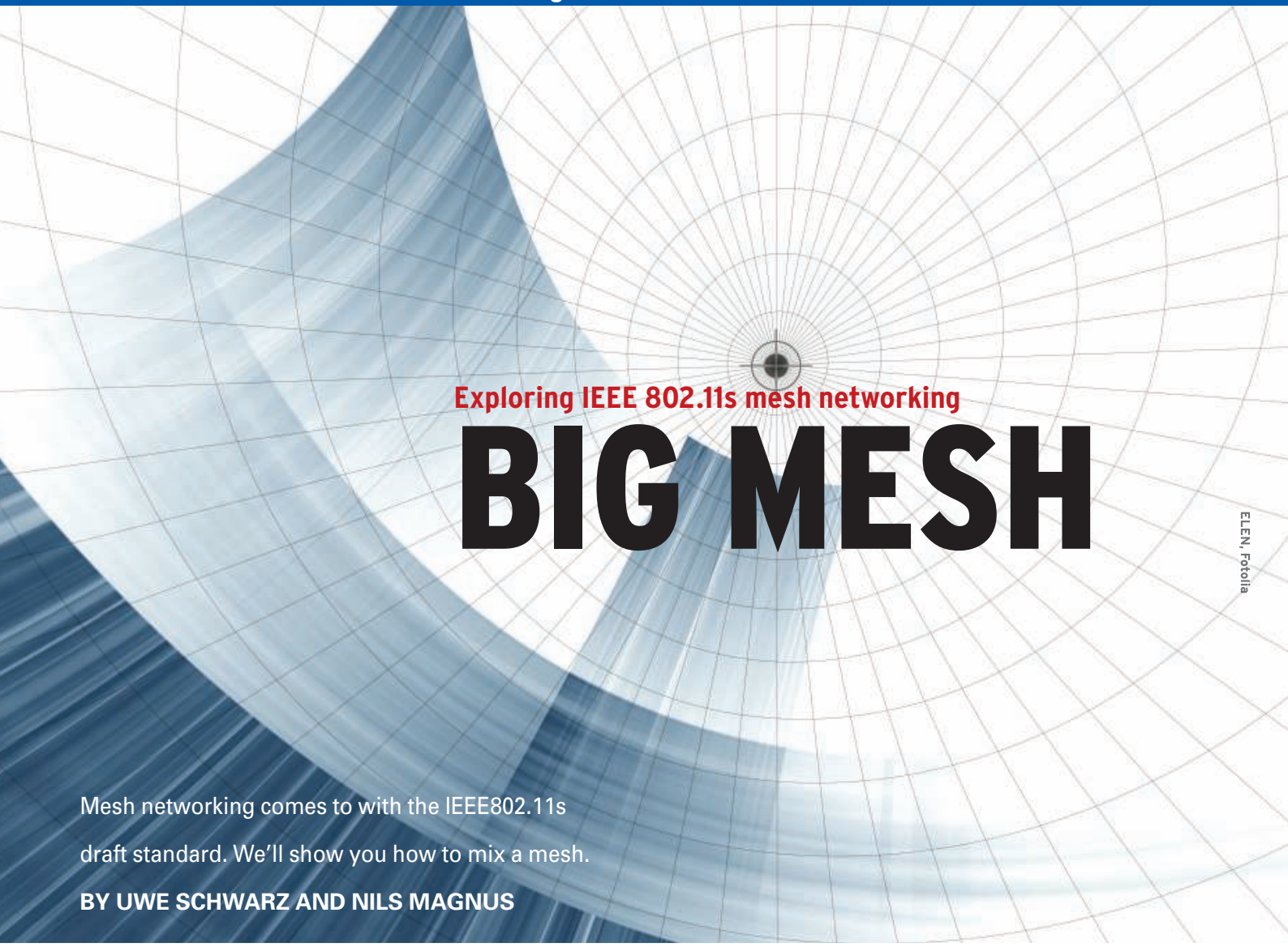

Exploring IEEE 802.11s mesh networking

BIG MESH

ELEN, Fotolia

Mesh networking comes to with the IEEE802.11s draft standard. We'll show you how to mix a mesh.

