中国科学技术大学
University of Science and Technology of China

# NdnSIM Getting Start

Hebi Li
12/16/2013

# Content

- NS3
  - compile
  - LOG
  - Schedule
  - Names
  - Basic Structure
- ndnSim

# 1.1 NS3 overview

- Network Simulator
- Doxygen: Documentation Generator
- Mercurial: code maintain
- Waf: build system
- Website:
  - Www.nsnam.org
  - Www.selenic.com/mercurial
  - Code.google.com/p/waf

# 1.2 compile

- ./waf configure -d debug –enable-examples –enable-tests
- ./waf
- ./waf --doxygen

# 1.3 NS_LOG

- Environment variable: NS_LOG
  - NS_LOG_ERROR
  - NS_LOG_WARN
  - NS_LOG_DEBUG
  - NS_LOG_INFO
  - NS_LOG_FUNCTION
  - NS_LOG_LOGIC
  - NS_LOG_ALL

# 1.3.1 NS_LOG examples

- In coding
  - NS_LOG_COMPONENT_DEFINE(“MyLog”);
  - NS_LOG_DEBUG(“xxx”);
- Environment Variable
  - NS_LOG=”MyLog=level_all|prefix_func”
- I use:
  - NS_LOG=”*=level_all|prefix_func|prefix_time”

# 1.4 Ns3::Simulator

- Simulator::Stop(Time)
- Simulator::Run()
- Simulator::Destroy()
- Simulator::Now()
- Simulator::Schedule()

# 1.4.1 Simulator::Schedule

- Schedule(Time, void(*f), T1, T2)
  - *f: function name
- Schedule(Time, MEM, OBJ, T1 [, T2 .. ])
  - Time:
  - MEM: &func
  - OBJ: this
  - T[12345]: param

# 1.5 Names

- Ns3::Names
  - ::Add(std::string name, Ptr<Object> object)
  - ::Find(std::string) ---> Ptr<T>
  - ::FindName(Ptr<Object>) ---> std::string

# 1.6 Basic Structure

- Node
  - GetId()
  - GetObject()
- NodeContainer
  - Iterator
  - Begin
  - End
  - GetN
  - Get(index)
  - Add(): para ---> NodeContainer, Node, std::string

# 2. ndnSIM

- Config && Init && Stack
- FIB
- PIT(no reference)
- Interest
- Data
- App
- Start

# 2.1 Config && Init && Stack

```
Config::SetDefault("ns3::PointToPointNetDevice::DataRate", StringValue("1Mbps"));
Config::SetDefault("ns3::PointToPointChannel::Delay", StringValue("10ms"));
Config::SetDefault("ns3::DropTailQueue::MaxPackets", StringValue("20"));
```

```
/* topology */
AnnotatedTopologyReader topologyReader("", 25);
topologyReader.SetFileName("topo/topo.txt");
topologyReader.Read();
```

```
/* NDN stack */
ndn::StackHelper ndnHelper;
ndnHelper.SetContentStore("ns3::ndn::cs::Fifo", "MaxSize", "10000");
ndnHelper.SetForwardingStrategy("ns3::ndn::fw::BestRoute");
ndnHelper.InstallAll();
```

CS: FIFO, LRU, RANDOM, Stats::LRU, Freshness::LRU
Forwarding  Strategy: BestRoute, Flooding, SmartFlooding

