The moɔpTi*k*Z package

a library for typesetting computer networks

Stephan M. Günther

v1.1, 2026/11/15

The moɔpTi*k*Z package provides shapes for typesetting computer networks, e. g. shapes for hubs, switches, routers, and more. It has been in use since 2012 for the lecture materials of an introductory course in computer networks due to shortcomings of some commercial products and (at that time) the lack of meaningful alternatives. Currently the package provides the following shapes, styles and commands:

- `hub`, `switch`, `mlswitch`, `router`, `fwrouter`, `fw`, `nuc` (alias `ap`), `modem`, `wlanmodem`
- `client`, `notebook`, `smartphone`, `server`, `rack`, `db`, `internet`
- `user`, `usergroup`, `messageclosed`, `messageopen`, `key`, `keyhole`
- `waves` (wireless links, Section 4.5)
- `\tikzextractx`, `\tikzextracty`
- `\moeptikzset`

1 What it can do

Did you ever try to build slides for a networking lecture from scratch? If so, do you care about off-topic details such as using solely vector graphics, perfect alignment of nodes, easy modification of hundreds of figures when you decide to change fonts or colors, and doing all that without even thinking about a pointing device? Then you probably like what moɔpTi*k*Z can do for you:

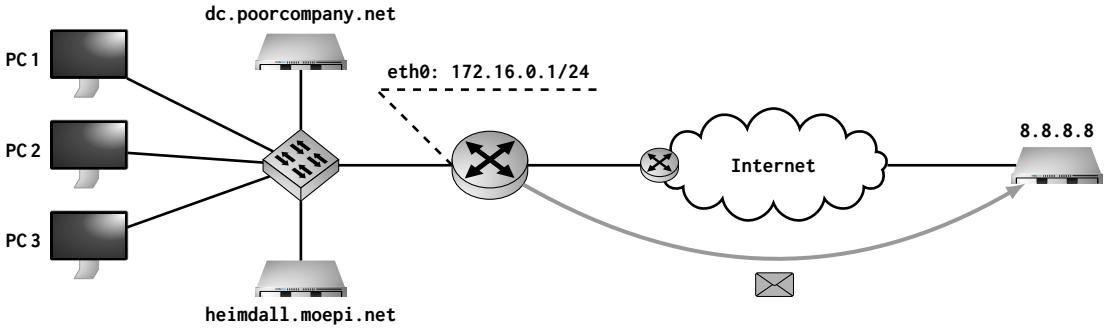


Figure 1: What moɔpTi*k*Z can (or cannot?) do for you

2 What it cannot do

You probably noticed the misalignment of `PC2` (and its friend) in Figure 1. Well, you might not have noticed it in the first place (I didn't either), but now, as you know about that, it must drive you crazy. The good news: it's your fault (in this particular case it is my fault, of course). moɔpTi*k*Z cannot calculate coordinates for you. If you are interested in writing Ti*k*Z instead of pointing around you will notice the difference:

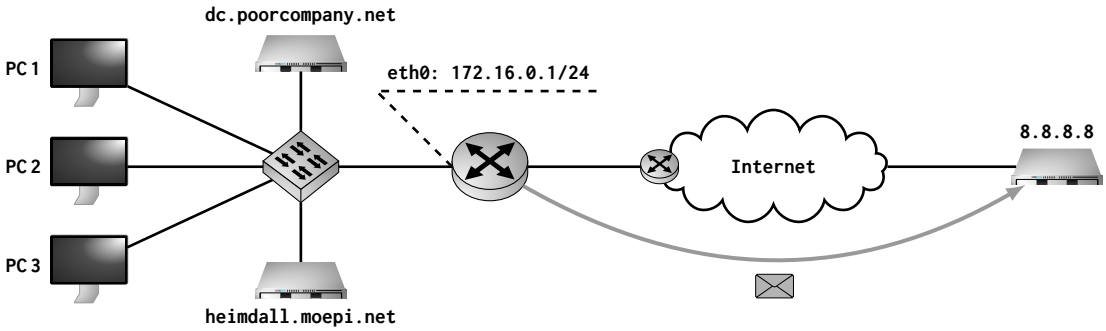


Figure 2: What moɔpTi*k*Z can do for you if you can do your math

3 Package options

The only package option currently defined is `shading=<true|false>`. If true (default), nodes are shaded to create the visual effect of a spotlight. If false, nodes are filled with a flat color instead (the envelopes of the message shapes are flat in either case). The options `shading` and `noshading` are short forms of `shading=true` and `shading=false`. The shading has only a marginal impact on the resulting document size but may be undesirable depending on the personal preference.

