2011-11-29周报告

by can.

OUTLINE

- 镜像存储算法
- NS3的实验配置文件
- testbed实验拓扑配置文件

镜像存储算法

镜像存储算法

几个定义

- 原始镜像：系统内预置的镜像文件
- 用户镜像：由用户使用一段时间后的镜像文件
- delta文件：用户镜像与原始镜像的不同

镜像存储算法

基本思路

- 用户镜像 - 原始镜像 = delta文件
- 即把用户镜像与原始镜像比较，只存储delta文件
- 简单粗暴，坚实可靠

镜像存储算法

How to

- 将两个镜像文件按每块大小为K分块
- 计算每块的MD5 sum,存入文本文件src.md5和user.md5
- 根据两个文件的不同生成delta文件

镜像存储算法

How to

src.md5

```
50 c597c7bc7f2a69224ec886acf7344022
51 4aafa7ff9c67ce815a049db17df6cca8
52 4aafa7ff9c67ce815a049db17df6cca8
53 4aafa7ff9c67ce815a049db17df6cca8
54 4aafa7ff9c67ce815a049db17df6cca8
55 7d179af260e129a78bbc035e40e26765
56 4aafa7ff9c67ce815a049db17df6cca8
57 8b63e7732bbbae5f3a71bcb135be66ac
58 d1c073c43649eecbbb0389e0777b3be2
59 02eed73cccdb4f9a57364aabccb55bd0
60 4aafa7ff9c67ce815a049db17df6cca8
61 8b63e7732bbbae5f3a71bcb135be66ac
62 113b7f2f33d9035e4d9c5f52fc8b54d6
63 0cf34723f6e194263d31fcea2e47a258
64 02eed73cccdb4f9a57364aabccb55bd0
65 4aafa7ff9c67ce815a049db17df6cca8
66 113b7f2f33d9035e4d9c5f52fc8b54d6
67 113b7f2f33d9035e4d9c5f52fc8b54d6
68 113b7f2f33d9035e4d9c5f52fc8b54d6
69 113b7f2f33d9035e4d9c5f52fc8b54d6
70 113b7f2f33d9035e4d9c5f52fc8b54d6
```

diff

```
50,54c50
< c597c7bc7f2a69224ec886acf7344022
< 4aafa7ff9c67ce815a049db17df6cca8
< 4aafa7ff9c67ce815a049db17df6cca8
< 4aafa7ff9c67ce815a049db17df6cca8
< 4aafa7ff9c67ce815a049db17df6cca8
---
> b010c0adfcc083df077189c363c19963
55a52
> 113b7f2f33d9035e4d9c5f52fc8b54d6
57,58c54,56
< 8b63e7732bbbae5f3a71bcb135be66ac
< d1c073c43649eecbbb0389e0777b3be2
---
> 4aafa7ff9c67ce815a049db17df6cca8
> 217bc4c918bb75dde4aee5ec25a91dcb
> 1792b9facc579a00e7bbf50cc5cf7abe
59a58,60
> 113b7f2f33d9035e4d9c5f52fc8b54d6
> 113b7f2f33d9035e4d9c5f52fc8b54d6
> 113b7f2f33d9035e4d9c5f52fc8b54d6
61c62
< 8b63e7732bbbae5f3a71bcb135be66ac
---
> d1c073c43649eecbbb0389e0777b3be2
63,64c64
< 0cf34723f6e194263d31fcea2e47a258
< 02eed73cccdb4f9a57364aabccb55bd0
---
> 1792b9facc579a00e7bbf50cc5cf7abe
```

