Real world networks have long distance links, satellites, congested routers, firewall inspection, packet interception, hardware failures, and other outages. But the networks in most software development labs are too fast and too clean – not at all like the real Internet. You must test and verify application performance prior to deployment. A poorly performing app can erode user confidence and damage your reputation.

Assure success by emulating real-world network conditions in the lab, and subject the application to the full range of network conditions likely to occur.

**Emulate the Real World in Your Lab**

The KMAX network emulator helps network engineers test and measure performance in order to identify and remove defects. Network emulators turn well behaved development and test networks into the kind of slow, congested, and less-than-reliable services encountered on the internet.

In addition, unlike the real internet, network emulators allow the operator to control these conditions so that products and apps can be subjected to controlled and repeatable tests, by routing selected packets through a series of impairment nodes.

**The Result:**

- Deliver robust, high quality products and applications
- Accelerate time to market
- Eliminate guesswork and surprises
- Completely characterize your app’s network performance

**State of the Art, Fully Intuitive User Interface**

Use the KMAX Network Emulator from any web browser on a tablet or desktop. Use the Wizard to guide you through set up and operation. Point and click to select the scenario closest to your production environment. KMAX scenarios contain sets of network impairments -- drops, delay, jitter, etc. -- that may be fine tuned and adjusted to match your environment.

![Create a Real-World Network in Your Test Lab](image)

---

In addition, unlike the real internet, network emulators allow the operator to control these conditions so that products and apps can be subjected to controlled and repeatable tests, by routing selected packets through a series of impairment nodes.

**The Result:**

- Deliver robust, high quality products and applications
- Accelerate time to market
- Eliminate guesswork and surprises
- Completely characterize your app’s network performance

**State of the Art, Fully Intuitive User Interface**

Use the KMAX Network Emulator from any web browser on a tablet or desktop. Use the Wizard to guide you through set up and operation. Point and click to select the scenario closest to your production environment. KMAX scenarios contain sets of network impairments -- drops, delay, jitter, etc. -- that may be fine tuned and adjusted to match your environment.

![Create a Real-World Network in Your Test Lab](image)
## Scenarios

Select a pre-defined, network emulation scenario from our library. Some examples include:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Atlantic T1 ATM link</td>
<td>Cellular radio handoff</td>
</tr>
<tr>
<td>Low earth orbit satellite</td>
<td>VoIP: To remote office via WAN</td>
</tr>
<tr>
<td>Geosynchronous satellite</td>
<td>VoIP: multi-way conference over WiFi</td>
</tr>
<tr>
<td>Streaming media over satellite</td>
<td>Connection impairments</td>
</tr>
<tr>
<td>Periodic network downtime</td>
<td>Failover to backup servers</td>
</tr>
</tbody>
</table>

Pre-built library scenarios feature intuitive controls that appear on top of an end-to-end diagram of the whole network, to provide a bird's eye view. You can also add your own custom background diagram. Here are three examples from the library:

1. **Multi-hop Low Earth Orbit Satellite**

2. **Cellular Radio Handoff**

3. **VoIP to Remote Office via WAN**
Create Your Own Scenario

Emulate the real world in your test network by setting basic and advanced parameters for these impairments:

- **Packet Drop/Loss**: Add “burst mode” for a realistic emulation.
- **Packet Delay/Latency**: From zero to 10 seconds, with microsecond precision.
- **Packet Jitter**, with or without reordering. Specify a custom jitter distribution: Uniform, Gaussian, Pareto, or Pareto-Normal.
- **Packet Duplication**, with rapid back-to-back transmission.
- **Packet Corruption**: Specify packet-centric or bit-centric probabilities of corruption.
- **Rate limiting**: Choose from several queue management algorithms.

Model Worst-Case Situations:

- Turn on burst mode for any impairment, to simulate back-to-back real-world stochastic bursts.
- Add time-varying expressions to any parameter

Regardless of the production environment – anything from the routine to the extreme to legacy – KMAX permits customization to precisely emulate that world.

Incorporate Real or Simulated Traffic

KMAX accommodates all forms of real or simulated network traffic:

- Use real devices to generate packets, then impair only selected packets and protocols under consideration, while filtering out the rest.
- Or, use a packet generator to create simulated network traffic for the device under test, then apply impairments only to those packets, leaving the rest of the network unaffected.

