NS2 versus NS3

Comparison between NS2 and NS3:

	NS2	NS3
Programming languages	• NS2 is implemented using a combination of oTCL (for scripts describing the network topology) and C++ (The core of the simulator). • 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++. • oTCL disadvantage: there is overhead introduced with large simulations. • oTCL is the only available scripting language.	• NS3 is implemented using C++ • With modern hardware capabilities, compilation time was not an issue like for NS2, NS3 can be developed with C++ entirely. • A simulation script can be written as a C++ program, which is not possible in NS2. • There is a limited support for Python in scripting and visualization.
Memory Management	• NS2 requires basic manual C++ memory management functions.	• Because NS3 is implemented in C++, all normal C++ memory management functions such as new, delete, malloc, and free are still available. • 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.
Packets	• A packet consists of 2 distinct regions; one for headers, and the second stores payload data. • 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.	• A packet consists of a single buffer of bytes, and optionally a collection of small tags containing meta-data. • The buffer corresponds exactly to the stream of bits that would be sent over a real network. • 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. • Unlike NS2, there is generally easy way to determine if a specific header is attached.

Performance	• The total computation time required to run a simulation scales better in NS3 than NS2. • This is due to the removal of the overhead associated with interfacing oTcl with C++, and the overhead associated with the oTcl interpreter.	• NS3 performs better than NS2 in terms of memory management. • The aggregation system prevents unneeded parameters from being stored, and packets don't contain unused reserved header space.
Simulation output	• 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.	• NS3 employs a package known as PyViz, which is a python based real-time visualization package

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.

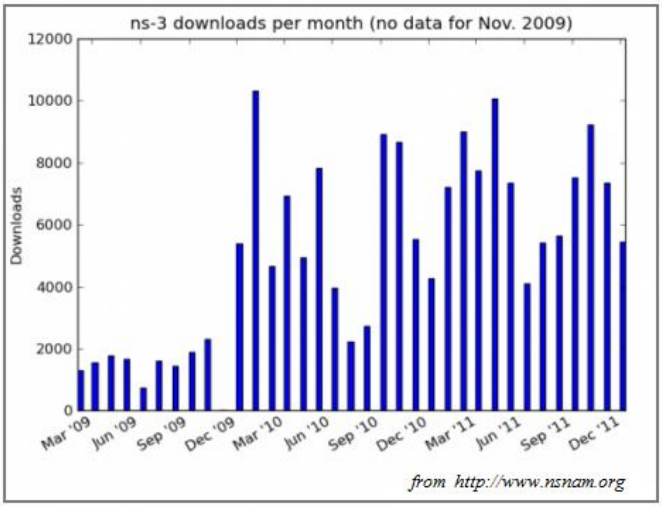

Figure 1: NS3 statistics

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.