user.md5

```
50 b010c0adfcc083df077189c363c19963
51 7d179af260e129a78bbc035e40e26765
52 113b7f2f33d9035e4d9c5f52fc8b54d6
53 4aafa7ff9c67ce815a049db17df6cca8
54 4aafa7ff9c67ce815a049db17df6cca8
55 217bc4c918bb75dde4aee5ec25a91dcb
56 1792b9facc579a00e7bbf50cc5cf7abe
57 02eed73cccdb4f9a57364aabccb55bd0
58 113b7f2f33d9035e4d9c5f52fc8b54d6
59 113b7f2f33d9035e4d9c5f52fc8b54d6
60 113b7f2f33d9035e4d9c5f52fc8b54d6
61 4aafa7ff9c67ce815a049db17df6cca8
62 d1c073c43649eecbbb0389e0777b3be2
63 113b7f2f33d9035e4d9c5f52fc8b54d6
64 1792b9facc579a00e7bbf50cc5cf7abe
65 4aafa7ff9c67ce815a049db17df6cca8
66 113b7f2f33d9035e4d9c5f52fc8b54d6
67 113b7f2f33d9035e4d9c5f52fc8b54d6
68 113b7f2f33d9035e4d9c5f52fc8b54d6
69 113b7f2f33d9035e4d9c5f52fc8b54d6
70 113b7f2f33d9035e4d9c5f52fc8b54d6
```

镜像存储算法

复杂度分析

最耗时的部分主要有:

- 计算MD5 sum：O(n)
- 两个镜像进行比较：O(mn)*
- 生成delta文件/从delta文件恢复：O(max(m,n))

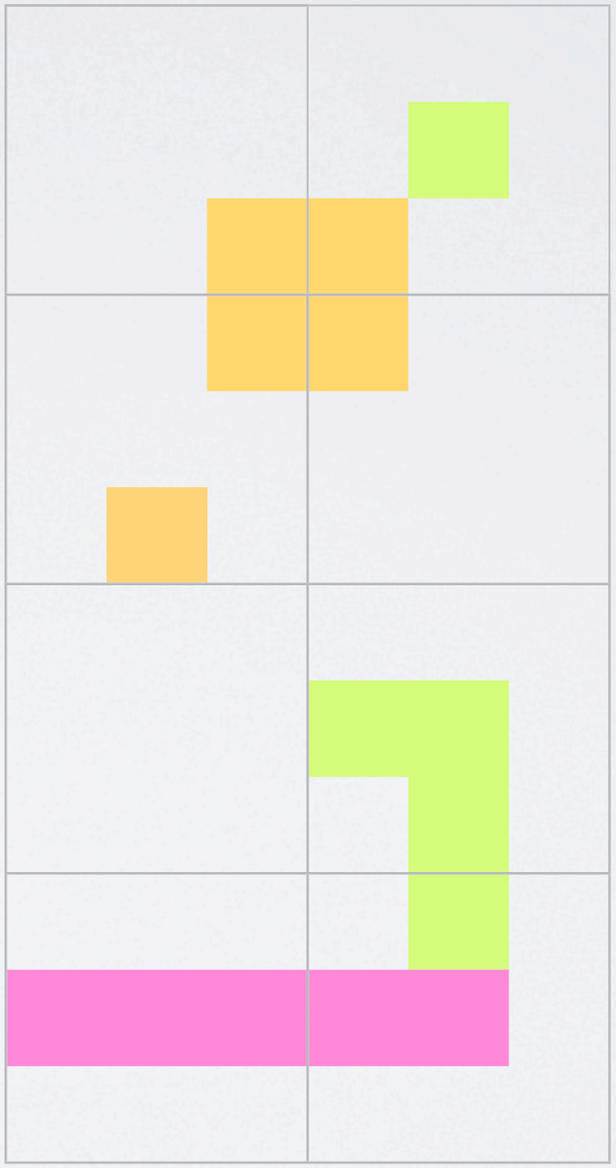
*http://www.avatar.se/molbioinfo2001/dynprog/dynamic.html

