目标）：

1. 操作的粒度按每个多个过滤器所有范围
2. 所需的保证：无 LF LF + OP

13、OpenNF：SLA +成本+准确性

->处理多样性--根据与流量的关联处理导出/导入状态

->处理竞争条件：事件+锁定步骤转发更新

->限制开销：应用程序选择粒度和保证

14、实现

控制器（3.8K行Java）通信库（2.6K行C）修改后的NF（代码增加3-8％）

15、弹性缩放的总体优势

Bro IDS处理10K pkts / sec 180秒：将HTTP流（489）移动到新IDS 360秒：移回旧IDS SLA：260ms移动（无损）准确度：相同的日志条目使用一个IDS VM复制：不正确的日志条目成本：移动状态后的缩放等待流程死亡：延迟25+分钟缩放25

16、未来的工作

减少缓冲->在状态传输期间允许数据包处理，然后重放输入以使状态达到速度

提高可扩展性->点对点状态传输

（半）自动修改NF->静态程序分析

17、结论

数据包处理的动态重新分配

使新服务实现SLA +成本+准确性需要快速，安全地控制内部NF状态

OpenNF提供灵活高效的控制，只需很少的NF修改了解更多信息并尝试一下

83备份复制和共享示例应用程序：弹性NF缩放评估：控制器可扩展性评估：保证的重要性评估：粒度控制的好处30

18、复制和共享操作

当多个实例需要某些状态时

使用复制--无或最终一致性--一次，定期，基于事件等。

共享--强或严格一致性---为所有数据包引发事件；一次释放事件状态在发布之前；被复制下一个事件复制