erlang学习(4)

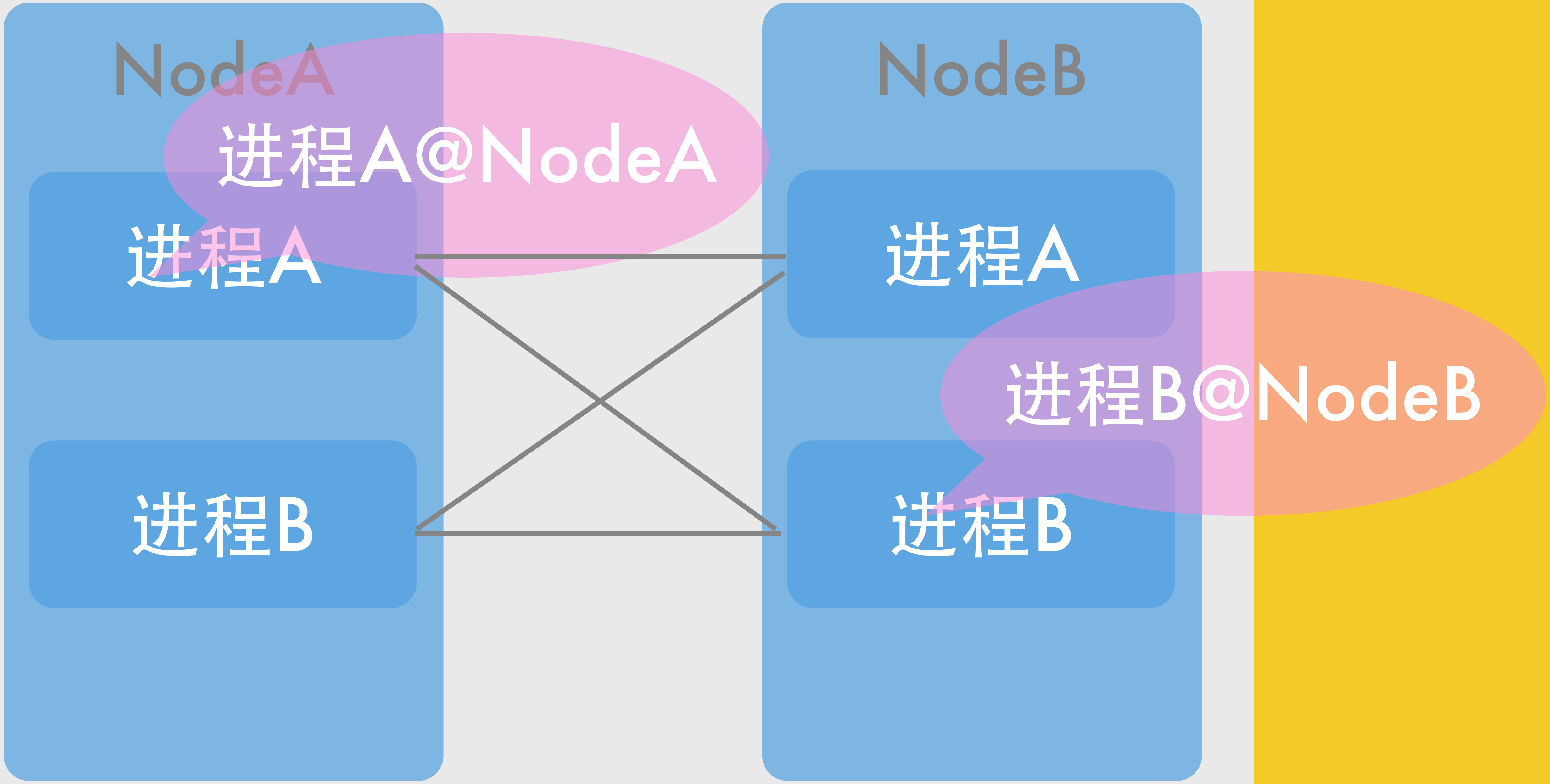
李小红 lixiaohong@gmail.com
黄冬 huangdong@gmail.com