BY UWE SCHWARZ AND NILS MAGNUS

The average wireless network depends on the presence of an access point device, which serves as a connection point for wireless clients and links the WLAN to the Internet. This technology is so widespread in urban areas that a wireless user is never far from a point of connection – in airports, coffee houses, offices, and even millions of homes.

In spite of its omnipresence, the access point is, in fact, not really an essential element of the wireless network. An access point–style network is said to be operating in *infrastructure mode*. An alternative approach, known as *ad hoc* mode, lets computers establish direct connections with each other. Ad hoc mode is sometimes used for one-to-one connections between (say) a laptop and a desktop system when no access point is available. Computers that connect through ad hoc mode do not have the benefit of DHCP and other services that typically run on the access point, so they must negotiate any network settings.

In remote areas, and in developing countries without a pervasive networking infrastructure, decentralized alternatives such as ad hoc mode become increasingly attractive. However, users who attempt to operate in a conventional ad hoc setting run into restrictions fairly quickly. Although you can communicate with immediate neighbors, conventional ad hoc networking does not let you use neighbors as intermediaries to access other systems, so the range of the network is limited to the range of a single wireless device. Some operating systems, in fact, don't even support more than two peers in an ad hoc network.

A further extension of the ad hoc concept is the mesh network, which is embodied in the IEEE 802.11s draft standard. Mesh networking expands the size and range of the ad hoc network by letting network peers forward messages and transmit network information to other peers. The peers can thus build a collective picture of the network topology, and a message can pass through a chain of peers to a node beyond the immediate signal range.

IEEE 802.11s received renewed attention through the work of the One Laptop Per Child (OLPC) project, which was formed with the goal of providing network-enabled laptop computers for millions of school children in developing countries. The OLPC developers needed a way to interconnect these little laptops in a setting in which a whole village might have only a single access point or Internet gateway. Mesh networking provides the solution [1]; even if a device can only find one peer, it can still access other devices (and, ideally, an Internet access point) by transmitting through a chain of connections across the mesh. Mesh networking also has uses in the developed world, although it is unlikely to ever replace the convenience and simplicity of access point networks. In rural areas, or in situations in which a network must assemble and self-configure temporarily, mesh networking might someday play an important role.

IEEE 802.11s is still a draft [2], and it is unlikely to be officially approved before 2010.

In August 2008, the Linux kernel added a patch for meshing support in the ath5k module, with native support available for the *b43*, *libertas_tf*, and *zd1211rw* wireless drivers. In Linux kernel version 2.6.26, *zd1211rw* already has rudimentary IEEE 802.11s support based on the *mac80211* subsystem [3]. The code in kernel 2.6.26 does not let users set up genuine mesh networks, but the current developer version of the wireless-testing.git repository adds additional mesh networking support [4].

Unless you happen to own an OLPC laptop or some equivalently pre-configured device, the game of mesh networking in Linux is not exactly for beginners, but if you feel like experimenting with your own mesh network, the necessary tools are certainly available – but make sure your wireless hardware supports IEEE 802.11s.

How It Works

In an IEEE 802.11s network, each node is in contact with its direct neighbors. The nodes can be computers, laptops, or devices such as IP-capable smartphones. The mesh protocol tries to auto-discover the best possible route for a packet. (See the "OLSR and Batman" box for more abou routing on mesh networks.)

No fixed connections exist between individual nodes. A node transmits packets to a specific target node that it considers most suitable for taking the packets closer to their final destination. The next hop uses the same approach until the package reaches its target.

In meshing, data is synchronized as often as possible – typically every few seconds. This approach keeps the network dynamic and allows it to react to changing conditions caused by moving nodes, obstacles, or direct connections. The mesh evaluates each active link path on the basis of specific criteria and chooses the best path. Thus, routing is one of the most critical, and the most expensive, tasks in meshing.