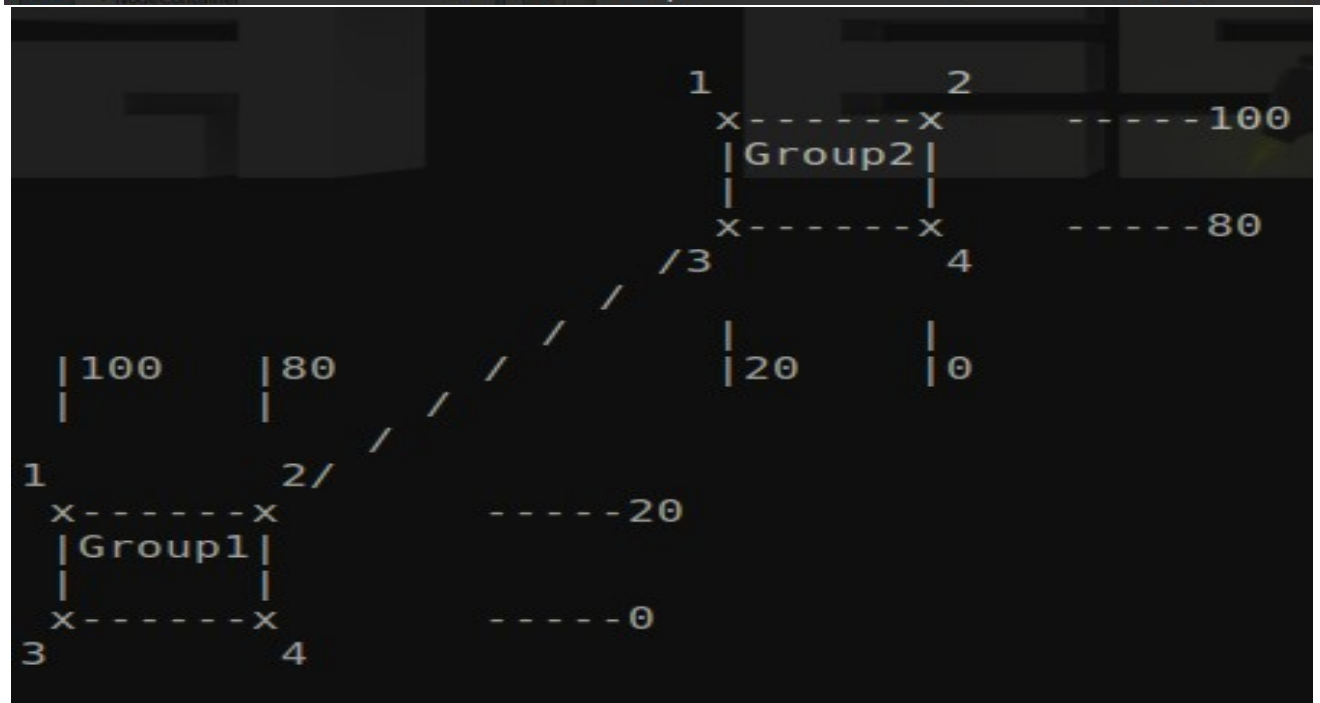
# 2.1.1 TopoReader



```
router

#node    comment yPos   xPos
Node11   NA      20     100
Node12   NA      20     80
Node13   NA      0      100
Node14   NA      0      80
Node21   NA      100    20
Node22   NA      100    0
Node23   NA      80     20
Node24   NA      80     0

link

# bandwidth: link bandwidth
# metric: routing metric
# delay:  link delay
# queue:  MaxPackets for transmission queue on the link (both directions)

#srcNode        dstNode  bandwidth       metric  delay    queue
Node11          Node12   1Mbps           1       10ms     10
Node11          Node13   1Mbps           1       10ms     10
Node14          Node12   1Mbps           1       10ms     10
Node14          Node13   1Mbps           1       10ms     10
```

Basic Structure

- Node
- NodeContainer



# 2.2 FIB(routing)

```
/* routing configure */
ndn::GlobalRoutingHelper routingHelper;
routingHelper.InstallAll();
for (int i=0;i<chatterNodes.GetN();i++) {
        Ptr<Node> node = chatterNodes.Get(i);
        routingHelper.AddOrigins("/"+Names::FindName(node), node);
        routingHelper.AddOrigins("/broadcast/chatapp", node);
}
ndn::GlobalRoutingHelper::CalculateRoutes();
```

```
void ChatApp::Config()
{
    Ptr<ndn::Fib> fib = GetNode()->GetObject<ndn::Fib>();
    Ptr<ndn::Name> prefix = Create<ndn::Name>("/broadcast/chatapp");
    fib->Add(*prefix, m_face, 0);
    prefix = Create<ndn::Name>("/"+m_name);
    fib->Add(*prefix, m_face, 0);
}
```

