Speed & Scalability: 
Requirements for Industrial Strength 
Network Simulators

Waiting Days and Weeks for Simulations is Not Acceptable
Decreasing Model Complexity is Problematic

How Speed is Achieved
  A Small but Specialized Kernel
  Unstructured vs. Structured Kernel Extension

QualNet’s Stack Architecture

Relationship between Speed and Scalability

Scalability Techniques
  Pooled Memory
  Global Variables Used Sparingly
  Memory Speed Relative to Processor Speed
  Good Memory Management in QualNet

Summary

References

About Scalable Network Technologies
Modelers use network simulation to assess the performance of networking technologies in experimental scenarios. A lot of number crunching is required to answer questions such as "How will the latency of my video conferencing application be affected if I add 50 new users to the network?" or "How will the response time of my web server be impacted if my traffic mix changes dramatically?"

The process of running simulations is often prohibitively long. Historically, it would take days to execute a one-minute simulation of a high-fidelity model of a large wired network with heavy traffic and 1000 nodes. The same applies to a wireless network with complex propagation models and thousands of nodes and multiple types of traffic. In addition, the random nature of network traffic requires multiple trials for a single reliable data set. To ensure confidence in simulation results, 10, 20 or more trials may be performed. In the past, a single reliable data set could take weeks or even months to produce.

With simulation periods that long, modelers historically had two options:

1. Wait days or weeks for good results, delaying the project.
2. Decrease the complexity of the model to reduce execution time.
1. Waiting Days or Weeks for Simulations Is Not Acceptable

12-hour Threshold

The first option is not very realistic. Simulation projects often require custom coding of network protocol models, taking additional months of the modeler’s time. Adding more months of model execution is prohibitively expensive. As a rule of thumb, to keep productivity high in a simulation project, models should take no longer than 12 hours to run. Any run that cannot be finished by the next morning will severely limit the productivity of the modeler.

Real-time Simulation

The demand for real-time simulation is another important factor for determining the necessary speed of simulation. Real-time simulation is achieved when one second of simulation data takes at most one second of run time to be calculated. QualNet, a network simulation tool, achieves real time simulation speeds. Real-time network simulation means the network can be modeled in parallel with other tools. For instance, OTB/ModSAF is a real-time war-fighting simulator that models realistic unit behaviors in a wide-reaching strategic view of a simulated battle. OTB/ModSAF can be hooked into QualNet, a network simulator that models communications to support combat. Communication results are calculated on the fly by QualNet and returned to the war simulator in High Level Architecture* (HLA) format for a continual feedback loop.

Speed Impacts Scalability

Another reason why speed is important is how it impacts scalability. With advances in networking technologies, there has been a trend towards larger networks made up of more components. Modelers have found that large networks behave much differently than small networks. In other

* For additional information on ModSAF/OTB and HLA, please refer to the SNT white paper entitled “Network Centric Warfare”.

Figure 1: QualNet real-time results for a high fidelity 100-node wireless model
words, it is not possible to extrapolate the performance of large networks simply from simulations of small networks. When a model is large, i.e., there are a lot of devices (workstations, routers, hosts, links, etc.) and traffic sources (many users and numerous varieties of network traffic), the number of events the simulator must process increases. Thus, real-time simulations and large network simulations require the same thing: a simulation engine that processes events fast. We will discuss the requirements for fast and scalable software design later in the paper.

Real-time simulations and large network simulations require the same thing: a simulation engine that processes events fast

2. Decreasing Model Complexity is Problematic

Unrealistic Network Behavior

To increase the speed of simulation, many modelers choose to decrease the complexity of the model. This tactic often yields inaccurate results because the model lacks realistic networking behavior due to missing functionality. What's more, to eliminate functionality from a model requires the modeler to measure its impact and prove that it is negligible to the final results of the simulation. It is difficult and time-consuming to isolate individual details of protocol models and prove that they are superfluous to the simulation. Therefore, high detail models are usually a safer bet.

Small Details Make a Large Impact

In a study that illustrates the importance of high fidelity simulation, something as small as the physical layer preamble length in a wireless communication device was shown to affect
packet delivery significantly in Mobile Ad hoc Networks (MANET). The physical layer preamble in the IEEE 802.11 standard is specified in microseconds, regardless of the data rate. For a simulation using the AODV routing protocol, the difference between a 192-microsecond preamble versus a 96-microsecond preamble meant a reduction in network-wide packet delivery by 7%[2]. Results must be accurate or the information is useless.