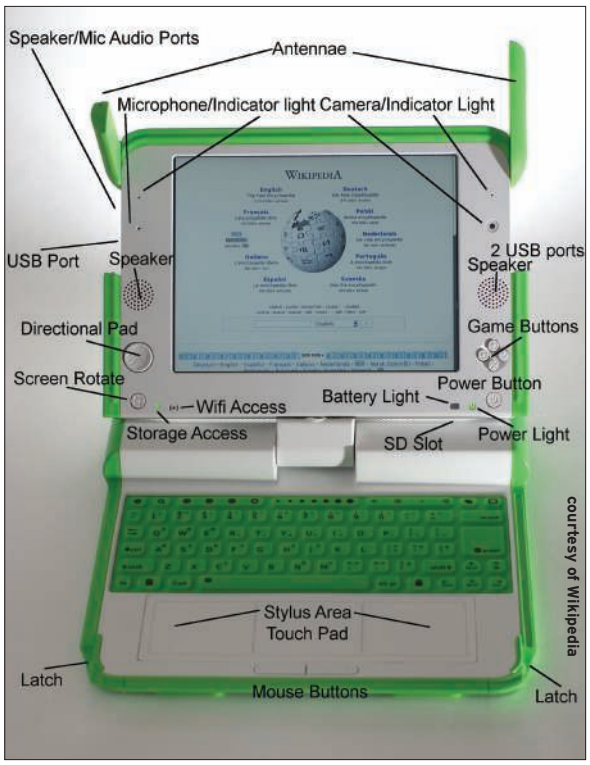

Figure 2: The OLPC XO system comes with two built-in external antennas.

Mesh Networking Up Close

Mesh networks with Internet access typically have a main node that provides broadband Internet access via DSL or UMTS. This main node often offers other network services, such as DHCP. The network admin sets up gateway functionality on this node only by, say, running the *echo 1 > /proc/sys/net/ipv4/ip_forward* command and possibly adding NAT rules for masquerading.

The individual mesh nodes should have at least two WLAN modules. Three modules will improve availability and increase the maximum number of possible clients. Typically, one module will run in access point mode to give other clients access, or the other modules will use ad hoc mode for meshing. In a perfect world, all of these modules have omnidirectional antennas that cover a large reception area (see Figure 1).

As you might guess, the effectiveness of mesh networks is largely dependent on the range at which the nodes can receive the signal. For systems that are built specifically for mesh networking, the antenna configuration receives significant attention. (Notice the two

