A Performance and Scalability Evaluation of the NS-3 Distributed Scheduler

Ken Renard
Secure Mission Solutions
kdrenard2@gmail.com

Charles Peri
STG, Inc.
charles.peri@us.army.mil

Jerry Clarke
US Army Research Laboratory
jerry.clarke@us.army.mil

ABSTRACT

Network simulation of MANETs has substantial benefits for planning, engineering and research for military networks. Achieving high fidelity in network models, complex radio systems, and RF propagation effects results in significant computational loads for even moderately sized networks. Network simulators capable of performance and scalability for MANETs are highly valuable. In this paper, we share results of performance and scalability tests of the ns-3 network simulator in representative scenarios using ns-3 on the supercomputing platforms at the US Army Research Laboratory.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation – discrete event, distributed, parallel

General Terms

Algorithms, Performance, Design.

Keywords

MPI, network simulation, ns-3, distributed, performance, high performance computing.

1. INTRODUCTION AND MOTIVATION

The study of mobile ad-hoc networks has traditionally used various network simulation techniques to analyze protocol and network routing effects in various environments. The fidelity and accuracy of network simulation models such as RF propagation effects, network protocols, and mobility, are paramount to the credibility of these simulations. Achieving a high degree of fidelity at the full range of the network stack requires significant computational power for networks of even moderate size. In this paper, we present our results of performance studies using the NS-3 network simulator run on high-performance computing (HPC) platforms to demonstrate the capability of performing high-fidelity, scalable network simulations using high-performance computing.

WNS3 2012, March 23, Desenzano del Garda, Italy

DOI 10.4108/icst.simutools.2012.247679

The US Department of Defense (DoD) has long studied the performance of MANETs to gain a tactical advantage for deployed communications systems. By optimizing throughput, latency, and network capacity of MANETs, troops can reduce costs, improve situational awareness, and introduce new technical capabilities to the field. DoD mobile communications platforms employ complex, highly optimized routing and physical-layer capabilities that must perform with severe power constraints, limited spectrum availability, and multiple sources of interference. Modeling these radio waveforms and protocols requires significant code and computational power to accurately represent their behaviors. Additionally, the wide variety of radio-frequency (RF) environments expected for deployment require detailed modeling of RF effects that account for terrain, buildings, foliage, and atmospheric conditions.

The Mobile Network Modeling Institute (MNMI) at the Army Research Laboratory (ARL) is working to address the challenges of high-fidelity modeling of MANETs by exploiting the HPC capabilities of the DoD. Commercial simulation tools such as OpNet and QualNet were evaluated on HPC platforms. While these commercial tools have a wide range of models for specific military waveforms of interest, they did not demonstrate adequate scalability on HPC platforms. An open-source alternative for network simulation was sought that could support parallelism on HPC platforms, provide standard models of sufficient fidelity, and support custom development of models for military waveforms, applications, and high-fidelity RF propagation models. The target size of networks to be modeled is on the order of an entire division and larger (10,000+ network nodes). This capability will provide DoD researchers and planners with the ability to do analysis for acquisition, network and protocol research, and deployment optimization studies.

2. THE DISTRIBUTED NS-3 SCHEDULER

The ns-3 Distributed Scheduler was introduced in version 3.8 in May 2010. This scheduler implements a conservative parallel discrete event simulator where networks can be broken into "federates" across specialized point-to-point links. A full description of this work is in [1]. Communications between federates is via MPI where packets are serialized and delivered to

be scheduled as receive events on remote federates. Simulator timing is carefully controlled among federates such that causality is preserved given a minimum network latency (look ahead) between the partitioned sections of the simulated network. Each federate is allowed to process local events with perfect parallelism up to a specific time step that is agreed upon by all federates. Each federate then computes its next minimum time step where events will not be generated for any remote federate. The minimum of these collected values from all federates is granted time for each federate to process events in parallel.

The performance gains afforded by the distributed scheduler are highly dependent on the traffic profile and topology of the simulated network. If partitions (federates) of a network never communicate with each other, performance can be greatly improved by running completely independent simulations in parallel. As communication between federates increases, the amount of time granted between synchronizations tends to decrease, resulting in higher overhead for grant time negotiation. The minimum latency between connected nodes in different federates (look ahead) plays a significant role in the performance characteristics of the overall simulation. The larger the latency in the simulated network between federate boundaries, the larger the grant time can be, resulting in pure parallelism.

