ns-3: Quick Intro and MANET WG Implementations

IETF 72

MANET WG

2k8/7/29

Presented by:

Joe Kopena

tjkopena@cs.drexel.edu

http://nsnam.org/

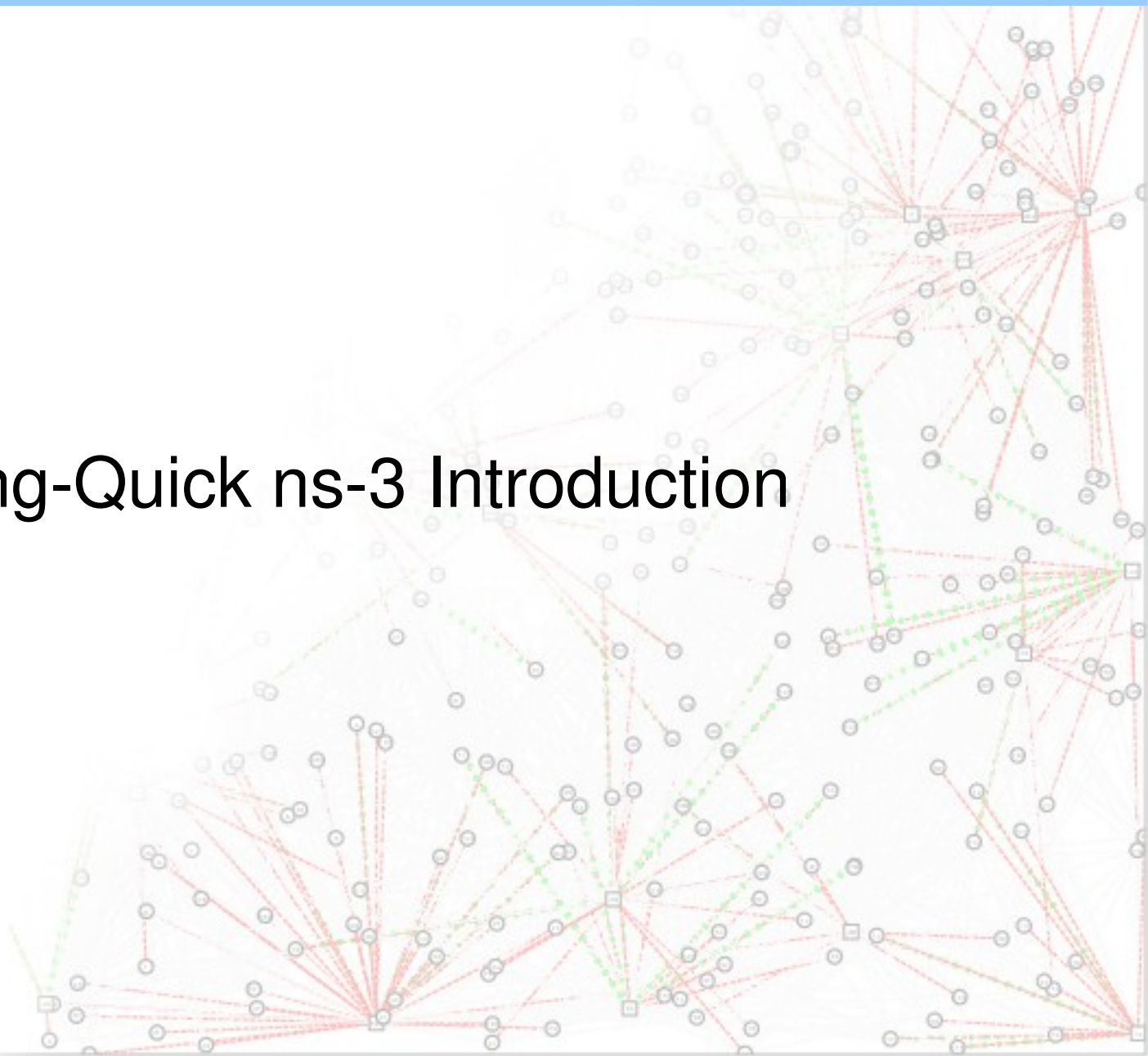
Presentation Contents

- Quick introduction to NS3
  - Highlight its capabilities, readyness for practical use
    - Steady trickle of papers using ns-3 starting to appear
  - ns-3 team is very eager to get more people using the system and contributing feedback or code
  - Demo table @ SIGCOMM 2008, come check it out!
- Notes on implementation of MANET specs for ns-3
  - Already released, done, in progress