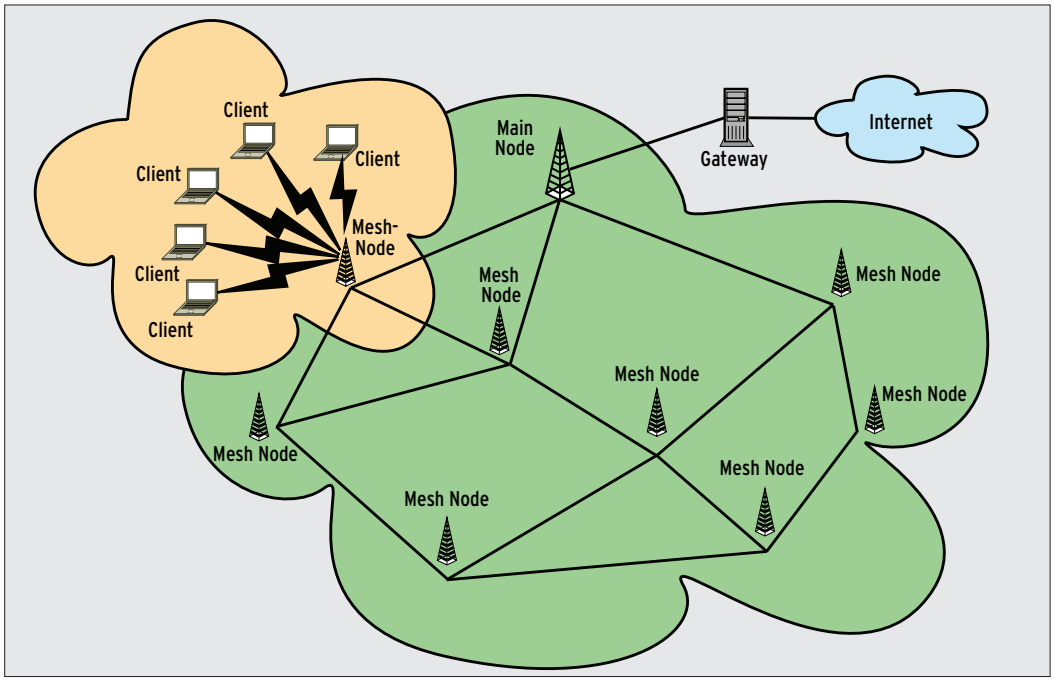

Figure 1: A typical village network comprises two kinds of participants: clients that use legacy infrastructure mode to attach to nodes linked by an IEEE 802.11s network, and one node that has internet access.

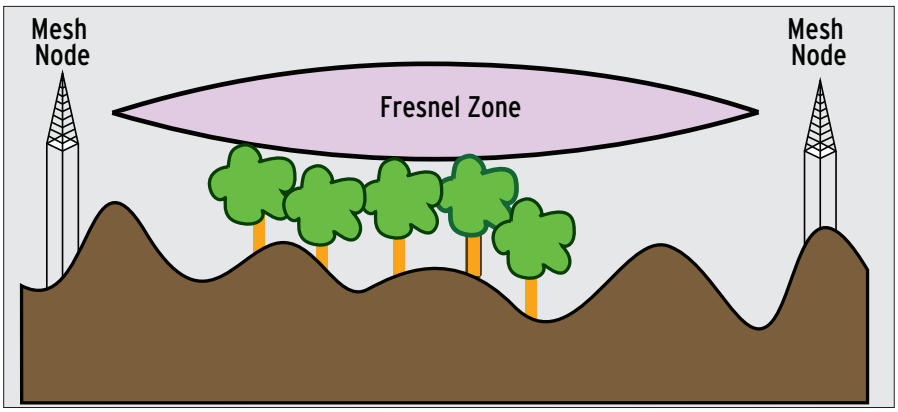

Figure 3: Mesh nodes should be located as high up as possible in Fresnel zone and without obstacles obscuring the line of sight.

external antennas in the OLPC XO laptop shown in Figure 2.) A network with multiple modules per node can use a more flexible antenna configuration; for example, one module can use a sector antenna to reach peers within a wide range, and another can work for nodes in close proximity. The following sections describe how to set up mesh networking in Linux.

Setting Up the Mesh

The wireless-testing.git repository now has an updated, mesh-ready version of the kernel with matching drivers. The command

```
git clone git://git.kernel.org/➥
pub/scm/linux/kernel/git/➥
linville/wireless-testing.git
```

downloads the current version.

Configuration

To configure meshing, become root, then select the *mac80211* driver platform (*CONFIG_MAC80211*) and the mesh property (*CONFIG_MAC80211_MESH*). For the ZD1211 WLAN card, you will need matching drivers, such as *CONFIG_ZD1211RW*. Now build and boot the new kernel.

In user space, again working as root, you will need the *iw* tool to configure the network card [5]. The tool is intended as a standardized, long-term replacement for *iwconfig* and its sibling commands, much as *ip* has become an alternative to *ifconfig*. Many distributions do not have a package for *iw*; you might need to download the *iw* source code via git:

```
git clone ➥
http://git.sipsolutions.net/➥
iw.git
```

The package depends on *libnl*; you will need version 1.0-pre8 or newer [6]. Users with Debian or Ubuntu need the libnl-dev package.

