

# NS2 versus NS3

## Comparison between NS2 and NS3:

	NS2	NS3
<b>Programming languages</b>	<ul style="list-style-type: none"> <li>• NS2 is implemented using a combination of oTCL (for scripts describing the network topology) and C++ (The core of the simulator).</li> <li>• This system was chosen in the early 1990s to avoid the recompilation of C++ as it was very time consuming using the hardware available at that time, oTCL recompilation takes less time than C++.</li> <li>• <u>oTCL disadvantage</u>: there is overhead introduced with large simulations.</li> <li>• oTCL is the only available scripting language.</li> </ul>	<ul style="list-style-type: none"> <li>• NS3 is implemented using C++</li> <li>• With modern hardware capabilities, compilation time was not an issue like for NS2, NS3 can be developed with C++ entirely.</li> <li>• A simulation script can be written as a C++ program, which is not possible in NS2.</li> <li>• There is a limited support for Python in scripting and visualization.</li> </ul>
<b>Memory Management</b>	<ul style="list-style-type: none"> <li>• NS2 requires basic manual C++ memory management functions.</li> </ul>	<ul style="list-style-type: none"> <li>• Because NS3 is implemented in C++, all normal C++ memory management functions such as new, delete, malloc, and free are still available.</li> <li>• Automatic de-allocation of objects is supported using reference counting (track number of pointers to an object); this is useful when dealing with Packet objects.</li> </ul>
<b>Packets</b>	<ul style="list-style-type: none"> <li>• A packet consists of 2 distinct regions; one for headers, and the second stores payload data.</li> <li>• NS2 never frees memory used to store packets until the simulation terminates, it just reuses the allocated packets repeatedly, as a result, the header region of any packet includes all headers defined as part of the used protocol even if that particular packet won't use that particular header, but just to be available when this packet allocation is reused.</li> </ul>	<ul style="list-style-type: none"> <li>• A packet consists of a single buffer of bytes, and optionally a collection of small tags containing meta-data.</li> <li>• The buffer corresponds exactly to the stream of bits that would be sent over a real network.</li> <li>• Information is added to the packet by using subclasses; Header, which adds information to the beginning of the buffer, Trailer, which adds to the end.</li> <li>• Unlike NS2, there is generally easy way to determine if a specific header is attached.</li> </ul>

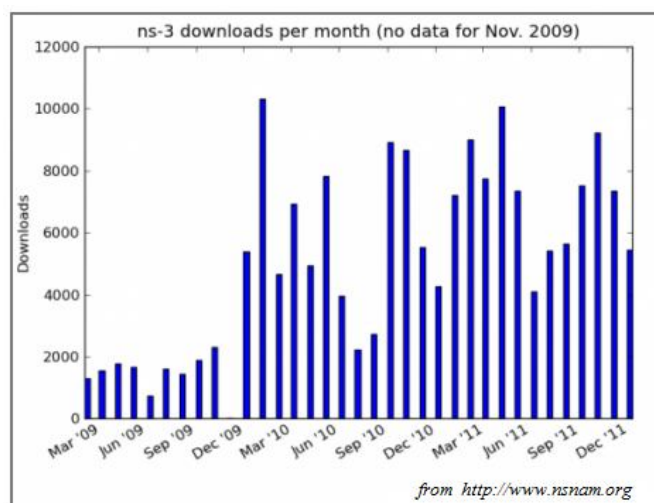
<b>Performance</b>	<ul style="list-style-type: none"> <li>• The total computation time required to run a simulation scales better in NS3 than NS2.</li> <li>• This is due to the removal of the overhead associated with interfacing oTcl with C++, and the overhead associated with the oTcl interpreter.</li> </ul>	<ul style="list-style-type: none"> <li>• NS3 performs better than NS2 in terms of memory management.</li> <li>• The aggregation system prevents unneeded parameters from being stored, and packets don't contain unused reserved header space.</li> </ul>
<b>Simulation output</b>	<ul style="list-style-type: none"> <li>• NS2 comes with a package called NAM (Network Animator), it's a Tcl based animation system that produces a visual representation of the network described.</li> </ul>	<ul style="list-style-type: none"> <li>• NS3 employs a package known as PyViz, which is a python based real-time visualization package</li> </ul>

### Important notes about NS3:

- NS3 is not backward compatible with NS2; it's built from the scratch to replace NS2.
- NS3 is written in C++, Python Programming Language can be optionally used as an interface.
- NS3 is trying to solve problems present in NS2.
- There is very limited number of contributed codes made with NS3 compared to NS2
- In NS2, bi-language system make debugging complex (C++/Tcl), but for NS3 only knowledge of C++ is enough (single-language architecture is more robust in the long term).
- NS3 has an emulation mode, which allows for the integration with real networks.

### Growth of NS3:

- In 2009, ns-3 releases were downloaded around 14,000 times. In 2011, ns-3 releases were downloaded 86,014 times.
- As of ns-3.13 release, 93 people are listed as authors.
- Figure 1 describes the number of downloads per month from 2009 to 2011.