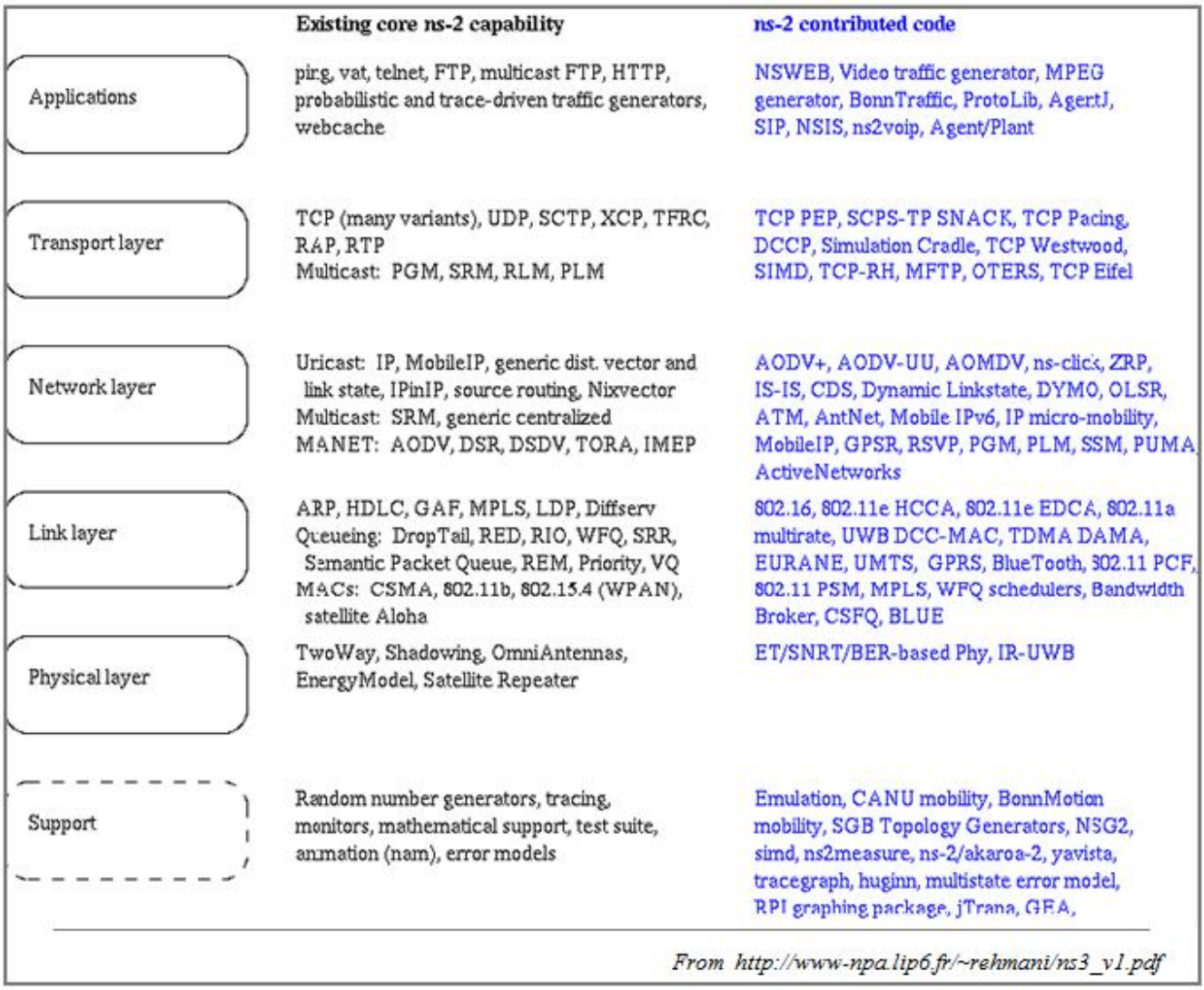

Figure 2: NS2 contributed code (in July 2010)

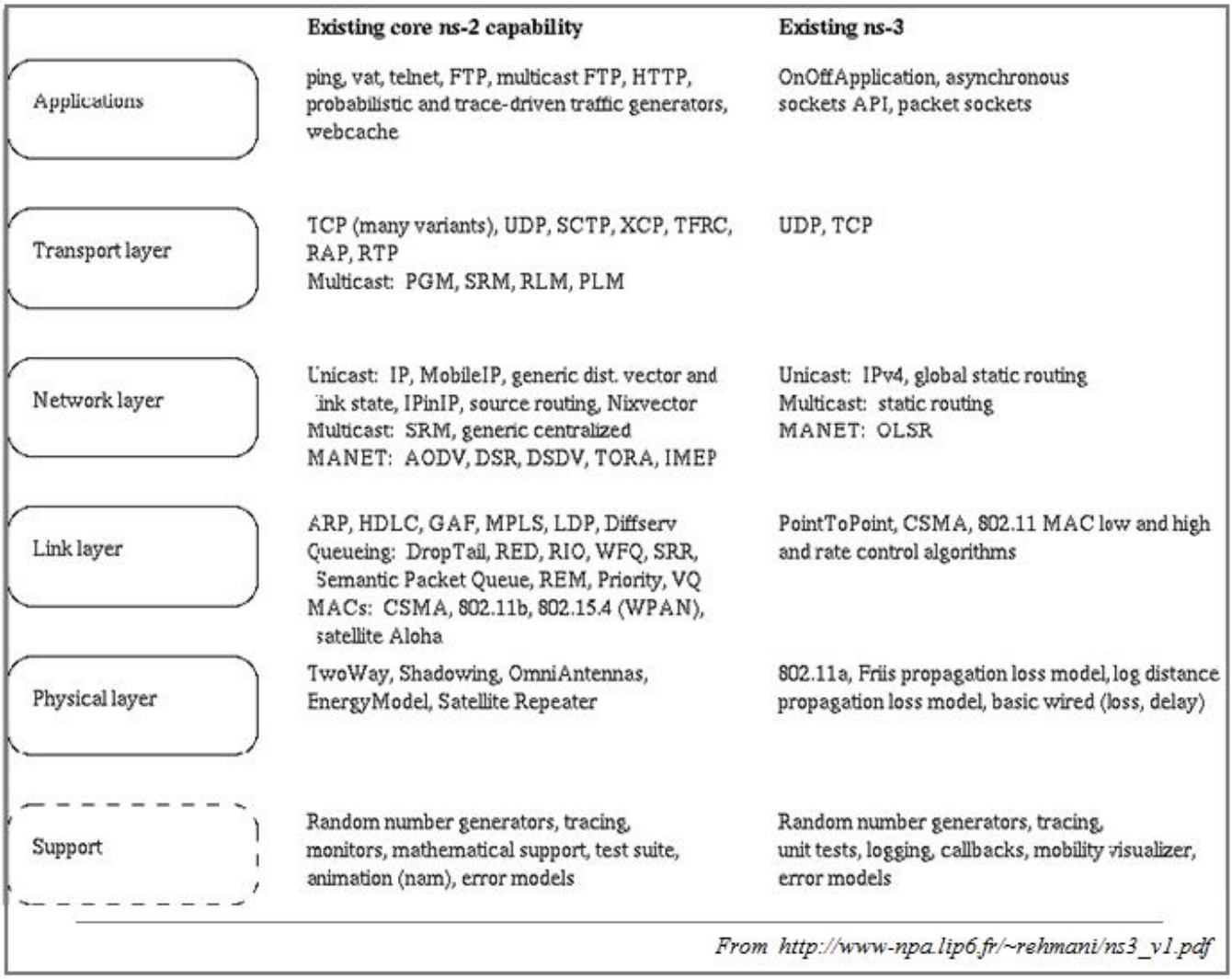

Figure 3: NS2 and NS3 existing core capabilities (in July 2010)

[References]:

- http://www.nsnam.org
- http://www-npa.lip6.fr/~rehmani/ns3_v1.pdf
- http://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