Label Placement, Port Placement and Node Sizing

This page documents how KLay Layered implements label placement, port placement, and node sizing.

Contents

Introducing Important Stuff

When talking about label placement and node sizing, it helps to know what the two actually are. Let's start with node sizing:

Node Sizing

Node sizing is the act of determining the size of a node. In KIML, a layout algorithm can be granted different kinds of freedom in calculating the size of a node. The different kinds are expressed through a subset of the following options, as defined (and documented) in the `SizeConstraint` enumeration:

- `PORTS`
- `PORT_LABELS`
- `NODE_LABELS`
- `MINIMUM_SIZE`

On the one extreme, the subset can be empty, thereby fixing the node size. On the other extreme, the set can contain all options, thereby giving the layout algorithm the maximum amount of flexibility.

The way the node size is determined can also be influenced by specifying a subset of the following options, as defined (and documented) in the `SizeOptions` enumeration:

- `DEFAULT_MINIMUM_SIZE`
- `MINIMUM_SIZE_ACCOUNTS_FOR_INSETS`
- `COMPUTE_INSETS`

Label placement can be divided into port label placement and node label placement:

Port Placement

Port placement is the act of determining the position of ports. This includes determining the side of their node where the port gets attached, determining an order between ports on the same side, and determining the final position of each port. There are different levels of constraints on placing ports, as defined (and documents) in the `PortConstraints` enumeration:

- `FREE`
- `FIXED_SIDE`
- `FIXED_ORDER`
- `FIXED_RATIO`
- `FIXED_POS`

Port placement can take place after crossing minimization, since the order of ports must be known and port placements needs to be fixed before node placement.

Label Placement

Label placement is the act of determining the position of labels, with the aim of keeping readability high. The two most critical objectives in label placement are the following:

1. Avoiding overlaps between labels and other graphical objects, including other labels.
2. Making sure that each label is closer to its associated graphical feature than to other, unrelated graphical features.

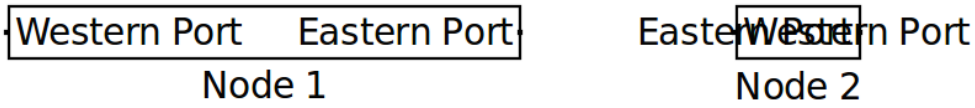
KLay Layered distinguishes three kinds of labels:

1. Node labels
2. Port labels
3. Edge labels
    a. Tail labels
    b. Mid labels
    c. Head labels

Node labels and port labels can be placed inside or outside of their particular node.

Relationship Between Label Placement, Port Placement, and Node Sizing

At first glance, label placement and node sizing are two separate problems. However, out of the three types of labels we currently support, two have considerable influence on the size of nodes (node labels and port labels, but you already figured this out yourself). Well, in fact, that's not completely true. It's not the labels that influence the size of nodes, it's the placement of labels. And if we didn't care for readability, the placement wouldn't influence node size at all. But we do. Take this simple example:



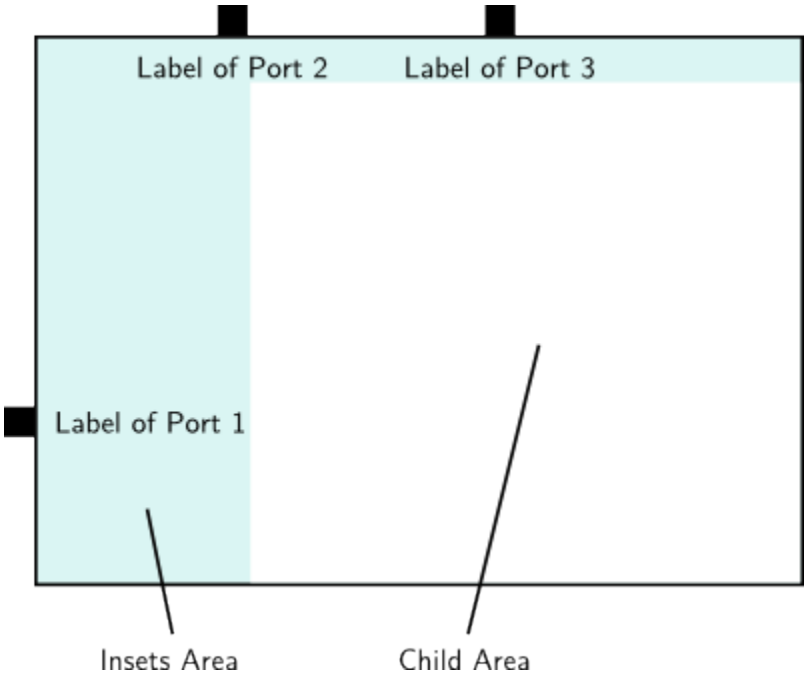
The two nodes have two labelled ports each. Let's assume port positions to be fixed, and labels to be placed inside the nodes. While the port labels fit perfectly fine into Node 1, Node 2 is clearly too narrow for them to be placed without overlaps: we would have to increase the width of the node. Label placement influences node sizing.