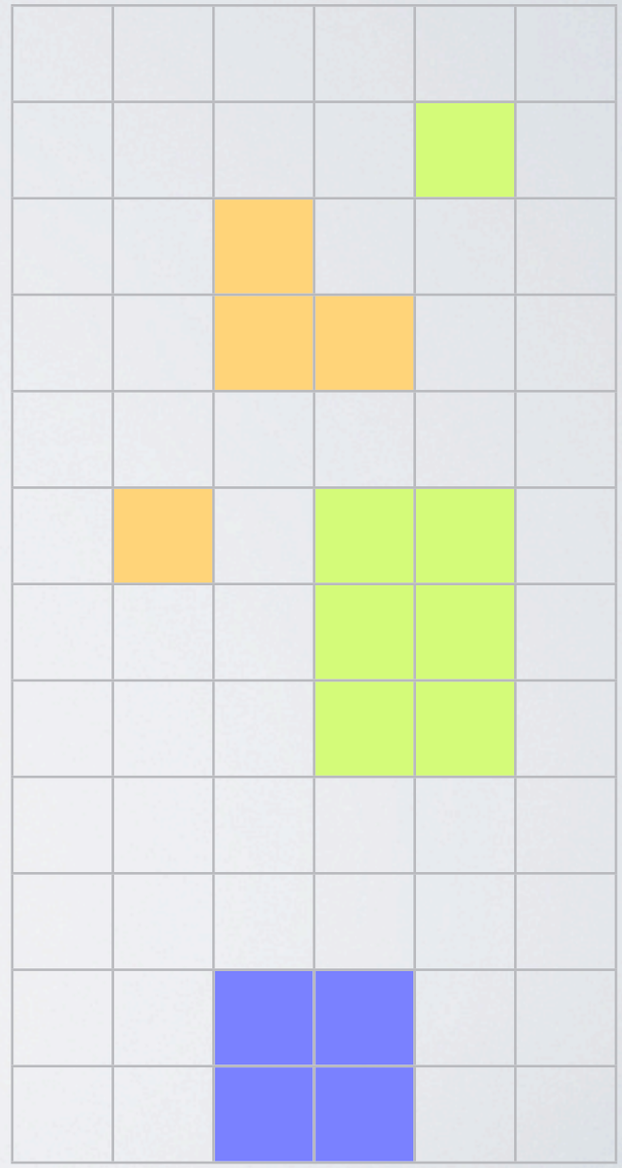
镜像存储算法

Trade-off

分块K的大小(颗粒度):

- 太大：浪费空间
- 太小：浪费时间





镜像存储算法

瓶颈

计算用户镜像MD5：

- 文件大小：3.57GB
- 分块大小：64K,共54173块;与原始镜像相比改变约1% *
- CPU占用：5%~6% (Core 2,2.4GHz)
- 硬盘速率：约34M/s
- 总耗时：1min35.316s

SSD?
fiber channel?
RAID?

*用apt-get update更新了一些内容

镜像存储算法

future work

- 实现一个可用的demo
- 探求在常规环境中K的优化值
- 基于RAID对算法优化

NS3的实验配置文件

NS3的实验配置文件

NS3简介

- NS3： Network Simulator 3
- 与NS2没有直接继承关系
- 本身使用C++编写,提供C++和Python的API
- tutorial:

  http://www.nsnam.org/docs/release/3.12/tutorial/ns-3-tutorial.pdf

NS3的实验配置文件

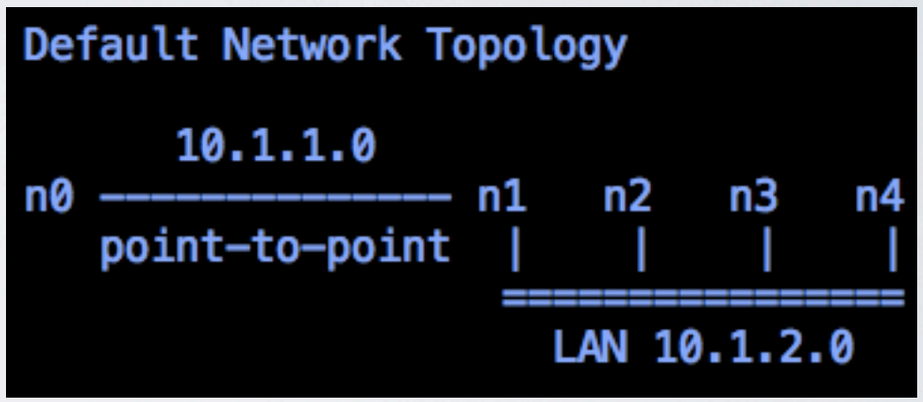
基本概念

- Node：”a computer to which you will add functionality”
- Application：”a user program that generates some activity to be simulated”
- Channel：“managing communication subnetwork objects and connecting nodes to them”
- Net Device：Network Interface Cards,NIC