Discrete Event Simulation (DES) is a computer model of some physical system, where the state of the system is assumed to change only at discrete points in simulated time. In DES, to trade speed for high fidelity, or accuracy for fast results, yields unacceptable results in both cases. In response to this problem, software developers built QualNet, a specialized network simulation engine that is fast enough to ensure that neither fidelity nor scalability is sacrificed.

We have just described the problems historically associated with network simulation. The rest of this white paper discusses the science behind QualNet’s speed and scalability. Faster processors and new programming methods have raised the bar in terms of fast run times and scalability. QualNet exploits the following characteristics of DES to achieve superior performance:

1. Speed requires a small kernel and smart, efficient architecture.
2. Speed and scalability are directly related.
3. Scalability requires smart memory management.

QualNet is a specialized network simulation engine that is fast enough to ensure that neither fidelity nor scalability is sacrificed.

Figure 3: High fidelity models are necessary for accurate results. The Packet Delivery Ratio is shown to be heavily dependent on the routing protocol and physical layer preamble length. Ad hoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR) were the two routing protocols compared. AODV showed much higher PDR, but it was also highly susceptible to packet drops when the physical layer preamble length was increased.
How Speed is Achieved

Speed of simulation relates to the time it takes to execute a model. Software-based network simulators are usually DES models. DES is a computer model of some physical system, where the state of the system is assumed to change only at discrete points in simulated time. The main activity in DES is the processing of a global event list, where events are kept in chronological order. An example of an event is a packet arriving at a node. With seven layers in the network stack, each performing complex processes on packets passing through, the processing load for DES is heavy. A network simulation’s job is made even harder by the fact that networks themselves are increasing in size and complexity. Even taking into consideration Moore’s Law of processor speeds doubling every 18 months, advancements in processor speeds are not significant enough to speed up network simulation. Real networks are increasing in complexity too fast.

In DES, the kernel takes events off the global event list and processes them. The architecture of the software program, or the main workflow for processing events, is the single most critical element for speed of simulation.

A Small but Specialized Kernel

The kernel, or core loop in the simulation application, must use processor cycles efficiently. The kernel is an event list scheduling and management mechanism that is at the heart of all DES. It schedules the next event, processes it, and inserts other events into the list. Designing an efficient kernel is like designing an assembly line. Processes on the critical path get priority and dictate how secondary processes are structured. With this objective in mind, the kernel is designed to be as small and efficient as possible.

Unstructured vs. Structured Kernel Extensions

Next, to build a model, the kernel must be enhanced with additional functionality that represents the network behavior for the scenario of interest. For instance, a wired network supporting Voice over IP (VoIP) might have DiffServ and MPLS LDP protocols in the network stack (among others). Kernel extensions, in the form of code written in a general purpose programming language such as C, tells the simulator how to process events.

There are two schools of thought on how the extension code should be built onto a DES kernel. The first school considers it the software user’s role to determine how the additional functionality is structured. The second school of thought pushes some of the responsibility back on the software, such that users build extensions around an existing framework. For instance, if a user writes the code for a network protocol model, he or she builds it around existing APIs. The reason why APIs and other kinds of frameworks are important in model development is that the organization of the code plays a critical role in determining the ultimate speed and efficiency of the DES. Therefore, simulators of the second school provide users with extensions to the general kernel that enforce fast simulation speeds.

Old network simulators subscribe to the first school of thought. This philosophy can result in prohibitively slow model execution time if the model designer isn’t an expert in software development and networking. QualNet is built based on the second school of thought. QualNet provides a well-defined set of interfaces to aid the user in developing efficient network models.
QualNet’s APIs exploit the universal network stack architecture to help users quickly implement any functionality between the layers of the stack. The APIs have been designed to work with a wide set of networks including IP, cellular, ATM, optical, satellite and wireless networks. The APIs are also extensible to accommodate new communication technologies. Providing users with an extensible set of APIs means QualNet simulation is extremely efficient yet extensible.

Modelers accustomed to an unstructured model development environment may fear the loss of flexibility using QualNet. Not so. QualNet still allows users the flexibility to design any realistic network model around the intuitive stack framework and APIs. QualNet’s architecture—a small kernel with stack-based extensions on top—helps ensure that user code can model any scenario, and do it efficiently.

QualNet’s Stack Architecture

QualNet’s architecture—a small kernel with stack-based extensions on top—helps ensure that user code can model any scenario, and do it efficiently.