Matters get more complicated if we allow port positions to be changed. If the western port is moved upwards and the eastern port downwards, the labels don't overlap any more. Thus, label placement also influences port placement.

Just how much and in what ways the three influence each other is one source of complexity. One important task in implementing label placement, port placement, and node sizing is to determine the cases where we simply give up. If the user gives us fixed node sizes and fixed port positions, he cannot expect us to find an overlap-free label placement if port labels are to be placed inside the node.

Anatomy of a Node

When it comes to label placement and node size calculations, we can think of a node as having the following anatomy:



The insets area is the part of the node where port and node labels are placed (if they are placed inside the node as opposed to outside the node). The child area is what is left of the node with the insets subtracted. With the `SizeOptions.COMPUTE_INSETS`, KLay Layered can compute the child area and return it. This makes it easy to use in cases where the child area of the node is going to be used for displaying graphics or further text. If a minimum size is set on the node and the options `SizeConstraint.MINIMUM_SIZE` and `SizeOptions.MINIMUM_SIZE_ACCOUNTS_FOR_INSETS` are set, the minimum size will effectively only apply to the child area. Thus, KLay Layered can be told to ensure that the child area of a node has at least a given size, whatever space the port labels and node labels require.

Limitations of Our Algorithms

We cannot support everything – actually, many combinations of label placement, port placement, and node sizing constraints don't even make much sense. So here's a (possibly still growing – science can only do so much...) list of things we don't support:

- More than one port label.
- Different placements for each node label. As of now, node labels are placed by arranging them in a vertical stack at the desired node label position.

General Approach

The general approach to solving label placement, port placement, and node sizing follows the following general pattern:

1. Place port labels
2. Calculate required insets
3. Resize node
4. Place ports
5. Place node labels
6. Calculate insets

Each task will be described in more detail in the following.

Place Port Labels

Each port's labels are placed. The exact placement strategy may differ depending on node sizing and port placement constraints. The easiest way of placing port labels, however, is not to care about other constraints just yet.

Calculate Required Insets

Before resizing the node, we will need to find out how large it will have to be at the minimum. This requires iterating over ports and labels to find out how much space they will need, which is what this step is all about.

Resize Node

With the insets calculated, this step resizes the node, ensuring that the node size constraints are met. If the node is to have a minimum size, it may be resized accordingly, for instance. If ports are to be taken into account for the size calculation, the node size is adjusted to accommodate for its ports.

Place Ports

All ports are placed. If node sizing constraints allow for it, ports are placed in a way that ensures that port labels don't overlap each other. This is only possible, though, if `SizeConstraint.PORT_LABELS`is set. In all other cases, we can only hope that the result is free of overlaps.

The only cases where we even have to talk about the placement of ports are those where their position is not fixed. That more or less leaves us with the three port constraints `FREE`, `FIXED_SIDES`, and `FIXED_ORDER`. The first is reduced to the second by assigning ports to the eastern or western side depending on the number of incoming and outgoing edges (if a port has more outgoing edges than incoming ones, it can be regarded as an output port). The second is reduced to the third during crossing minimization, in an attempt to find an order that will yield the fewest amount of edge crossings. Thus, the only case of real interest to us is `FIXED_ORDER`.

Port placement is influenced in the following ways by label placement and node sizing:

- Label Placement: Ports should be placed in a way that avoids label overlaps.
- Node Sizing: If node sizing doesn't pay any attention to ports, port placement options may be severely restricted.

Place Node Labels

With the node size set and ports placed, the node labels are now placed.

Layout Options Influencing This Stuff

The following layout options influence how the algorithm places ports and labels and calculates the node size:

Option	Target	Description
`LayoutOptions.NODE_LABEL_PLACEMENT`	Node	Determines where node labels are placed. A valid set of values contains exactly one constant from each of the following sets of constants: • `NodeLabelPlacement.INSIDE` and `NodeLabelPlacement.OUTSIDE` • `NodeLabelPlacement.H_LEFT`, `NodeLabelPlacement.H_CENTER`, and `NodeLabelPlacement.H_RIGHT` • `NodeLabelPlacement.V_TOP`, `NodeLabelPlacement.V_CENTER`, and `NodeLabelPlacement.V_BOTTOM`
`LayoutOptions.PORT_LABEL_PLACEMENT`	Node	Determines where port labels are placed: inside or outside their node.
`LayoutOptions.LABEL_SPACING`	Graph	Determines the amount of space left between labels and the objects they label.
`LayoutOptions.SIZE_CONSTRAINT`	Node	The amount of freedom in determining the size of a node.
`LayoutOptions.SIZE_OPTIONS`	Node	Options for node size calculation.
`LayoutOptions.MIN_WIDTH`	Node	The minimal width of a node. If set, overrides the default minimal width.
`LayoutOptions.MIN_HEIGHT`	Node	The minimal height of a node. If set, overrides the default minimal height.
`LayoutOptions.PORT_CONSTRAINTS`	Node	Freedom in placing ports.
`Properties.PORT_SPACING`	Node	How much space should be left between ports.