4 The shapes

Figures 3 to 5 present the different shapes as printed by `\node[<shape>] at ...` (the label font is typewriter here, but that's up to you). The default label shows the name of the corresponding shape. The nodes are drawn with `minimum size=2cm`; the gray box around each shape measures 4 cm × 4 cm. The red crosses mark the positions of `label=<pos>:<labeltext>`; the node text hides the cross below (except for `internet` and `key`). The gray lines end at the automatically calculated border points; the compass anchors lie there as well, e. g. `(x.north east)` is `(x.45)` (except for `internet`), `base` on the baseline of the node text and `mid` .5ex above it.

If the node option `minimum size` is omitted, nodes will be drawn in a bounding box of 1 cm × 1 cm. The line widths of the shapes scale with their size; the line width of the picture (e. g. `thick` or `line width=<width>`) does not affect them, it is meant for the links between the nodes. The only exception is the outline of `internet`. Since not all shapes fill their bounding box, the node text is not placed at a fixed position but directly below the visible shape, exactly where `label=below:<text>` would put a label (exceptions: `internet` and `key`, see Sections 4.2 and 4.4). Hence the text of a server is closer to the node center than the text of a client, but the gap between shape and text is the same for all shapes. Like a label, the node text counts for the size of the picture.

Some shapes combine others: `mlswitch` is a multilayer switch, i. e. a switch with the arrows of a router on its lid, `fwrouter` is a router with a firewall on top (Section 4.3), and `rack` shows four servers on top of each other. These shapes have the size, the anchors and the border of the shapes they are based on, hence they can replace them in a figure without moving anything. The exception is `rack`, which is as high as a router or a switch. Similarly, `usergroup` shows three users and `wlanmodem` is a modem with two antennas, e. g. a home router with built-in modem and WLAN.

4.1 Changing node colors

There are very few occasions where you might want to highlight a specific node by changing its color. Or you are upset by the default mark color (black), i. e., the color of arrows on top of some shapes. You may change it at will: all shapes support custom colors using the `fill=<color>` option. In addition to that, the shapes with arrows (`hub`, `switch`, `mlswitch`, `router`, `fwrouter`), the message shapes and `internet` also let you change the color of their arrows or lines with the `draw=<color>` option. The shape names take no value: put `fill`, `draw` and `minimum size` after the shape name, as in `\node[router,fill=red]`; in front of it, the defaults of the shape may override them. The sheet