Classify Network Traffic

Create filters for the packet classifier using the Classifier Filter Library, or create your own filter, then apply different impairments to different filter outputs.

Examples of filter criteria include:

- IEEE 802 header fields: MAC source/destination addresses, Ethernet type/length, MPLS criteria, VLAN tags.
- IPv4 or IPv6 header fields, including chained headers, fragment status, and QoS.
- Higher level protocols: UDP or TCP port numbers.
- Data within the payload, for any layer.
Add Time-varying Impairments to your Emulation

Real world networks change their behavior over time: excessive jitter, dropouts, or reductions in bandwidth can vary smoothly, or jump from one value to another, over the course of minutes, or days. Create a network emulation that varies the impairment parameters over any time period:

Specify a time-varying expression for any metric, using equations, piecewise curves, or shaped pulses:

Examples

- Test your streaming clients: The device under test should handle time-varying impairments, and perform flawlessly, for days.
- Run your protocols continuously, while the time-varying impairments cover all corner cases – perfect for overnight stress testing.

Remote Control via REST API

The KMAX web server exposes a REST API that provides remote control over all aspects of KMAX, including the impairment engine and statistics gathering. The KMAX REST API Programmer’s Guide describes the commands that you can use to insert KMAX directly into your testing pipeline:

- Plug KMAX into your existing diagnostics tools, for custom statistics collection and display.
- Automate KMAX reconfiguration, to run different configurations for different tests - essential for overnight testing.
- Add a fully custom user interface, by using any scripting language with an HTTP client.

The KMAX REST API adheres closely to all “RESTful” principles, which means you get scalability, discoverability, and manageability for free.

KMAX Documentation

- KMAX System Guide
- KMAX QuickStart Guide
- KMAX Quick Reference Guides
- Stencils for creating network diagrams
- KMAX REST API Developer’s Guide

Pictured: KMAX-MM
Other options, including a rackmount version, are available.
Charts, Graphs and Tables

View network emulation statistics in tables and charts (graphs). Add any number of tables or graphs to one or more browser windows, then monitor the values while you change impairment settings. You have the flexibility to see everything going on in the KMAX server:

▶ Display packet rates and packet counts at all points in the KMAX impairment process, such as interface ports and flow inputs, for precise analysis.
▶ Display individual impairment statistics: packets dropped / duplicated / jittered / delayed, etc: A total of 17 metrics.
▶ Select any unit of representation: bytes/sec, bits/sec, packets/sec, total bytes, total bits, and total packets.

Graph highlights:
▶ Each graph can chart two parameters, for side-by-side comparison.
▶ Switch between linear or semi-log axes: essential for analyzing metrics that vary over many orders of magnitude.

Table highlights:
▶ The table can display dozens of parameters simultaneously
▶ The table allows quick identification of many correlated metrics, for enhanced troubleshooting and understanding.

See Pages 6-7 for KMAX Specifications
KMAX Network Emulator Specifications

Hardware Enclosure:
- Tablet sized: 6.6” x 6.2” x 1.2”
- Lunchbox sized: 12.9” x 8.7” x 3.8”
- Rackmount option: 15.7” x 11.8” x 1.7”
- Virtual Machine (VM) HyperVisor: software only

Interfaces:
- 10/100 Mbits/sec
- 10/100/1000 Mbits/sec
- 1 Gbit/sec
- 10 Gbits/sec for two ports

Operating System: FreeBSD

Impairment Engine: Kernel-driven impairment engine

Transparency / Configuration: Layer 2 transparent bridge

User Interface:
- Web-based graphical user interface
- Command Line Interface
- Script control:
  - TCP API with Python or Java, using JSON
  - RESTful interface, using JSON
  - Load/Save configuration files in easy-to-use YAML format

Standard Impairments:
Packet Delay/Latency:
- Specify packet delay / latency
- Vary packet delay / latency over time *
- Add jitter to the delay, with or without packet reordering
- Resequence groups of packets to simulate queue bursts

Packet Jitter
- Jitter Distribution:
  - Normal / Gaussian / Bell curve
  - Pareto
  - Pareto-Normal
  - Uniform (equal)