To configure the mesh with the new tool, again as root, select a mesh ID. The mesh ID must be identical on each node. The ID can comprise a maximum of 32 characters; the example calls this *MyMesh*. Then go on to set up an interface – *wmaster0* in this example:

```
iw dev wmaster0 ➥
interface add mesh0 type mp ➥
mesh_id MyMesh
```

This step creates a new interface, which you still need to set up for operations with the gateway *10.0.0.1*. To set the IP address and netmask, you can use the standard *ifconfig* command, then use *iwconfig* to select, for example, channel 7 for the mesh network:

```
iwconfig mesh0 channel 7
ifconfig mesh0 10.0.0.1 ➥
netmask 255.255.255.0 up
```

After following these steps for all the devices on the mesh network, the mesh will start to converge. To check its progress, issue the following command:

```
iw dev mesh station dump
```

Nodes the command reaches are added to the list. This lets you check to see whether the protocol has set up connection between the nodes.

To finalize the configuration, you need to set up Internet access by configuring DHCP and NAT on the main node that will act as a gateway for the mesh network.

On Debian, you would configure the dhcp3-server package in */etc/dhcp/dhcpd.conf*, as described in Listing 1. This step configures a private subnet of 10.0.0.0/24 for the mesh network *my-mesh.org*, sets the gateway to *10.0.0.1*, and tells the clients to request the name server. The name server is available in the Debian bind9 package. In the simplest case, you would add the IP address of a known DNS server on the Internet to the *forwarders* section of */etc/bind/named.conf.options*. As a final step, enable the setting

```
net.ipv4.ip_forward=1
```

in */etc/sysctl.conf* to route packages out of the mesh network and onto the Internet. At the same time, add

```
iptables -A POSTROUTING ➥
-t nat -s 10.0.0.0/24 -j ➥
MASQUERADE
```

A Question of Location

In a truly mobile networking scenario, such as the world of OLPC laptops, the positioning of the nodes within the mesh zone is essentially arbitrary, and it is not possible to define a fixed location for the peers operating in the mesh. In other mesh scenarios, however, the admin might have some control over the location and overall topology. For example, a mesh network might consist of desktop systems assembled temporarily at a conference or construction site. In such cases, it is a good idea to choose the location of the nodes carefully.

To optimize throughput, the WLAN devices should have a line of sight connection if possible. If a node can reach one or multiple other nodes reliably and permanently, a directional beam antenna can improve the mesh's performance.

The space in which the radio waves propagate between a transmitter and receiver is referred to as the Fresnel zone. It is an ellipsoid in which the transmitter and receiver occupy the focal points (see Figure 3). Because the ellipsoid is higher in the middle, mesh participants should locate their nodes as high up as possible.

to enable masquerading for the private addresses on the wireless network on the router's public IP address. This step completes the setup for the main node.

Working as root, you still need to run *iw* on each node to configure the mesh network. The IP addresses and DNS details are provided courtesy of the DHCP server. If you intend to set up access for normal clients without mesh support, you need to configure additional wireless modules with Host AP mode support on these nodes [7].

Linux IEEE 802.11s support is still at an early stage. It will take some time for more mesh-ready NICs and drivers to appear. Users can look forward to IEEE 802.11s becoming far simpler in the next version of the kernel, with more and more drivers supporting the *mac80211* subsystem.

Conclusions

Until the standard is widely adopted by vendors, don't be surprised if the details of the configuration tools change. In the meantime, assuming you have the right cards and tools, nothing is stopping you from setting up your very own mesh network right now. ■

OLSR and Batman

Developers have used different approaches to implement meshing. Two popular options are Optimized Link State Routing (OLSR) [8] and the more sophisticated Better Approach to Mobile Ad hoc Networking (Batman) [9]. Both options use OSI Layer 3 routing to discover paths across the mesh. The more recent Batman Advanced [10], a fork of Batman under independent development, uses OSI Layer 2 switching.