NS3的实验配置文件

简单样例

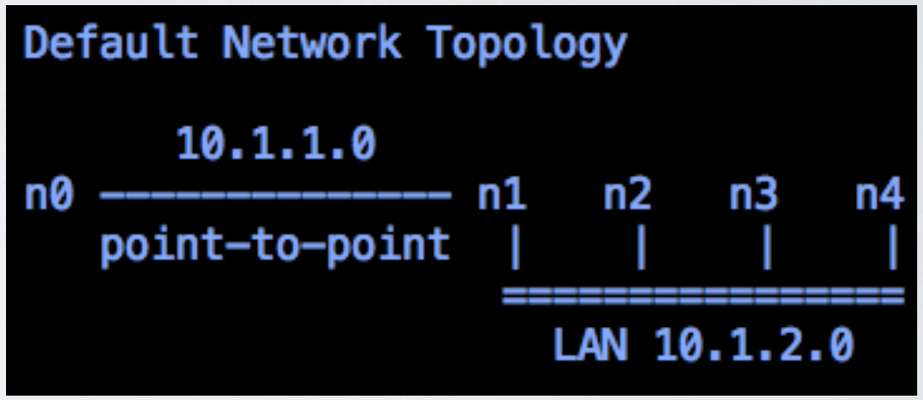
```
NodeContainer p2pNodes;

p2pNodes.Create (2);
```




```
NodeContainer csmaNodes;

csmaNodes.Add (p2pNodes.Get (1));

csmaNodes.Create (nCsma);
```

NS3的实验配置文件

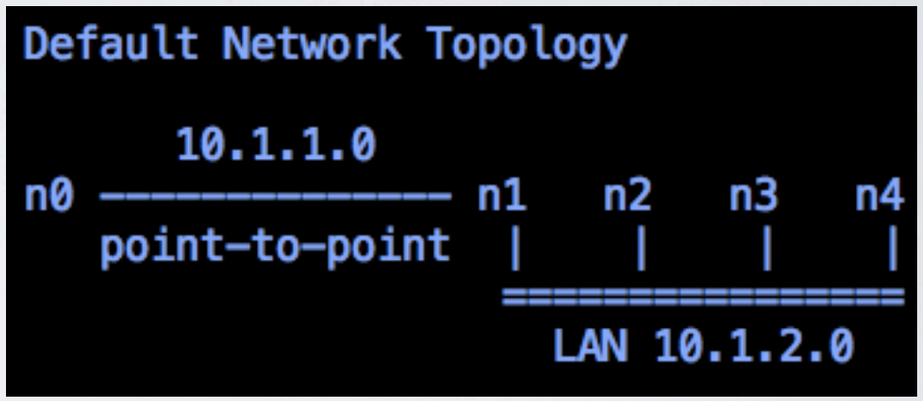
简单样例




```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
```

NS3的实验配置文件

简单样例

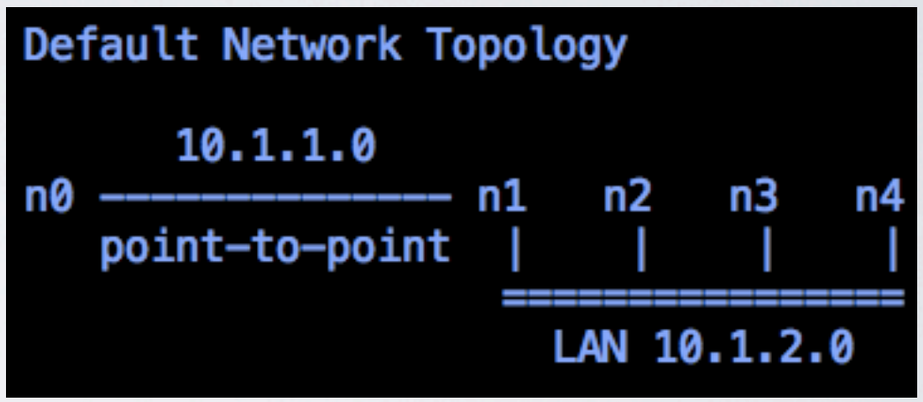



```
NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);
```