Packet Duplication:
- Specify duplication probability
- Vary duplication probability over time *
- Enable bursts:
  - Specify burst probability
  - Specify burst window duration
  - Enable burst skew
- Vary burst probability over time *

Packet Drop / Loss:
- Specify drop probability
- Vary drop probability over time *
- Enable bursts:
  - Specify burst probability
  - Specify burst window duration
  - Enable burst skew
- Vary burst probability over time *

Rate Limitation:
- Specify:
  - target rate
  - queue length
  - overhead bits
  - minimum packet size
  - maximum packet size
- Vary any of the above over time *
- Support bit clocking rate control
- Support token bucket rate control
- Mark for drop or random early detection
- Specify transition action on queued packets
- Flush queue

Packet Corruption:
- Specify corruption probability
- Limit corruption to Ethernet payload
- Specify burst probability
- Vary burst probability over time *

Emulated Links / Multiple Flows:
- Unidirectional and bidirectional packet flow
- At least 5 flows in each direction; 10 total
- Larger models support > 64 impairment nodes

Classify packets based on:
- IPv4 header fields
- IPv6 header fields
- MAC source or destination addresses
- Ethernet type or length field
- IEEE 802 header fields (includes VLAN tags)
- Data within the packet

Classify packets based on fields in protocols:
- Low-level protocols:
  - arp
  - dhcp
  - icmp
  - mpls
  - spanning-tree
- IP layer:
  - IPv4 packets of varying lengths
  - IPv4 QoS bits
  - IPv6 fragment
  - IPv6 flow range

* Time-varying waveforms can be specified via a simple pulse model, via an equation, or using a list of value/duration pairs
TCP / UDP layer:
- Any TCP packet
- TCP port numbers
- Any UDP packet
- UDP port numbers
- http
- DNS
- SIP

Applications layer:
- VoIP
- VoIP with QoS bits in various configs
- Apply different impairments on each emulated link

Manipulate contents of headers and payloads:
- Randomly corrupt a packet
- Intelligently corrupt a packet
- Create a packet and selectively insert into the flow
- Dynamically create and change distinct flows of packets
- Inspect packet headers to select packets for modification
- Rewrite packet headers
- Rewrite packet data

Alter/Transform:
- Modify a live packet from a real-world network conversation; no need to create a new packet from scratch.
- Supports “man in the middle” operation by sitting between two devices, intercepting the network traffic; it is not limited to creating a packet to send it to an end point.
- Permits multiple, concurrent packets to be altered; operation is not limited to one packet at a time.
- Changes to the Alter Transform code is made via a graphical interface that provides guidance and feedback to the engineer.
- Replicate a specific fuzz test to verify the robustness of a network device.
- Create a new set of tests for a particular feature using the simple Python programming language.
- During the network conversation:
  - Insert or append bytes into a protocol packet’s payload or header.
  - Remove bytes from a protocol packet’s payload or header.
  - Find and remove specific IPv4, IPv6, or TCP options.
  - Rewrite any header field or payload bytes, such as source or destination IP addresses.
  - Automatically update protocol length fields after byte insertions and deletions.
  - Perform operations on byte strings inside the packet, such as addition, bit-wise “and”, “or”, and “xor”.
  - Recompute all checksums upon the engineer’s request.
  - Perform alterations to a copy of the original packet and send the altered copy before or after the original packet - thus allowing insertion of entirely new packets into the network dialog.
  - Supports tunneling
  - Any Alter Transform functions defined by an engineer may be saved and reloaded at a future time.
  - The Alter Transform functions may also be combined with other standard impairments such as jitter, delay, re-ordering, etc.
  - Compared with other commercial and open source tools (such as Scapy), the Alter Transform function:

Maxwell Patent No 7310316
Testing device
US 7310316 B2

ABSTRACT
A test device (21) sits between two or more nodes (20, 22). The nodes (20, 22) communicate in conversations, according to some predetermined protocol. The test device (21), under user control, may introduce jitter, drop packets, create new packets, reroute packets, and reorder packets in the conversations. Particular conversations are detected and tracked by respective virtual state machines (38, 39, 40) within the test device.