In OLSR, each node sends *Hello* messages at fixed intervals. Each node that receives the messages evaluates them. After creating a map of its environment, a node reports its findings to its neighbors in the form of a TC (topology change) message. When a TC message reaches a node, the node recalculates the network topology. This means that each node will know the best way to route a packet at any time. Topology changes are recalculated with the Dijkstra algorithm [11]. Individual nodes propagate their knowledge of the network structure to neighbors, thus incrementally improving their knowledge of individual path characteristics. The algorithm is regarded as stable, but slow to converge in some cases.

OLSR does not recalculate the topology for each packet, and in this way saves computational cycles. It uses the existing network while recalculating the topology. OLSR evaluates the number of hops to the target over a certain path to calculate the best route. This approach guarantees a cheap route to each reachable target on the network after an initialization phase. Because the structure of a wireless network is susceptible to change, it is important to exchange Hello messages at relatively short intervals. The typical update interval is 5 seconds. In addition, message packets from other nodes arrive, causing the node to recalculate the topology. This can be an issue, especially for small devices with less powerful CPUs: These devices could use up much of their computational power just recalculating the network topology.

Plugins and alternative algorithms are available for OLSR to mitigate the effect of poor hardware scalability. Batman took this issue into consideration at the development stage. Both protocols let users specify whether a node has Internet access and is thus capable of acting as a gateway for other nodes. In addition, Batman lets you configure the bandwidth the gateway uses to connect to the Internet. This feature allows clients to discover the most favorable gateway and removes the reliance on a centralized entity to manage this information.

The Batman alternative attempts to solve the problem of OLSR with the use of too much computation time to calculate the topology. Batman uses a different approach to discovering the best route among the available alternatives. Just like OLSR, Batman first floods the network with originator messages.

Each node generates a message and lets its neighbors distribute it. Each neighbor thus duplicates each message it receives and broadcasts the message. Nodes do not store the complete network topology, in contrast to OLSR, but just the part they need to reach their immediate neighbors. Each node thus only knows which node to use to transmit a packet, but not what the rest of journey will look like for the packet. This approach reduces the overhead for recalculating the best route and allows the protocol to scale more easily. Although the physical limit for OLSR is around 100 nodes – especially if the nodes happen to be embedded systems – Batman will scale to support networks with as many as 500 nodes.

Because the more recent Batman Advanced protocol operates in the link layer, to the user, the mesh network looks like a segment managed by an Ethernet switch. From the user's point of view, each participant has a direct connection; the protocol hides the underlying details. Batman Advanced is most compatible with the 802.11s standard because it relies entirely on Layer 2 (switching). Unfortunately, this design removes the ability to define a node as a gateway. This administrative task is left to the administrator or a higher-layer protocol.

Listing 1: Simple DHCP Configuration

```
01 subnet 10.0.0.0 netmask
   255.255.255.0 {
02   range
   10.0.0.100 10.0.0.199;
03   option routers
   10.0.0.1;
04   option domain-name
   "my-mesh.org";
05   option domain-name-servers
   10.0.0.1;
06   default-lease-time 600;
07 }
```

INFO

[1] Meshing at One Laptop Per Child: *http://wiki.laptop.org/go/Mesh_Network_Details*

[2] IEEE 802.11s Task Group status report: *http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm*

[3] Open80211s Project: *http://www.open80211s.org/trac/*

[4] *Wireless-testing.git* repository: *http://linuxwireless.org/en/users/Download#Checkingoutcompat-wireless-2.6.gittree*

[5] *iw* HOWTO: *http://linuxwireless.org/en/users/Documentation/iw/*

[6] Netlink library: *http://people.suug.ch/~tgr/libnl/*

[7] Host AP drivers: *http://hostap.epitest.fi*

[8] Optimized Link State Routing (OLSR): *http://www.olsr.org*

[9] Open-mesh.net Batman pages: *http://www.open-mesh.net/batman*

[10] Batman Advanced: *http://open-mesh.net/newsfolder/0-2-final-0-3-alpha-batman-advanced-battools-0-1-alpha*

[11] Dijkstra algorithm: *http://www.wikipedia.org/wiki/Dijkstra-Algorithmus*