It is assumed that any event in a federate could directly or indirectly generate a message that is destined for another federate. Therefore, the grant time calculation is performed by adding the latency of the inter-federate network link with the offset in time to the next scheduled event in a federate. Each federate advertises and collects these time values to/from all other federates in a call to `MPI_AllGather()`. This (rather expensive) operation takes more time as the number of federates increases and the system interconnect latency increases.

Careful decomposition of network topologies into federates can be done to optimize performance by reducing inter-federate traffic, increasing inter-federate latency, and balancing computational load. In ns-3 versions to date (up to ns-3.12), federates can only be "split" along specialized point-to-point links. In many cases, a network can be artificially dissected into federates by inserting these point-to-point links transparently.

3. HARDWARE PLATFORMS

Two different systems were used for our performance and scalability studies. Initial testing was done on the “MJM” system and later continued on the “Harold” system. MJM and Harold were both High Performance Computing Systems at the U.S. Army Research Laboratory DoD Supercomputing Resource Center (DSRC).

MJM is Linux Networx Advanced Technology Cluster consisting of 1100 two processor socket, dual-core compute nodes, each with 8 GB of memory; and 8 two processor socket, dual-core login nodes, each with 16 GB of memory. The system utilizes a high speed 4x DDR Infiniband interconnect and global shared file system. The processors on MJM operate at a frequency of 3.0 GHz, resulting in a peak floating-point rate of 12.0 Gigaflops per core. In total, MJM consists of 4400 compute cores, delivering 52.8 Teraflops.

Harold is an SGI Altix ICE 8200 system, with 1344 compute nodes and is rated at 109.3 peak TFLOPS. Harold uses 2.8 GHz Intel Xeon X5560 Nehalem-EP processors on its login and compute nodes. There are 2 processors per node, each with 4 cores, for a total of 8 cores per nodes. Each processor has 4x256 Kbytes of L2 cache, and 8 Mbytes of L3 Cache. Each compute node has SUSE Linux Enterprise Server 11 SP1 installed as operating system, sharing 24 GB of DDR3 memory, with no user accessible swap space. Each compute node contains 20 GB of user accessible shared memory. Harold uses the 4X DDR Infiniband as its high-speed network for MPI messages and IO traffic. Harold uses Lustre to manage its parallel file system that targets SGI's IS4600 (Infinite Storage) RAID arrays.

4. PERFORMANCE SCENARIO

For performance analysis of the ns-3 distributed scheduler, a network scenario was developed that tried to balance realism and performance considerations. A hierarchical network configuration is somewhat representative of deployment scenarios where small teams may be connected via a shared, wireless, ad-hoc link. One of the team members will have an "uplink" capability that connects the team to others. This uplink could be a long-haul radio link or even a satellite link that has a large latency. Teams are assumed to be separated by enough space that they do not interfere with each other at the RF layer. Routing is hierarchical and network latencies between any two teams are the same. Within a team, an ad-hoc network routing protocol will direct traffic and advertise external routes.

Given this topology, it is easy to split the network into federates at team boundaries allowing a federate to contain 1 or more teams. Treating the uplink between teams as a satellite link with high latency allows the insertion of a point-to-point link as the connection between federates. Instead of 2 high-latency links (up to satellite and back down) plus a low-latency switching operation in the satellite, we treat this as a low-latency uplink, plus a very high latency switching operation, plus the low-latency downlink. We are essentially "moving" the latency from the up and down links into the satellite switch. This switch then becomes the point-to-point channel in ns-3 that is implemented via MPI.

The switching network inside the "satellite" ends up being a set of N*(N-1)/2 point-to-point links where N is the number of federates. While this grows exponentially with size, any single federate will only have to implement order N links.

The network within a team will use an ad-hoc 802.11 channel among itself to communicate and routing will be done with OLSR. While OLSR may not be representative of the exact routing protocols used in waveforms of interest, it does provide a representative work load in generating and processing messages that scales exponentially with the number of nodes in the OLSR domain. A significant amount of the simulation will be spent processing OLSR which is independent among teams. This means that there is a significant amount of computational load that

can be performed in parallel. The overhead of the 802.11 protocol along with the OLSR messaging provides a decent amount of traffic on the channel which must all pass through RF propagation effect processing.

Traffic load on the network is mostly provided by OLSR with the addition of a situational awareness application run on each node in the team. The application periodically reports its position (X,Y,Z vector as determined by the node's mobility model) in a UDP unicast message over IPv4 to the team "leader". The team leader application collects the position reports of all of its team members and sends out the full list of node positions as a UDP multicast message over IPv4 to all other team lead nodes across all federates. Individual position reports are sent every 10 seconds and aggregate reports are sent every 60 seconds.