(Presentation goal: ~25 minutes)

Part 1

Lightning-Quick ns-3 Introduction



Basic Facts

- ns-3: A new, NSF-primed, open source simulator for networking research and education
  - Clean slate design from ns-2, aiming to be easier to use and more ready for extension
- ns-3 core is written entirely in C++
  - User code---protocols and scenarios---also in C++
  - Python wrappers for user code also exist
- Library-based usage, no “ns-3 program” (yet)

`http://www.nsnam.org/getting_started.html`

Key Code Features

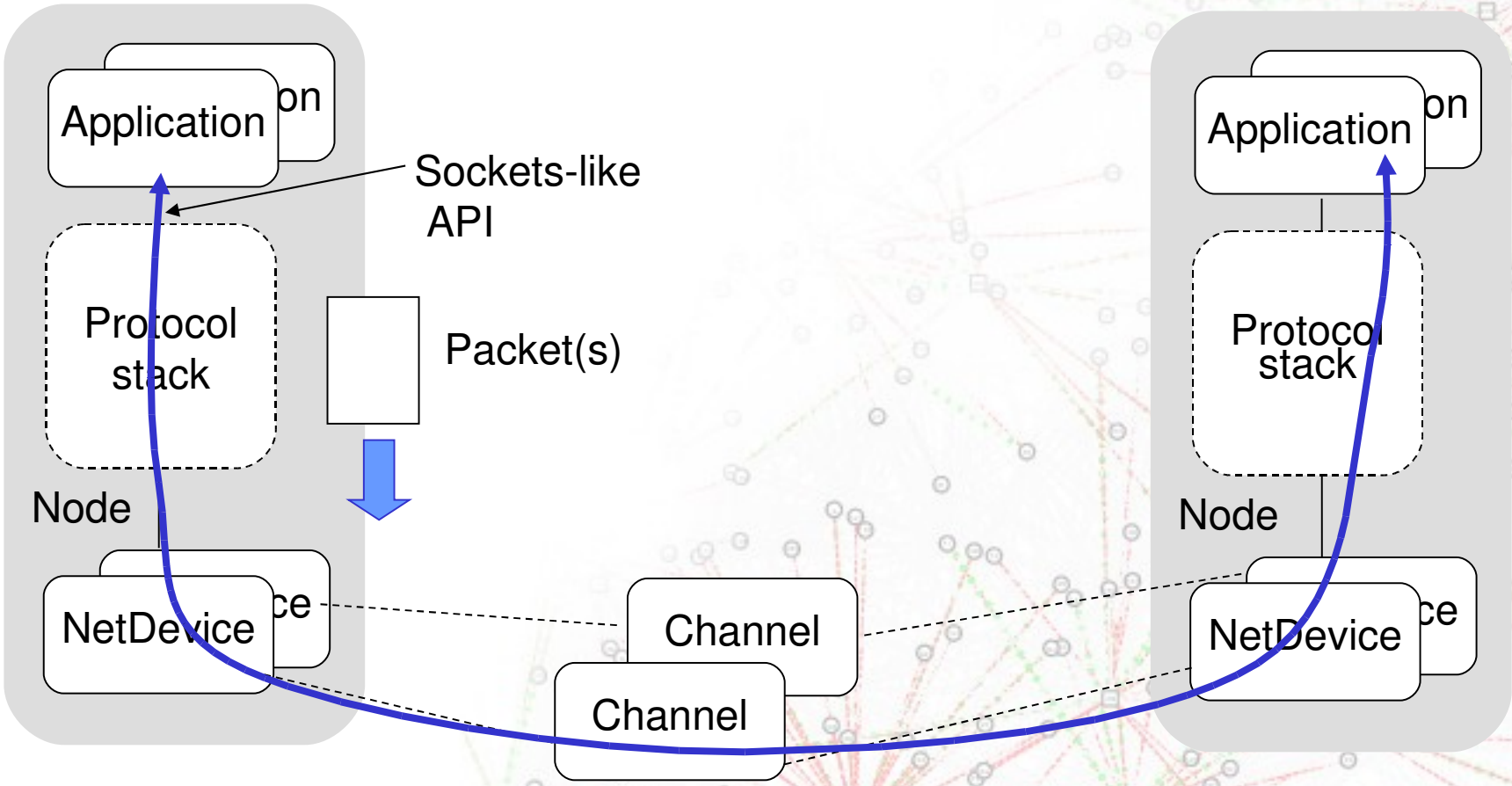
- Sophisticated simulation features included
  - Extensive parameterization system
  - Configurable embedded tracing system, with standard outputs to text logs or PCAP (tcpdump/wireshark)
- Object oriented design for rapid coding, extension
  - Automatic memory management
  - Object aggregation/query for new behaviors & state
    - E.g., adding mobility models to nodes