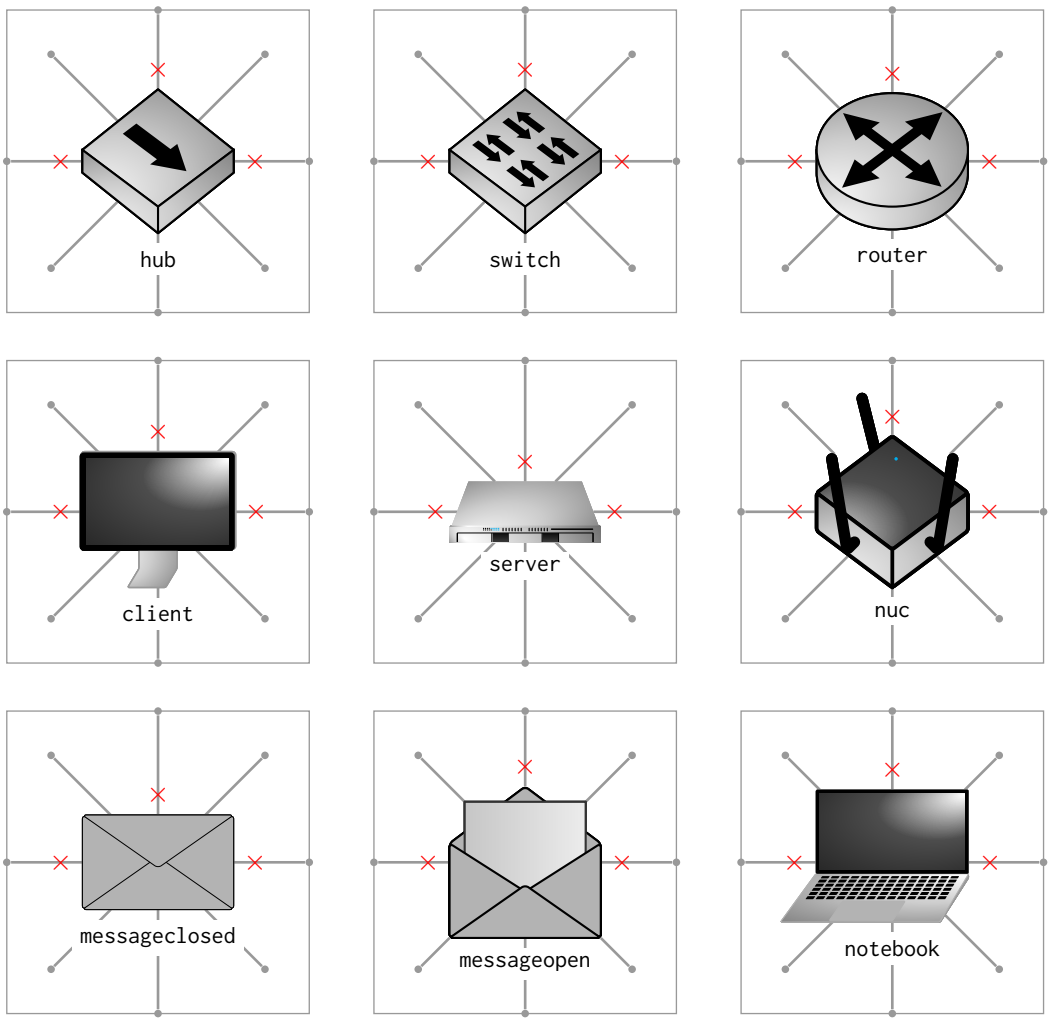


Figure 3: Shapes currently defined

of paper in `messageopen` has a color of its own, `paper color=<color>` (default black): it is shaded in light tones of this color (also with `noshading`) and outlined in `<color>!80`.



```
\node[router,fill=blue,draw=orange] at (0,0) {};
```



```
\node[messageclosed,fill=red!40,draw=red] at (0,0) {};
```



```
\node[messageopen,fill=orange!40,paper color=blue,draw=orange] at (0,0) {};
```

The effect of the `draw` option depends on the actual shape: the shapes with arrows will change the color of their marking (the arrows) while `messageopen` and `messageclosed` change the line color of the envelope and `internet` the color of its outline (Section 4.2); all other shapes ignore the `draw` option. This behavior may seem inconsistent but is intended – give it a try, or come up with a fix.

4.2 The internet shape

The `internet` shape draws a cloud with a configurable number of puffs. Unlike the other shapes, the node text is placed in the middle of the cloud and the cloud grows with its text.