All team nodes use a random walk mobility model inside a bounded region. Most tests were run using a 100m by 100m bounding box which pretty much ensured that all 802.11 nodes were within RF range of each other. Some tests were conducted in 1km by 1km and 10km by 10km regions specifically to reduce connectivity among nodes of the team which reduces OLSR overhead.

Routing outside of the ad-hoc team networks was static. In order to preserve a constant latency between teams whether the team was in the same federate or not, a route was established within a federate for traffic to be subjected to an arbitrary latency. All traffic leaving a team is sent to an "uplink" node which then routes packets to either a "mesh" node to be sent to another federate, or to a downlink node where it can then return directly to the destination team network.

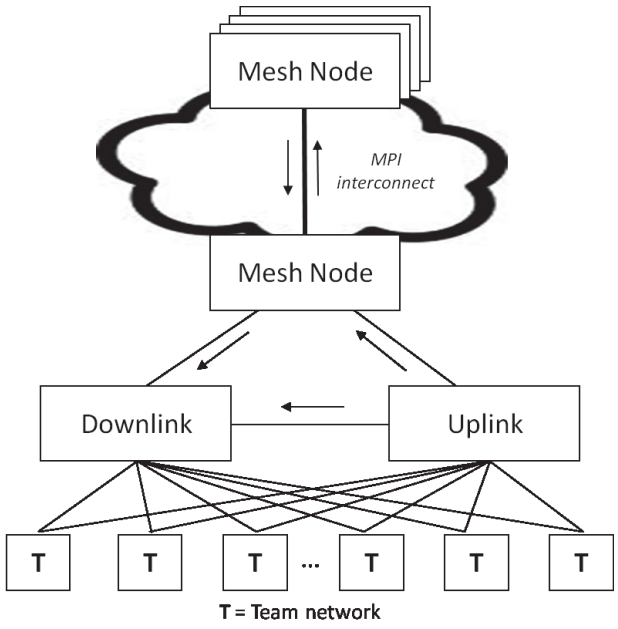

Figure 1. Routing Topology for each federate.

Each federate instantiates the desired number of team networks, its own uplink, downlink, and mesh nodes as well as mesh nodes for each other federate. Mesh nodes for other federates are simply "ghost" ns-3 nodes that are created with an m_systemId value that is not the same as the rest of the nodes in the federate. Traffic will appear to be sent and received by these nodes, but all packets on the links to/from the ghost nodes will actually come via MPI.

When packets are sent from one federate to another, the MPI message contains the node ID and interface ID of the receiving node. We must be careful to create mesh nodes in the exact same order in each MPI rank in order to preserve consistent node numbering. For example, each federate will create the mesh node for federate 0 as nodeID 0, the mesh node for federate 1 as nodeID 1, etc. This ensures that delivery of packets over the MPI link can be sent to the intended receiving node. Since we do not instantiate all links/interfaces for remote mesh nodes, we must reconcile the interface numbering such that packets arrive on the correct interface of the local mesh node. Federate X will only create one link/interface with Federate Y. When Federate X sends to Federate Y, the target interface on Federate Y's mesh node would always be interface 1. In such a case, all MPI packets received by a mesh node would be received on interface 1, regardless of which remote link they came from. This makes a difference in multicast routing where the source interface is part of the route entry.

5. PERFORMANCE RESULTS

The performance scenario was run on both hardware platforms described above with similar results, although presented in this paper was generated solely on 'MJM'. We first observed the scalability properties of the OLSR protocol where the number of nodes in a team (number of nodes participating in an independent OLSR routing domain) was varied. The compute time over various core counts each showed that the time increased exponentially with the number of nodes per team. This is expected as nodes were confined to a 100m x 100m 'box' where RF propagation between 802.11 nodes was unhindered. By increasing the size of the confining box to 1km x 1km and 10km x 10km, we saw the expected reduction in compute time due to failed propagation of OLSR packets. This demonstrated that OLSR was dominating the computational load of the simulation (this was seen on both 802.11 and CSMA networks). This is important for load balancing the work over multiple federates by ensuring that each federate has enough computational work to complete between synchronization intervals, thus improving the opportunity for parallelism. It is expected that the routing protocols and RF propagation effects will dominate computational load in military MANET simulations as well and that the performance properties demonstrated in the present scenarios will be similar.

We then measured performance of the simulator with respect to the number of CPU cores made available to the system. The metric for performance is packet transmissions per wall-clock time. In several trials with the given scenario, we observed nearly linear performance improvements by adding more cores (see fig. 2 and 3). We expect this linear performance up to a point where the workload per core is not significant enough to outweigh the time to synchronize simulation clocks among an increasing core count.