Key Simulation Features

- Models true IP stack, w/ potentially multiple devices & IP addresses on every node
- BSD-lookalike, event-based sockets API
  - Synchronous API alternative in-progress
- Packets include “wire formatted” (serialized) bytes, tags & metadata for easy extension and tracing

Simulation Network Architecture

- Looks just like IP architecture stack



Picture from Tom Henderson, SimuTools ns-3 Tutorial slides

Architecture Elements

- Nodes may/may not have mobility, other traits
- Nodes have “network devices,” e.g. WiFi, CSMA
  - NetDevices transfer packets over Channels
  - Incorporating Layer 1 (Physical) & Layer 2 (Link)
- Devices interface w/ Layer 3 (Network: IP, ARP)
- Layer 3 supports Layer 4 (Transport: UDP, TCP)
- Layer 4 is used by Layer 5 (Application) objects

Code Example 1/3

- Creating a WiFi, IP-based network:

```
NodeContainer nodes;  nodes.Create(g_numNodes);


WifiHelper wifi;
wifi.SetMac("ns3::AdhocWifiMac");  wifi.SetPhy("ns3::WifiPhy");
NetDeviceContainer nodeDevices = wifi.Install(nodes);


InternetStackHelper internet;
internet.Install(nodes);
Ipv4AddressHelper ipAddrs;
ipAddrs.SetBase("192.168.0.0", "255.255.0.0");
ipAddrs.Assign(nodeDevices);
```

Code Example 2/3

- Opening a socket:

```
InetSocketAddress addr(bcastAddr, m_bcastPort);

Ptr<SocketFactory> socketFactory = GetNode()->
    GetObject<SocketFactory>(UdpSocketFactory::GetTypeId());
m_socket = socketFactory->CreateSocket();

InetSocketAddress local = InetSocketAddress
  (Ipv4Address::GetAny(), m_bcastPort);
m_socket->Bind(local);  m_socket->Connect(addr);

m_socket->SetRecvCallback(MakeCallback(&Beacon::Receive, this));
```

Code Example 3/3

- Receiving a packet (callback set in previous slide):