# 2.3 Interest

```
void ChatApp::SendInterest(std::string name)
{
    Ptr<ndn::Name> prefix = Create<ndn::Name>(name);
    Ptr<ndn::Interest> interest = Create<ndn::Interest>();
    UniformVariable rand(0, std::numeric_limits<uint32_t>::max());
    interest->SetNonce(rand.GetValue());
    interest->SetName(prefix);
    interest->SetInterestLifetime(Seconds(1.0));
    NS_LOG_DEBUG("Sending Interest packet for" << *prefix);
    m_transmittedInterests(interest, this, m_face);
    m_face->ReceiveInterest(interest);
}
```

Ptr<Object> xxx = Create<Object>(constructor)

# 2.4 Data

```
void ChatApp::SendData(std::string name, const std::string msg)
{
        NS_LOG_DEBUG("Sending data for " << name);
        Ptr<Packet> packet = Create<Packet>(msg);
        Ptr<ndn::Data> data = Create<ndn::Data>(packet);
        data->SetName(Create<ndn::Name>(name));
        m_transmittedDatas(data, this, m_face);
        m_face->ReceiveData(data);
}
```

```
int size = contentObject->GetPayload()->GetSize();
const unsigned char *data = contentObject->GetPayload()->PeekData();
std::string msg(reinterpret_cast<const char *>(data), size);
```

* ns3::ndn::Data
  - SetName(Ptr<Name>)
  - SetName(const Name &)
  - GetName()
  - GetPayload()

# 2.5 Application

```
class ChatApp : public ndn::App
{
public:
        static TypeId GetTypeId();
        void StartApplication();
        void StopApplication();
        void OnInterest(Ptr<const ndn::Interest> interest);
        void OnData(Ptr<const ndn::Data> contentObject);
```

```
/* create app */
ndn::AppHelper appHelper("ChatApp");
srand(time(NULL));
for (int i=0;i<chatterNodes.GetN();i++) {
    /* random start */
    appHelper.Install(chatterNodes.Get(i)).Start(Seconds(5*(double)rand()/RAND_MAX))
}
```

# 2.6 Start Simulation

```
/* simulator action */
Simulator::Stop(Seconds(20.0));
ndn::L3AggregateTracer::InstallAll("aggregate-trace.txt", Seconds(1.0));
ndn::L3RateTracer::InstallAll("rate-trace.txt", Seconds(1.0));
Simulator::Run();
Simulator::Destroy();
```

# As for me

- Skeleton
  - Message
  - MessagContainer
  - ChatApp
  - Main
- Simulation

# Message

```
class Message
{
public:
        void SetContent(const std::string&);
        void AddLabel(const std::string &name, time_t theTime);
        const std::string& GetContent();
        time_t GetTime(const std::string &name);
        const std::map<std::string, time_t> GetMap();
        std::string ToXML();

private:
        std::string m_content;
        std::map<std::string, time_t> m_label;

}; //Message
```

