Multicast 组播方案

```
local mc = require "skynet.multicast"
```

引入 multicast 模块后，你可以使用 skynet 的组播方案。你可以自由**创建一个频道**，并可以向其中**投递**任意**消息**。频道的**订阅者可以收到投递的消息。**

你可以通过 new 函数来创建一个频道对象。你可以创建一个新频道，也可以利用已知的频道 id 绑定一个已有频道。

```
local channel = mc.new()  -- 创建一个频道，成功创建后，.channel 是这个频道的 id 。local c2 = mc.new {
  channel = channel.channel,  -- 绑定上一个频道
  dispatch = function (channel, source, ...) end,  -- 设置这个频道的消息处理函数
}
```

如上面的例子，new 函数可以接收一个参数表。**channel 是频道 id ，dispatch 是订阅消息的回调函数**。如果你不给出 channel id ，则新创建出一个频道来。

通常，由一个服务创建出频道，再将 .channel 这个 id 通知别的地方。获得这个 id 的位置，都可以绑定这个频道。

channel:publish(...) 可以向一个频道**发布消息**。消息可以是任意数量合法的 lua 值。

光绑定到一个频道后，默认并不接收这个频道上的消息（也许你只想向这个频道发布消息）。

你需要先调用 **channel:subscribe() 订阅消息**。

如果**不再想收到该频道的消息**，调用 **channel:unsubscribe** 。

当一个频道不再使用，你可以调用 **channel:delete() 系统回收频道**。注：多次调用 channel:delete 是无害的，因为 channel id 不会重复使用。在频道被销毁后再调用 subscribe 或 publish 等也不会引起异常，但订阅是不起作用的，消息也不再广播。

组播的原理

当只考虑一个进程时，由于同一进程共享地址空间。当发布消息时，由同一节点内的一个 multicastd 服务接收这条消息，并打包成一个 C 数据结构（包括消息指针、长度、引用计数），并把这个结构指针分别发送给同一节点的接收者。

虽然这并没有减少消息的数量、但每个接受者只需要接收一个数据指针。当组播的消息较大时,可以节省内部的带宽。引用计数会在每个接收者收到消息后减一、最终由最后一个接收者销毁。注：如果一个订阅者在收到消息、但没有机会处理它时就退出了。则有可能引起内存泄露。少量的内存泄露影响不大，所以 skynet 没有特别处理这种情况。即同一进程共享地址空间，线程之间的组播使用指针消息，可以节省内部带宽。

当有多个节点时，每个节点内都会启动一个 multicastd 服务。它们通过 DataCenter 相互了解。multicastd 管理了本地节点创建的所有频道。在订阅阶段，如果订阅了一个远程的频道，当前节点的 multicastd 会通知远程频道的管理方，在该频道有发布消息时，把消息内容转发过来。涉及远程组播，不能直接共享指针。这时，multicastd 会将消息完整的发送到接收方所属节点上的同僚，由它来做节点内的组播。

在 cluster 模式下使用

该功能在 cluster 模式下不能直接使用，需要自己实现。