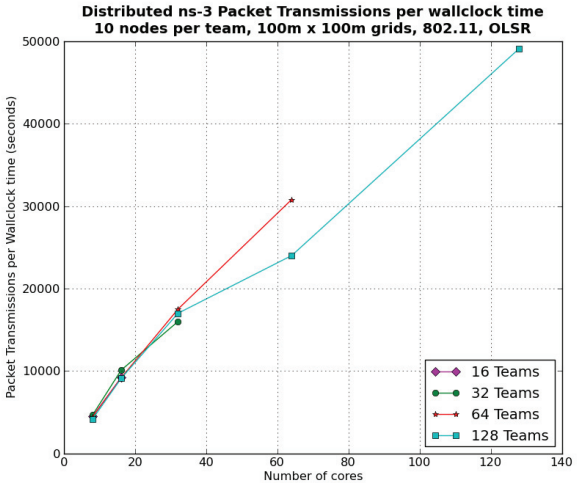

Figure 2. Simulator Performance with 10 nodes per team.

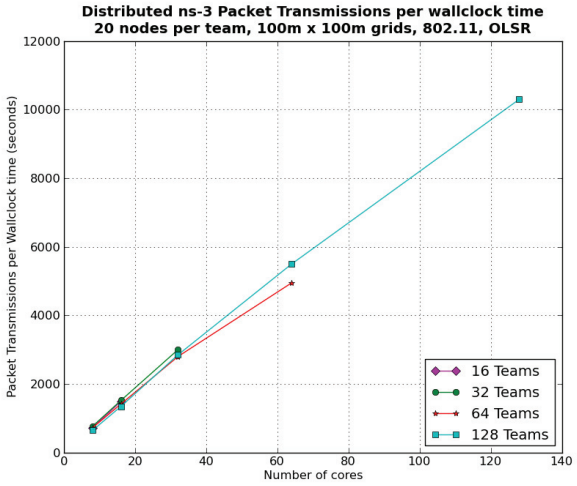

Figure 3. Simulator Performance with 20 nodes per team.

In order to demonstrate this, we needed to reduce the amount of work in each federate by replacing the 802.11 and OLSR processing overhead with CSMA and static routing models for the team subnets. With this significant decrease in work per grant time, the amount of time spent on synchronization (in `MPI_AllGather()`) became a more dominant factor in time and performance. We saw the linear performance fall off as core count increased beyond a limit (figures 4 and 5). This was further validated by using TAU (Tuning and Analysis Utility)[4] to observe the `MPI_AllGather()` performance over various core counts that consumed 7% to 20% of the simulation time. Given the complexity of MANET routing protocols used by the DoD and the level of fidelity required in the models at the routing and physical layers, we expect our use of the ns-3 simulator to perform nearly linearly with respect to core count in similar scenarios.

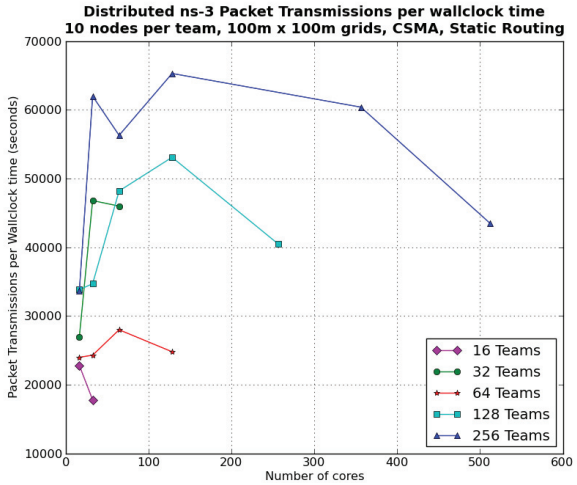

Figure 4. Simulator Performance with less computational load per federate (CSMA, Static Routing, 10 nodes per team).

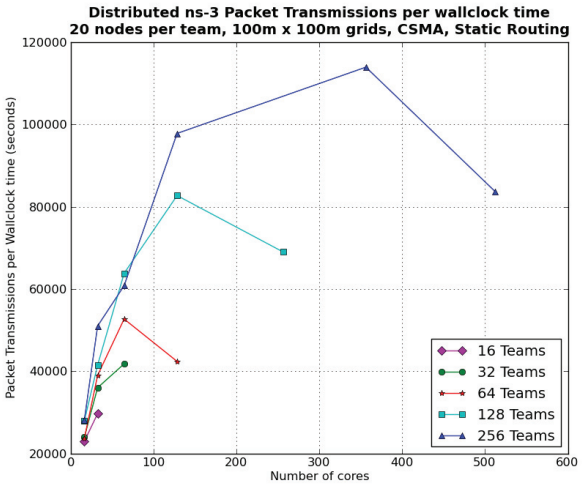

Figure 5. Simulator Performance with less computational load per federate (CSMA, Static Routing, 20 nodes per team).