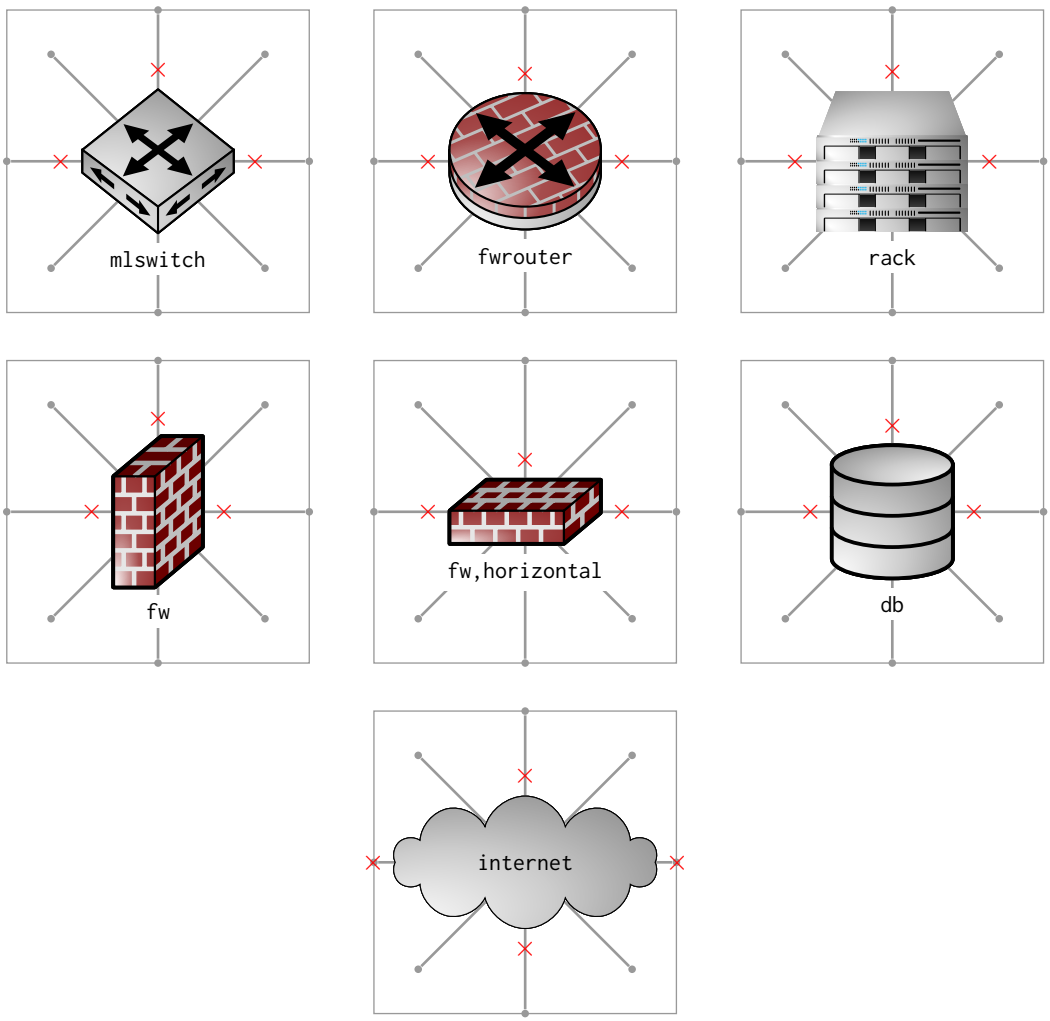


Figure 4: Shapes currently defined, continued (`internet` with `minimum width=3.5cm`, see Section 4.2)




```
\node[internet] {Internet};
```




```
\node[internet,puffs=7,fill=blue] {Internet};
```




```
\node[internet,puffs=13,minimum width=3cm,empty,draw=red] {Internet};
```

The following options are supported:

- `puffs=<n>`: number of puffs (default 10)
- `minimum width`, `minimum height`: enlarge the cloud
- `aspect=<ratio>`: ratio of width to height (default 2)
- `fill=<color>`: color of the cloud, shaded or flat depending on the package option `shading`
- `empty` or `fill=none`: the cloud is not filled
- `draw=<color>`, `line width=<width>`: color and width of the outline