QualNet is a network simulation engine that achieves fast execution speeds through smart architecture. Smart architecture starts with a clear modeling structure. QualNet uses standard network layers to segment the network into protocol models that are intuitive and clear. The TCP/IP and ISO protocols are stack-based. The ISO stack has seven layers, while the TCP/IP stack has only five layers (session and presentation layers are missing). For modeling purposes, the TCP/IP and ISO stacks are interchangeable, and they provide the structure needed to build a small but fully specialized kernel. Alternatively, non-network models (such as sensors and user behavior models) can be modeled with hooks provided with QualNet.

A solid structure provides a logical framework on which to build, but exceptions to the rules must also be accommodated. Modelers who want to abstract a layer (or more) to reduce detail in the network model can do so by removing the functionality in the layer between the two APIs.

No Unnecessary Calculations

Besides basing your simulator on a solid structure that’s easily extensible, there are additional ways to streamline core processes to achieve speed. An efficient simulation only performs calculations that are critical to the end result. For example, in QualNet and GloMoSim (the precursor to the commercial simulator QualNet), wireless nodes are grouped together in processor partitions so as to eliminate unnecessary calculations. In wireless networks, the location of each node with respect to all other nodes in the model is important in determining which nodes receive signals. For each receiver node, QualNet checks first whether it is in range before calculating path loss and other physical layer calculations that are processor-intensive. If the node isn’t in range, it saves processing cycles by avoiding the calculation [3].
Eliminating unnecessary calculations in the physical layer is where QualNet gains a lot of speed. Another example of performance gains achieved through extensive knowledge of networking is with signal interference in wireless networks. The computational load for calculating signal interference is very large. QualNet models signal to interference and noise ratio (SINR) by considering thermal noise, modulation, channel coding and antenna radiation patterns for each node. In fact, the number of events in the global event list increases by orders of magnitude when a high fidelity SINR model is implemented [3]. By using efficient data structure design, caching, and intelligent partitioning*, the developers of QualNet software have taken every opportunity to eliminate unnecessary computations to achieve higher simulation speeds.

The Relationship between Speed and Scalability

We’ve shown how speed in DES is achieved – smart architecture, a small kernel, and no unnecessary calculations. But speed and scalability go hand-in-hand. When a model is producing fast or real-time results, the number of events the simulator processes in a given period of time is large. Similarly, when a model is scaled up, that is, much more traffic and devices are added to the network, the number of events the simulator processes increases. Thus, real-time simulations and large network simulations require the same thing: a simulation engine that processes events fast.

An analogy for the interrelation between speed (via a fast kernel) and scalability (through good

---

* For more information about partitioning on sequential and parallel processors in QualNet, please refer to the white paper published by Scalable Network Technologies entitled “Parallel Execution.”
memory management) is a group of firefighters passing buckets to put out a fire. Firefighters throwing buckets of water on a fire have two limiting factors: 1) the speed at which the buckets can be passed from person to person, and 2) the flow rate of water at the tap. The time it takes for a bucket to be passed from person to person is analogous to the time it takes the kernel to process an event. The time it takes to fill a bucket is analogous to the time it takes to allocate and deallocate memory for an event. If one factor slows, the whole system slows. Speed and scalability go hand-in-hand.

Scalability Techniques

While increases in speed and scalability are directly linked, the way scalability is achieved is different from the way increases in speed are achieved. Smart memory management can be achieved by two methods 1) the economical use of memory, and 2) the efficient manipulation of that memory.

A large model has a large number of nodes. A large model logically takes up more space in memory than a small model. It is reasonable to assume that 1000 nodes take up 1000 times more space in memory than one node. However, the amount of memory required to perform operations to process events, is a polynomial function of the number of nodes. For instance, in multicasting and wireless signal transmission, if a node sends a message to 1000 nodes as opposed to 1, the processor might use $1000^2$, or 1 million times more memory. Thus, memory management is where software developers concentrate their efforts to support scalability in DES.

Pooled Memory

Making operations as efficient and economical in terms of memory usage is the way to scalability. One memory management tool is shared memory. The operating system expects each process to allocate an area in memory when it needs to store something. Conversely, when the data is no longer needed, the process deallocates that memory, freeing it up for another process. With pooled or recycled memory, an area of allocated memory is kept handy for re-use, rather than fully allocating new memory and deallocating it once it’s no longer needed. Pooling memory speeds up
the process of accessing and updating memory for both large and small models because the processor is expending fewer cycles to watch over a smaller amount of data in memory.

Global Variables Used Sparingly

Another concept that is important to memory management is global and local state variables. Global state variables are elements in a code base that are shared by every function. Local variables are used only within a function.