6. SCALING SCENARIO

Another network simulation was created to test pure scalability of the ns-3 distributed scheduler to see how many nodes could be run in a single simulation. Although this is not much more than an exercise in finding a machine with a large amount of memory, it did demonstrate some potential issues with scaling on HPC platforms. The scenario was a simplification of the DARPA NMS (Network Modeling and Simulation)[3] in a pure hierarchical network format. There are 4 levels of hierarchy above a host (subnet, department, campus, federate) which are all connected via point-to-point links. The number of elements at each level of the hierarchy can be altered as a multiplier of the number of nodes. All federates have identical topologies and nodes have no mobility.

Traffic is generated by an application that sends 10 UDP packets of 1000 bytes each over a 10 second interval. Packets are sent to different "departments" in the same federate or to the same department in another federate. A service application on the receiving node simply counts the number of packets received and reports that data at application end. The percentage of nodes that run the sending application and the percentage of nodes that are transmitting across federate boundaries are controlled by command line arguments.

7. SCALING RESULTS

Several runs were completed on 'Harold' at various node counts and traffic application loads. Since this scenario has little computational load versus with 802.11 and OLSR scenario, the limiting factor in scaling was the ability to instantiate node and application objects in memory. It was found to be more efficient to run a single core per compute node with all available system memory versus running additional cores with smaller amounts of memory per compute node. We expect that a more realistic scenario will quickly tip the scales to favor the use of more cores.

In the largest of simulations, a total of 176 compute nodes, each using 1 CPU core and 17.96 GB of memory, was able to simulate 360,448,000 nodes with 40% of hosts transmitting (144M), and 1% of those (1.4M) were transmitting across federate boundaries. This achieved a packet transmit rate of 413,704.52 packets per wall-clock second.

IPv6 addressing was used throughout the network which made allocation to such a large network much easier. It was necessary to build the hooks into ns-3 to connect the IPv6 layer to the TCP and UDP layers. Initially, this was done in a "quick-and-dirty" fashion to run the scalability tests. Since then, we have worked with core ns-3 developers to come up with a more robust and clean way of adding transport services via IPv6. These patches have been submitted for inclusion in to future releases of ns-3.

8. FUTURE WORK

The ns-3 simulator has shown significant promise as a scalable, high-performance network simulator capable of supporting high-fidelity network and physical layer models. The MNMI plans to continue its use of ns-3 to implement models for military waveforms on HPC platforms to conduct simulations of large-scale MANETs. There is also interest in combining the distributed scheduler with the real-time scheduler to achieve highly scalable real-time network emulations with ns-3 (using the "emu" net device). We would like to investigate the effects of inter-federate traffic loading on the MPI communications layer and resulting simulation performance. We are also interested in automating and optimizing the federation of network topologies to possibly include the movement of nodes between federates. A promising technology that could be used is METIS[2] to enable pre-simulation federation as well as in-simulation re-federation of the network. We would like to see a more flexible method of creating distributed point-to-point links such that federate, node, and interface identifiers of the remote endpoint can be specified arbitrarily. This would eliminate the need for special ordering of node creation, although care would still be required of the simulation designer to ensure that endpoints are "connected" appropriately. Finally, we plan to investigate the use of a unique distributed shared memory facility to couple various components of a complex simulation.

9. ACKNOWLEDGMENTS

This work was made possible by a grant of computer time and resources from the DoD High Performance Computing Modernization Program (HPCMP). Our thanks go to the core development team and contributors to the ns-3 project. Special thanks are due to George Riley and Josh Pelkey for their work on the distributed scheduler.

10. REFERENCES

[1] Pelkey, Joshua, Riley, George, 2011. *Distributed Simulation with MPI in ns-3*. Workshop in ns-3, Barcelona, Spain.

[2] "Family of Graph and Hypergraph Partitioning Software", Karypis Lab, University of Minnesota, http://glaros.dtc.umn.edu/gkhome/views/metis

[3] Standard baseline NMS challenge topology, http://www.ssfnet.org/Exchange/gallery/baseline, July 2002.

[4] Tuning and Analysis Utilities (TAU), Performance Research Lab, University of Oregon, http://www.cs.uoregon.edu/Research/tau/home.php