# MessageContainer

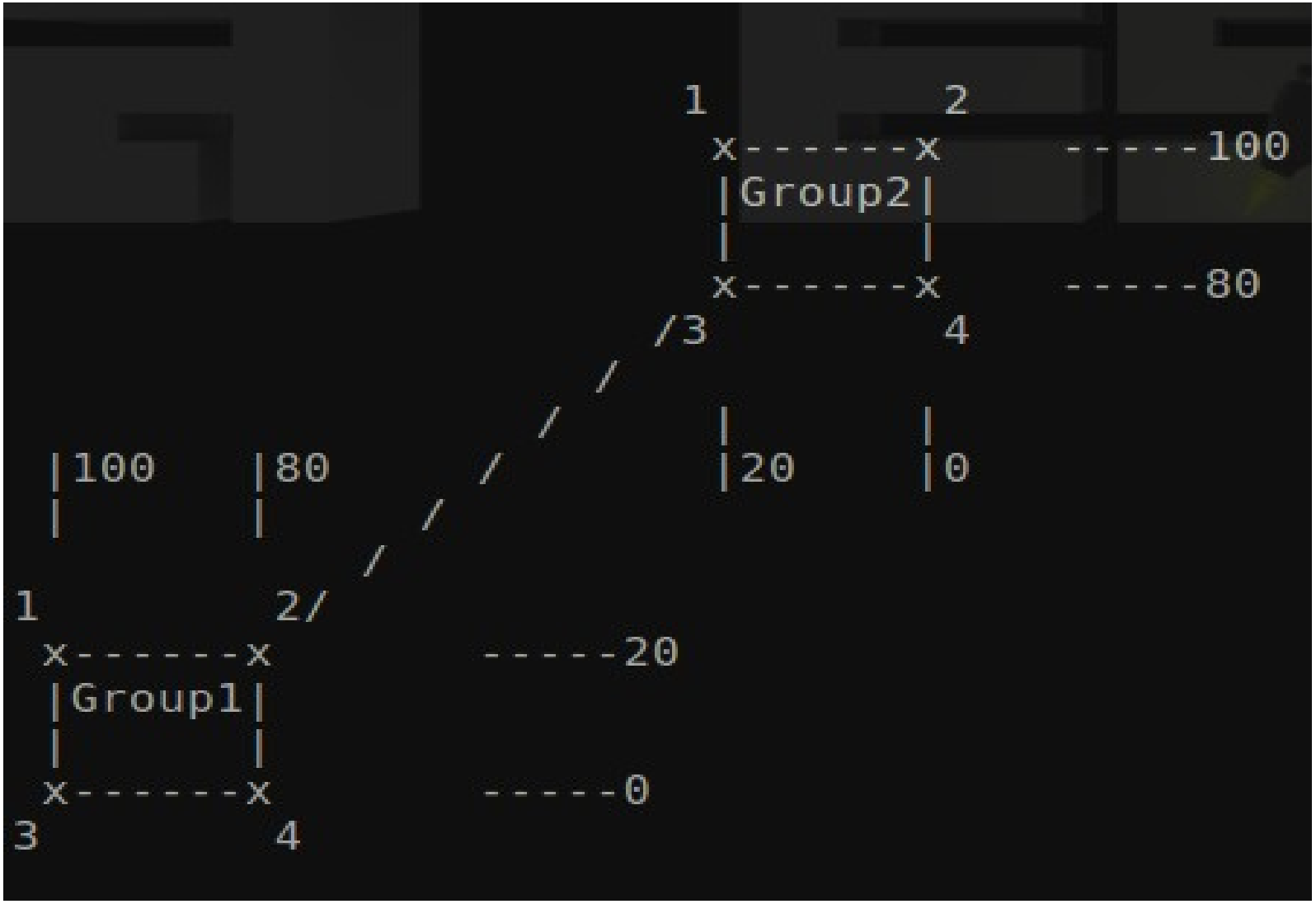
```
class MessageContainer
{
public:
        bool Add(Message msg);
        Message GetMessage(int index);
        std::string GetLatestLabels(std::string name);
        std::string GetLatestLabels();
        size_t Size();
        void AddIndex(std::string name, int index);
        std::vector<int> GetIndexVector(std::string);
        int GetIndexNewerThan(std::string name, time_t theTime);
        std::string ToXMLAfterIndex(int index);
private:
        std::vector<Message> m_messages;
        /*
         * name ----> index vector
         */
        std::map<std::string, std::vector<int> > m_record;
}; //MessageContainer
```