NS3的实验配置文件

简单样例

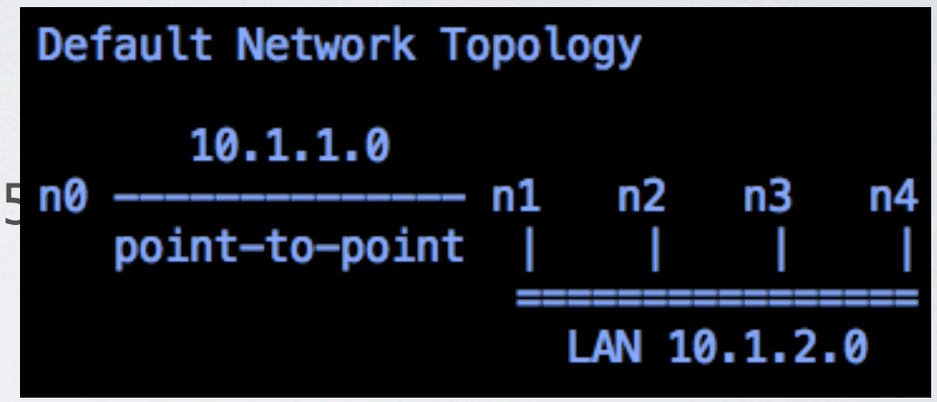
```
InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));

stack.Install (csmaNodes);
```



NS3的实验配置文件

简单样例

```
Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.25

Ipv4InterfaceContainer p2pInterfaces;

p2pInterfaces = address.Assign (p2pDevices);



address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer csmaInterfaces;

csmaInterfaces = address.Assign (csmaDevices);
```



NS3的实验配置文件

简单样例

```
  ApplicationContainer serverApps = echoServer.Install
(csmaNodes.Get (nCsma));

  serverApps.Start (Seconds (1.0));

  serverApps.Stop (Seconds (10.0));

  ApplicationContainer clientApps = echoClient.Install
(p2pNodes.Get (0));

  clientApps.Start (Seconds (2.0));

  clientApps.Stop (Seconds (10.0));
```

NS3的实验配置文件

样例输出

```
Sent 1024 bytes to 10.1.2.4          client -> server

Received 1024 bytes from 10.1.1.1   server received data

Received 1024 bytes from 10.1.2.4   client received echo data
```

NS3的实验配置文件

log & tracing

- 支持在某个端口模拟抓包
- 按重要程度分级的日志机制

  NS_LOG_ERROR,NS_LOG_WARN,etc...

- 基于回调函数的事件跟踪机制

NS3的实验配置文件

NS3的启发

- 用API代替“配置文件”?
- Log机制和事件触发的跟踪机制

TESTBED实验拓扑配置文件

v0.1

TESTBED实验拓扑配置文件

v0.1

- 使用xml
- “v0.1”：各种征求意见
- 网络拓扑、网络协议、操作系统、软件部署
- 基于node和link组织拓扑

TESTBED实验拓扑配置文件

v0.1

node type

- host:主机,{interface,software,protocol stack}
- switch:普通交换机,{interface}
- openflow switch:文艺交换机,{interface,controller}
- router:路由器(三层交换机),{interface}

TESTBED实验拓扑配置文件

v0.1

link type

- p2p:点到点链路,{from,to,bandwidth,packet loss,delay}
- broadcast:共享链路,{interfaces,bandwidth,packet loss,delay}

DEMO

https://github.com/cannium/xml-experiment-description-language.git