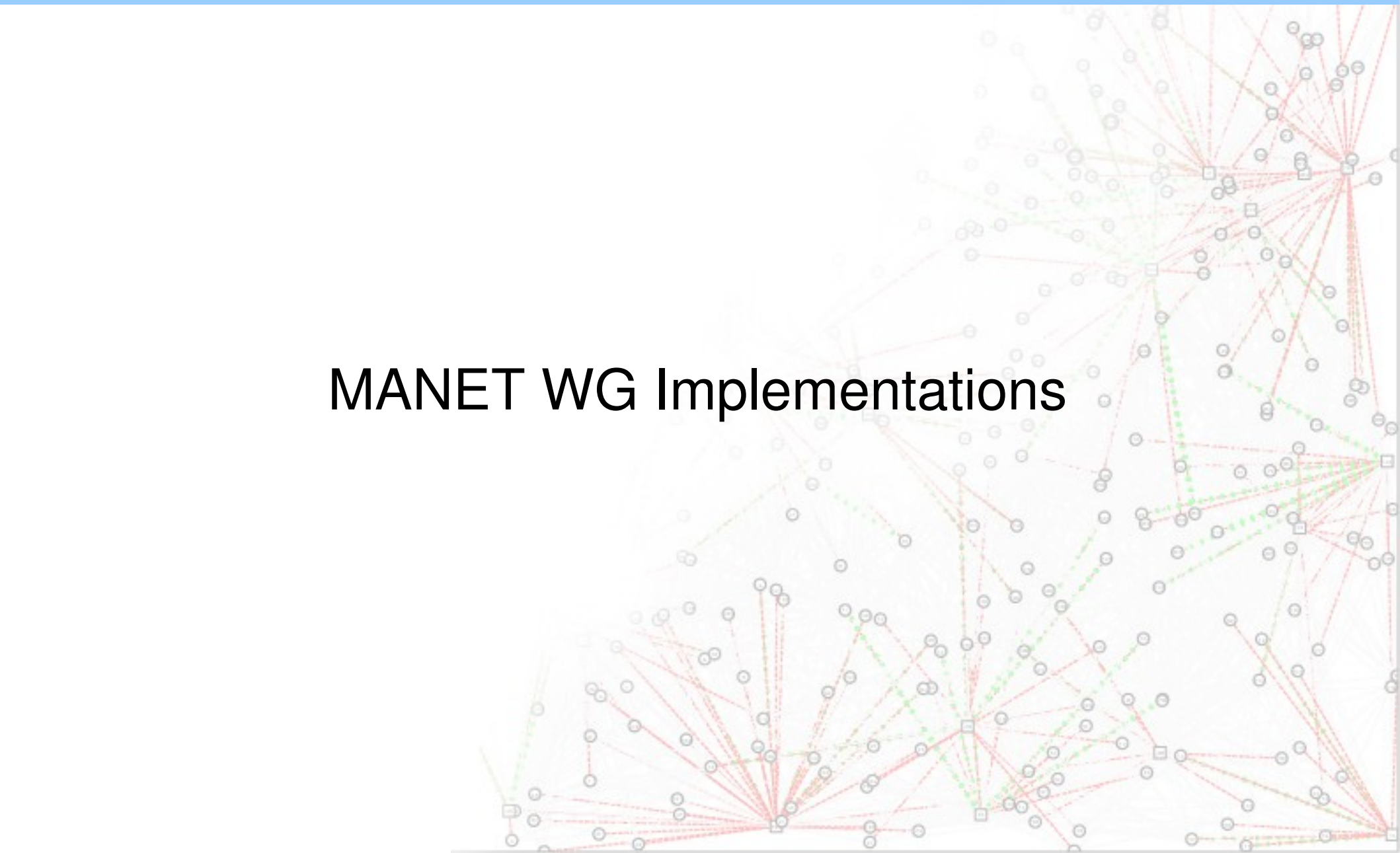
```
void Beacon::Receive(Ptr<Socket> socket) {
  Ptr<Packet> packet;
  Address from;
  while (packet = socket->RecvFrom(from)) {

    Ipv4Address ipv4From =
      InetSocketAddress::ConvertFrom(from).GetIpv4();
    NS_LOG_INFO("Received packet, " << packet->GetSize() <<
                " bytes from " << ipv4From);
    ...
```

Upcoming Features

- Known, established modules for near future:
  - NSC---Linux network stack ported into ns-3!
  - Ipv6---Integrated into ns-3 native network stack
  - Emulation---Run “simulations” in real-time over network
  - Statistics---Data collection, manipulation, visualization
  - Visualization---Watch network, application events
- And many more---apologies for any not listed!