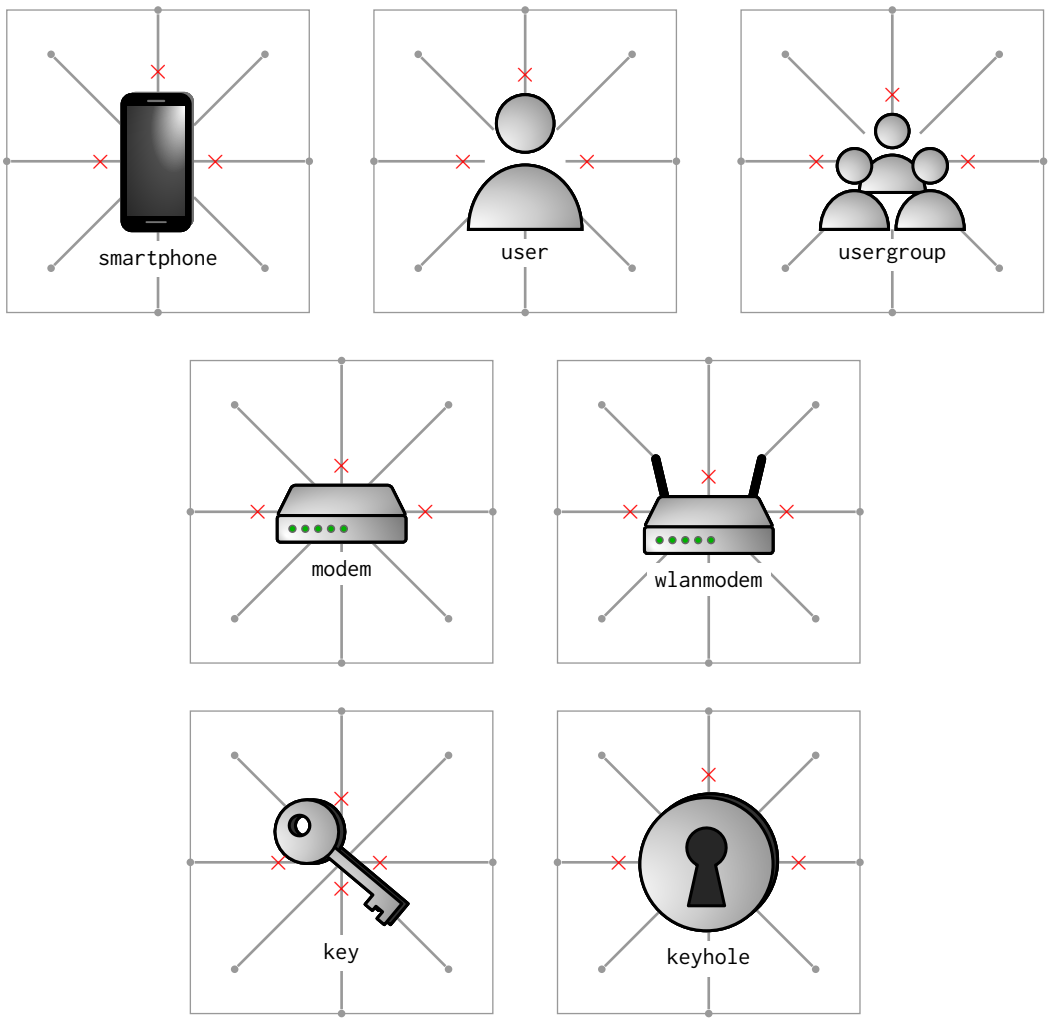


Figure 5: Shapes currently defined, continued (for `key` see Section 4.4)

These colors affect only the cloud, use `text=<color>` for the text.

The geometry, the border and the anchors `center`, `mid`, `base` and the compass anchors (`north`, `north east`, ...) are taken from the `cloud` shape of the Ti*k*Z library `shapes.symbols`, which is loaded by moэpTi*k*Z. An `internet` node does not have the anchors `puff 1` to `puff <n>` of the cloud. Use at least 5 puffs. Wide clouds (3 cm and more) need about 8 puffs, otherwise the puffs at the sides degenerate.