**Figure 1: NS3 statistics**

## NS2 and NS3 existing models:

- A lot of different organizations contributed to the models and components of NS2, it has the strongest asset in terms of contributed code as shown in Figure 2.
- There is very limited number of models and contributed codes in NS3 in comparison with NS2 as shown in Figure 3.
- Since NS3 is under development, it still requires strong community participation to add contributed codes and to improve it.

	<b>Existing core ns-2 capability</b>	<b>ns-2 contributed code</b>
Applications	ping, vat, telnet, FTP, multicast FTP, HTTP, probabilistic and trace-driven traffic generators, webcache	NSWEB, Video traffic generator, MPEG generator, BonnTraffic, ProtoLib, AgentJ, SIP, NSIS, ns2voip, Agent/Plant
Transport layer	TCP (many variants), UDP, SCTP, XCP, TFRC, R&P, RTP Multicast: PGM, SRM, RLM, PLM	TCP PEP, SCPS-TP SNACK, TCP Pacing, DCCP, Simulation Cradle, TCP Westwood, SIMD, TCP-RH, MFTP, OTERS, TCP Eifel
Network layer	Unicast: IP, MobileIP, generic dist. vector and link state, IPinIP, source routing, Nixvector Multicast: SRM, generic centralized MANET: AODV, DSR, DSDV, TORA, IMEP	AODV+, AODV-UU, AOMDV, ns-click, ZRP, IS-IS, CDS, Dynamic Linkstate, DYMO, OLSR, ATM, AntNet, Mobile IPv6, IP micro-mobility, MobileIP, GPSR, RSVP, PGM, PLM, SSM, PUMA, ActiveNetworks
Link layer	ARP, HDLC, GAF, MPLS, LDP, Diffserv Queueing: DropTail, RED, RIO, WFQ, SRR, Semantic Packet Queue, REM, Priority, VQ MACs: CSMA, 802.11b, 802.15.4 (WPAN), satellite Aloha	802.16, 802.11e HCCA, 802.11e EDCA, 802.11a multirate, UWB DCC-MAC, TDMA DAMA, EURANE, UMTS, GPRS, BlueTooth, 802.11 PCF, 802.11 PSM, MPLS, WFQ schedulers, Bandwidth Broker, CSFQ, BLUE
Physical layer	TwoWay, Shadowing, OmniAntennas, EnergyModel, Satellite Repeater	ET/SNRT/BER-based Phy, IR-UWB
Support	Random number generators, tracing, monitors, mathematical support, test suite, animation (nam), error models	Emulation, CANU mobility, BonnMotion mobility, SGB Topology Generators, NSG2, simd, ns2measure, ns-2/akaroa-2, yavista, tracegraph, huginn, multistate error model, RPI graphing package, jTrana, GFA,

*From [http://www-npa.lip6.fr/~rehmani/ns3\\_v1.pdf](http://www-npa.lip6.fr/~rehmani/ns3_v1.pdf)*

**Figure 2:** NS2 contributed code (in July 2010)

	Existing core ns-2 capability	Existing ns-3
Applications	ping, vat, telnet, FTP, multicast FTP, HTTP, probabilistic and trace-driven traffic generators, webcache	OnOffApplication, asynchronous sockets API, packet sockets
Transport layer	TCP (many variants), UDP, SCTP, XCP, TFRC, RAP, RTP Multicast: PGM, SRM, RLM, PLM	UDP, TCP
Network layer	Unicast: IP, MobileIP, generic dist. vector and link state, IPinIP, source routing, Nixvector Multicast: SRM, generic centralized MANET: AODV, DSR, DSDV, TORA, IMEP	Unicast: IPv4, global static routing Multicast: static routing MANET: OLSR
Link layer	ARP, HDLC, GAF, MPLS, LDP, Diffserv Queueing: DropTail, RED, RIO, WFQ, SRR, Semantic Packet Queue, REM, Priority, VQ MACs: CSMA, 802.11b, 802.15.4 (WPAN), satellite Aloha	PointToPoint, CSMA, 802.11 MAC low and high and rate control algorithms
Physical layer	TwoWay, Shadowing, OmniAntennas, EnergyModel, Satellite Repeater	802.11a, Friis propagation loss model, log distance propagation loss model, basic wired (loss, delay)
Support	Random number generators, tracing, monitors, mathematical support, test suite, animation (nam), error models	Random number generators, tracing, unit tests, logging, callbacks, mobility visualizer, error models

*From [http://www-npa.lip6.fr/~rehmani/ns3\\_v1.pdf](http://www-npa.lip6.fr/~rehmani/ns3_v1.pdf)*

**Figure 3:** NS2 and NS3 existing core capabilities (in July 2010)

**[References]:**

- <http://www.nsnam.org>
- [http://www-npa.lip6.fr/~rehmani/ns3\\_v1.pdf](http://www-npa.lip6.fr/~rehmani/ns3_v1.pdf)
- <http://cse.wustl.edu/Research/Lists/Technical%20Reports/Attachments/954/NS-3%20Simulation%20of%20WiMAX%20Networks.pdf>
- <http://www.nps.edu/Academics/Institutes/Cebrowski/Docs/RileyNPS-2010.pdf>