# ChatApp



```
 7 class ChatApp : public ndn::App
 8 {
 9 public:
10         static TypeId GetTypeId();
11         void StartApplication();
12         void StopApplication();
13         void OnInterest(Ptr<const ndn::Interest> interest);
14         void OnData(Ptr<const ndn::Data> contentObject);
15 private:
16         void ServerSelectPeriod();
17         void GenMessagePeriod();
18         void AnythingNewInterestPeriod();
19
20         void Config();
21         void Init();
22
23         void ProcessAllPendingInterests();
24         bool ProcessPendingInterest(Ptr<const ndn::Interest> interest);
25         void ProcessGiveMeInterest(Ptr<const ndn::Interest> interest, int index);
26
27         void ProcessAnyServerData(const Ptr<const ndn::Data> &contentObject);
28         void ProcessReceivedMessages(const Ptr<const ndn::Data> contentObject, bo
  methingnew);
29
30         void SendInterest(std::string name);
31         void SendAnythingNewInterest();
32         void SendSomethingNewInterest(int index);
33         void SendGiveMeInterest(std::string clientName, std::string index);
34
35         void SendData(std::string name, const std::string msg);
36         void GenMessage();
37         bool UpdateLocal(Message _msg);
38
39         virtual void SplitString(std::string &name, char indent);
```

# Simulation



# Server, anything new

```
0.730515s 7 ChatApp:StartApplication(): 0.730515
2.143s 5 ChatApp:StartApplication(): 2.143
2.73052s 7 ChatApp:GenMessage(): Gened message: I'm Node24 at Sun Dec 15 23:16:14 2013
2.99555s 7 ChatApp:GenMessage(): Gened message: I'm Node24 at Sun Dec 15 23:16:14 2013
3.1103s 5 ChatApp:SendInterest(): Sending Interest packet for /broadcast/chatapp/anyserver/0
3.12073s 7 ChatApp:OnInterest(): Received Interest packet for /broadcast/chatapp/anyserver/0
3.12073s 7 ChatApp:OnInterest(): Level: 0 My Level: 1
3.12073s 7 ChatApp:SendData(): Sending data for /broadcast/chatapp/anyserver/0
3.13126s 5 ChatApp:OnData(): Receiving Data packet for /broadcast/chatapp/anyserver/0
3.13126s 5 ChatApp:OnData(): DATA received for name /broadcast/chatapp/anyserver/0
3.13126s 5 ChatApp:ProcessAnyServerData(): My server changed to Node24
3.143s 5 ChatApp:SendInterest(): Sending Interest packet for /Node24/chatapp/anythingnew
3.1534s 7 ChatApp:OnInterest(): Received Interest packet for /Node24/chatapp/anythingnew
/Node24/chatapp/anythingnew
3.1534s 7 ChatApp:SendData(): Sending data for /Node24/chatapp/anythingnew
3.16411s 5 ChatApp:OnData(): Receiving Data packet for /Node24/chatapp/anythingnew
3.16411s 5 ChatApp:OnData(): DATA received for name /Node24/chatapp/anythingnew
3.36179s 7 ChatApp:GenMessage(): Gened message: I'm Node24 at Sun Dec 15 23:16:14 2013
3.47329s 0 ChatApp:StartApplication(): 3.47329
3.66602s 7 ChatApp:SendInterest(): Sending Interest packet for /broadcast/chatapp/anyserver/0
3.66602s 7 ChatApp:OnData(): Receiving Data packet for /broadcast/chatapp/anyserver/0
3.66602s 7 ChatApp:OnData(): DATA received for name /broadcast/chatapp/anyserver/0
3.66602s 7 ChatApp:ProcessAnyServerData(): My server changed to Node24
```

# Something new

```
6.04418s 5 ChatApp:GenMessage(): Gened message: I'm Node22 at Sun Dec 15 23:16:14 2013
6.04418s 5 ChatApp:SendInterest(): Sending Interest packet for /Node24/chatapp/somethingnew/Node22/5
6.05467s 7 ChatApp:OnInterest(): Received Interest packet for /Node24/chatapp/somethingnew/Node22/5
6.05467s 7 ChatApp:SendInterest(): Sending Interest packet for /Node22/chatapp/giveme/5
6.06506s 5 ChatApp:OnInterest(): Received Interest packet for /Node22/chatapp/giveme/5
6.06506s 5 ChatApp:SendData(): Sending data for /Node22/chatapp/giveme/5
6.07575s 7 ChatApp:OnData(): Receiving Data packet for /Node22/chatapp/giveme/5
6.07575s 7 ChatApp:OnData(): DATA received for name /Node22/chatapp/giveme/5
```