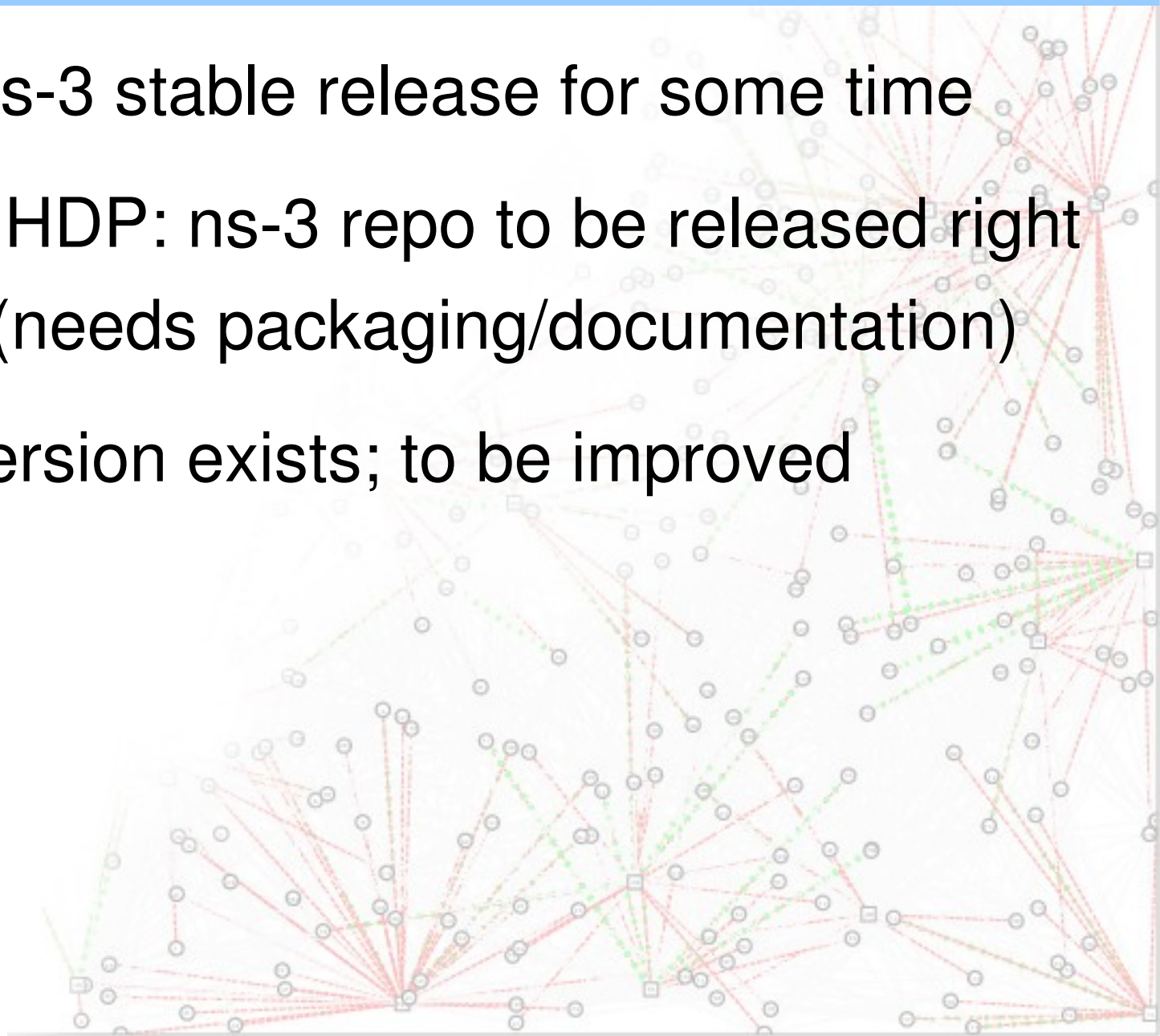
Part 2



MANET WG Implementations

MANET Implementations

- OLSR v1: In ns-3 stable release for some time
- PacketBB & NHDP: ns-3 repo to be released right after IETF 72 (needs packaging/documentation)
- SMF: Weak version exists; to be improved



OLSR

- OLSR v1 (RFC 3626), based on NS-2 code
  - Largely complete implementation of spec
    - ns-2 version by F. J. Ros, ns-3 port by G. Carneiro
  - NS-3 version: Supports multiple interfaces, does not support MAC layer feedback, HNA in-progress
- Installed simply using Helper class, e.g.:

```
OlsrHelper olsr;  olsr.InstallAll ();
```

PacketBB

- Recently updated to spec v13, latest ns-3 API
  - Latest PacketBB spec notably cleaner than previously
- Straightforward implementation
  - Ease of use, coding >> memory, processor use
  - Previous direct buffer version frustrating to use/develop
- Presents “query” interface
  - I.e.: Fetch TLVs by address, addresses by TLVs
    - But can control address block list, have empty addr TLVs, etc

NHDP

- Recently updated to spec v7, latest ns-3 API
- ns3::Application object which apps may access
  - NHDP object acts as “hub” of protocol architecture
    - Apps may register message type to receive
    - May provide message to include in next beacon

SMF

- Cheesy encapsulating version developed
  - ns3::Application object which presents Forward()
- Proper integration with ns-3 forwarding tables WIP

The End

- ns-3 main contact:
  - Tom Henderson <tomhend@u.washington.edu>
- Presentation questions/comments:
  - Joe Kopena <tjkopena@cs.drexel.edu>
- ns-3 quick start instructions:
  - `http://www.nsnam.org/getting_started.html`

`http://nsnam.org/`