4.3 Firewalls

The `fw` shape draws a brick wall. By default the wall stands upright and shows its narrow end, the option `horizontal` lays it down. The graph drawing library `graphdrawing` of Ti*k*Z (LuaLaTeX only) has an option `horizontal=<tail> to <head>` as well, which causes an error in the options of a picture or scope, e. g. `\tikz[layered layout,horizontal=a to b]`; write `/graph drawing/horizontal=a to b` there instead. Conversely, with this library loaded, the option `horizontal` of a node inside `\graph` goes to graph drawing and the firewall silently stays upright; write `/tikz/horizontal` there, e. g. `a[fw,/tikz/horizontal]`. The `fwrouter` shape is a router with a firewall on top. The bricks are drawn in the fill color of `fw` and in the color `fw color=<color>` of `fwrouter`, both `red!50!black` by default (dark enough to stand out in grayscale); `fill` and `draw` of `fwrouter` work as for a router.



```
\node[fw] at (0,0) {};
```



```
\node[fw,horizontal,fill=black!40] at (0,0) {};
```



```
\node[fwrouter,fw color=orange,draw=white] at (0,0) {};
```

4.4 Keys

The shapes `key` and `keyhole` are meant for security topics, e. g. private and public keys in different colors. Both are drawn in their fill color with some depth to the back. The key lies diagonally in its bounding box, so the border points at 90° and 270° lie on the blade in the middle of the key. Hence its node text is placed below the lowest point of the key (the bit) instead of where `label=below:<text>` would put a label, and labels above or below overlap the key (Figure 5). Use the node text or labels at other positions for keys.



```
\node[key,fill=red] at (0,0) {};
```



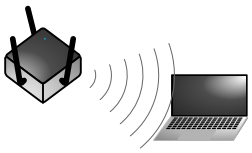
```
\node[key,fill=green!60!black] at (0,0) {};
```



```
\node[keyhole,fill=yellow!80!black] at (0,0) {};
```

4.5 Wireless links

The path style `waves` draws concentric arcs instead of a line, e. g. for a wireless link. The arcs are centered at the start of the path and grow up to its end; between two nodes they start at the border of the first one. The option `waves=<n>` sets the number of arcs (default 6), `wave angle=<angle>` their opening angle (default 70). By default the arcs are thin and gray; an own color or line width has to be given after `waves`.



```
\draw[waves] (a) to (n);
```



```
\draw[waves=10,wave angle=40] (0,0) to (2.5,0);
```



```
\draw[waves,blue,line width=.8pt] (0,0) to (2,1);
```

5 The commands

moɔpTi*k*Z provides three commands to make your life with Ti*k*Z easier:

1. `\tikzextractx{(x,y)}{\<macroname>}` stores the x-component of a coordinate in `\<macroname>`.
2. `\tikzextracty{(x,y)}{\<macroname>}` stores the y-component of a coordinate in `\<macroname>`.
3. `\moeptikzset{<key>=<value>}` sets the internal key `<key>` to value `<value>`. Currently, the only defined key is `shading=<true|false>`.

6 Known incompatibilities

Some keys of moɔpTi*k*Z have the names of keys or shapes of other packages, and only one of them can take effect; moɔpTi*k*Z warns about these clashes at `\begin{document}`.

- `circuitikz` defines the key `switch` (a switch in a circuit) and the shape `waves`. Whichever of moɔpTi*k*Z and `circuitikz` is loaded last wins the key `switch`; with moɔpTi*k*Z loaded last, `to[closing switch]` still draws the switch of `circuitikz`. With its option `compatibility`, `circuitikz` calls its switch `*switch`, which avoids the clash. The path style `waves` of moɔpTi*k*Z hides the shape `waves` of `circuitikz`; write `shape=waves` for the latter.
- `tikz-feynman` and the graph drawing library use the key `horizontal` for the orientation of a graph, e. g. `horizontal=a to b`. Write it with its path in the options of a picture, scope or diagram, e. g.

  ```
  \feynmandiagram[/graph drawing/horizontal=a to b] {...};
  ```

  Section 4.3 explains this for the graph drawing library. Without LuaTeX (pdfLaTeX, XeLaTeX, latex with dvips), load `tikz-feynman` before moɔpTi*k*Z; otherwise `horizontal` only warns that it needs LuaTeX.

The warnings can be hidden with the package `silence` and `\WarningsOff[moeptikz]`.