The use of global variables is a contentious issue in software development. In a sequential program, careful use of global variables can reduce memory requirements and improve performance. But using global variables violates widely accepted coding conventions, such as data encapsulation, and makes the code harder to debug and maintain. In a sequential simulation, special care must be taken to ensure that global variables are shared correctly. For example, if the variable represents a shared resource such as an Ethernet switch, additional variables are required to track overlapping signals passing through the switch. Global variables also present serious difficulties in the parallelization of DES*. Considered undesirable by software engineers in general, global variables slow down simulations and should be avoided whenever possible.

Memory Speed Relative to Processor Speed

The sharing of global variables between model partitions is one challenge in memory management. Another challenge relates to the relative speeds of processors and memory. Processors and memory technologies are constantly improving, but they are advancing at different speeds. According to Moore’s Law, processor speed doubles every 18 months, whereas memory speed increases by 25% every 18 months [4]. That means memory technologies are poised to become the limiting factor in hardware performance. But even setting aside the issue of memory speed slowing hardware performance, memory speed is a problem for software application developers because the ratio of memory speed to processor speed is always changing, making memory optimization difficult.

Examples of poor memory management adversely affecting performance are described in a research paper [4]. Figure 7 shows that the faster the processor, the greater the relative slowdown due to bad memory management. These findings were discovered when researchers modeled a state saving process as an inefficient memory practice. Thus, how a software application is designed determines the extent memory performance can slow down simulation speed. Applications must be extremely thoughtfully designed to prevent memory limitations from counteracting speedup due to parallel execution and faster processors.

Good memory management is important for general performance and extremely critical for scalability in DES. Next, we show you proof of superior memory management in QualNet.

Good Memory Management in QualNet

QualNet uses techniques such as pooled memory and minimal global state variables that are mindful of the increasing disparity between memory and processor performance. A way to measure good use of memory is by observing how close to linear the relationship is between the number of nodes and the execution time. QualNet performs well for 100, 1000, and 10,000 node models. In other DES tools, discontinuities in the relationship between the size of a model and the execution time indicate memory management problems.

* For more information about global variables and their influence on lookahead, please refer to the white paper published by Scalable Network Technologies entitled “Parallel Execution.”
Network simulation is inherently resource-intensive. Speed and scalability are two important requirements for modelers evaluating their options for DES software. We established the importance of high fidelity models and how new network simulators are capable of supporting high fidelity models due to improvements in design.

To maximize speed, network simulators must have a small yet specialized kernel. As much core functionality as possible should be included in the kernel to avoid additional overhead due to abstraction layers running on top of the main kernel. Next, a clear and well-defined model extension structure, such as that used by QualNet based on the ISO network stack, brings outstanding performance gains without sacrificing model extensibility. Modelers build their custom code around well-designed APIs that provide the flexibility they need. Finally, speed is also achieved through critical path processing, or the elimination of unnecessary calculations by partitioning and capitalizing on knowledge of networking technologies.

Speed, or the ability to process events fast, is inherently linked to scalability, the ability to accommodate a large model of heavy traffic and hundreds or thousands of nodes. Specifically, speed relates to the efficiency of the kernel, and scalability is linked to memory management.

Next, we discussed examples of smart memory management to increase scalability, such as pooled memory and sparing use of global state variables. The importance of good memory management was further underscored by evidence in an academic study. The faster the processor, the greater the slowdown measured due to poor memory management.

Finally, we presented proof of smooth scalability in QualNet. Whether modeling 2500 or 10,000 nodes, the graph of simulation time versus number of nodes had no discontinuities and was almost linear. Most importantly, as proof of both speed and scalability, QualNet showed consistent efficiency when modeling networks of increasingly large size.
References


About Scalable Network Technologies

Headquartered in Los Angeles, California, Scalable Network Technologies is the leader in parallel processing technology for network performance evaluation. The company develops and supports high-fidelity evaluation software tools used for predicting the performance of computing and communications networks and network devices.

SNT has created a new category of evaluation tools for today’s sophisticated networks that meets the demand for real-time, real-network performance testing. Widely recognized for its flagship product, QualNet, the company’s customers include major aerospace and defense contractors, the US Department of Defense, mobile network operators, as well as research agencies and universities.

For more information or a free 2-week trial of QualNet Developer software, please visit:


6100 Center Drive
Suite 1250
Los Angeles, CA 90045

http://www.scalable-networks.com
http://www.qualnet.com

For additional information, please send email inquires to: sales@scalable-networks.com.