你将会学到

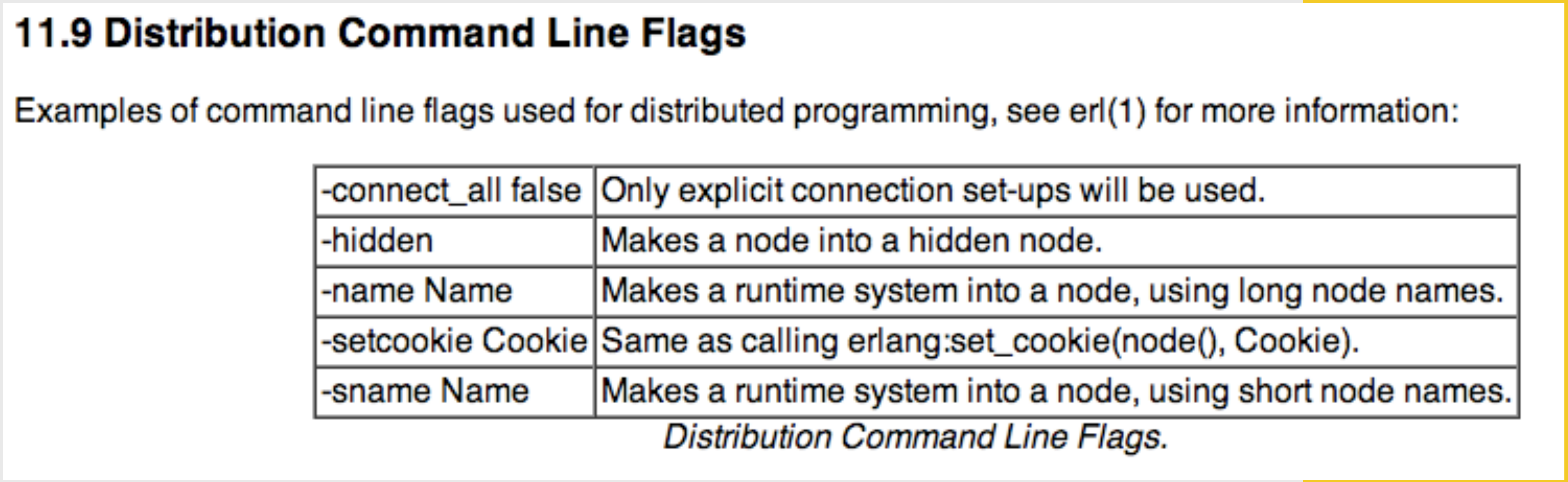
- erlang的分布式调用支持
- 支持分布式的TableServer

erlang的Node

Node的通讯



Node的启动



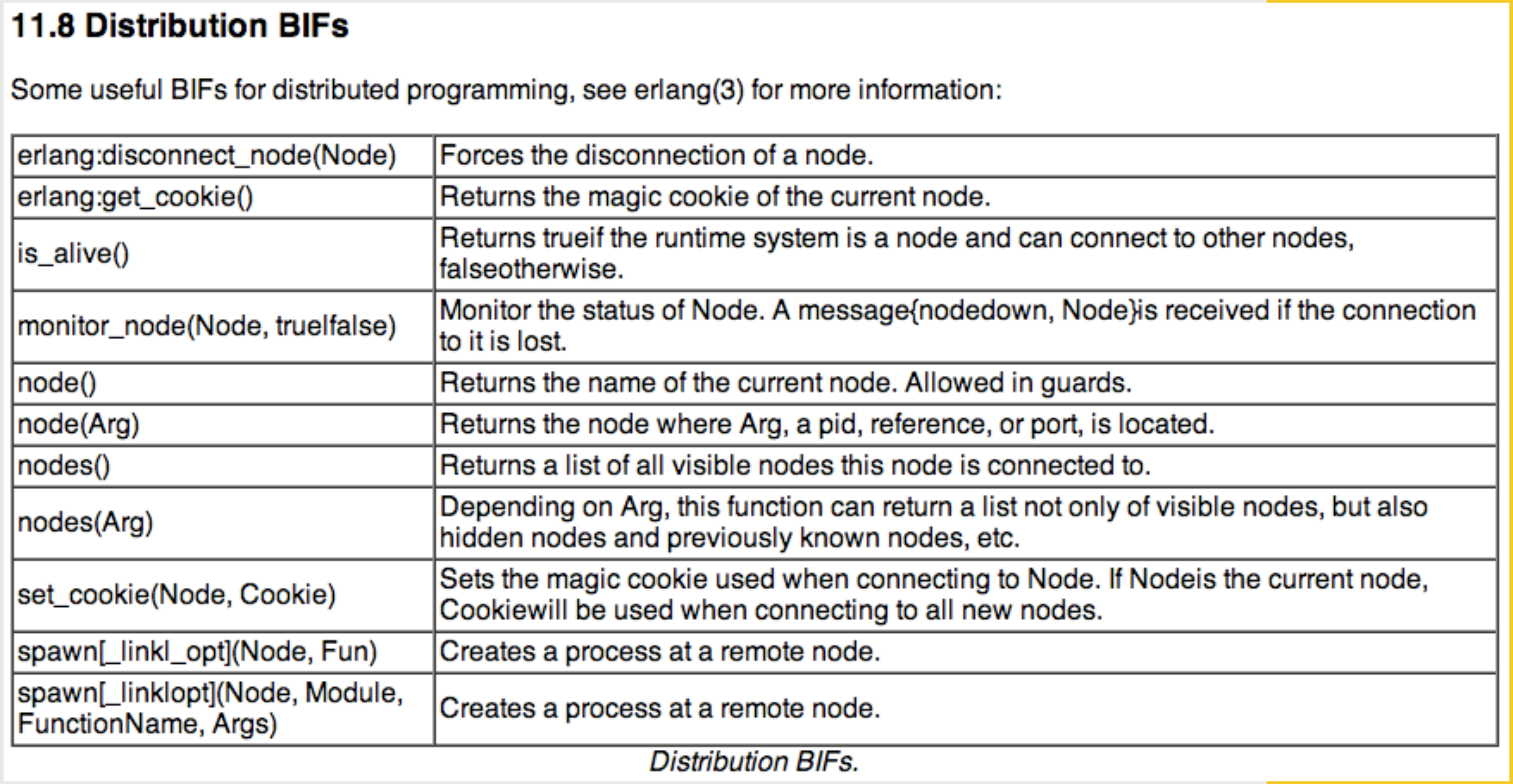
11.9 Distribution Command Line Flags

Examples of command line flags used for distributed programming, see erl(1) for more information:

-connect_all false	Only explicit connection set-ups will be used.
-hidden	Makes a node into a hidden node.
-name Name	Makes a runtime system into a node, using long node names.
-setcookie Cookie	Same as calling erlang:set_cookie(node(), Cookie).
-sname Name	Makes a runtime system into a node, using short node names.

Distribution Command Line Flags.

有关分布式的方法



11.8 Distribution BIFs

Some useful BIFs for distributed programming, see erlang(3) for more information:

erlang:disconnect_node(Node)	Forces the disconnection of a node.
erlang:get_cookie()	Returns the magic cookie of the current node.
is_alive()	Returns trueif the runtime system is a node and can connect to other nodes, falseotherwise.
monitor_node(Node, true|false)	Monitor the status of Node. A message{nodedown, Node}is received if the connection to it is lost.
node()	Returns the name of the current node. Allowed in guards.
node(Arg)	Returns the node where Arg, a pid, reference, or port, is located.
nodes()	Returns a list of all visible nodes this node is connected to.
nodes(Arg)	Depending on Arg, this function can return a list not only of visible nodes, but also hidden nodes and previously known nodes, etc.
set_cookie(Node, Cookie)	Sets the magic cookie used when connecting to Node. If Nodeis the current node, Cookiewill be used when connecting to all new nodes.
spawn[_link|_opt](Node, Fun)	Creates a process at a remote node.
spawn[_link|opt](Node, Module, FunctionName, Args)	Creates a process at a remote node.

Distribution BIFs.

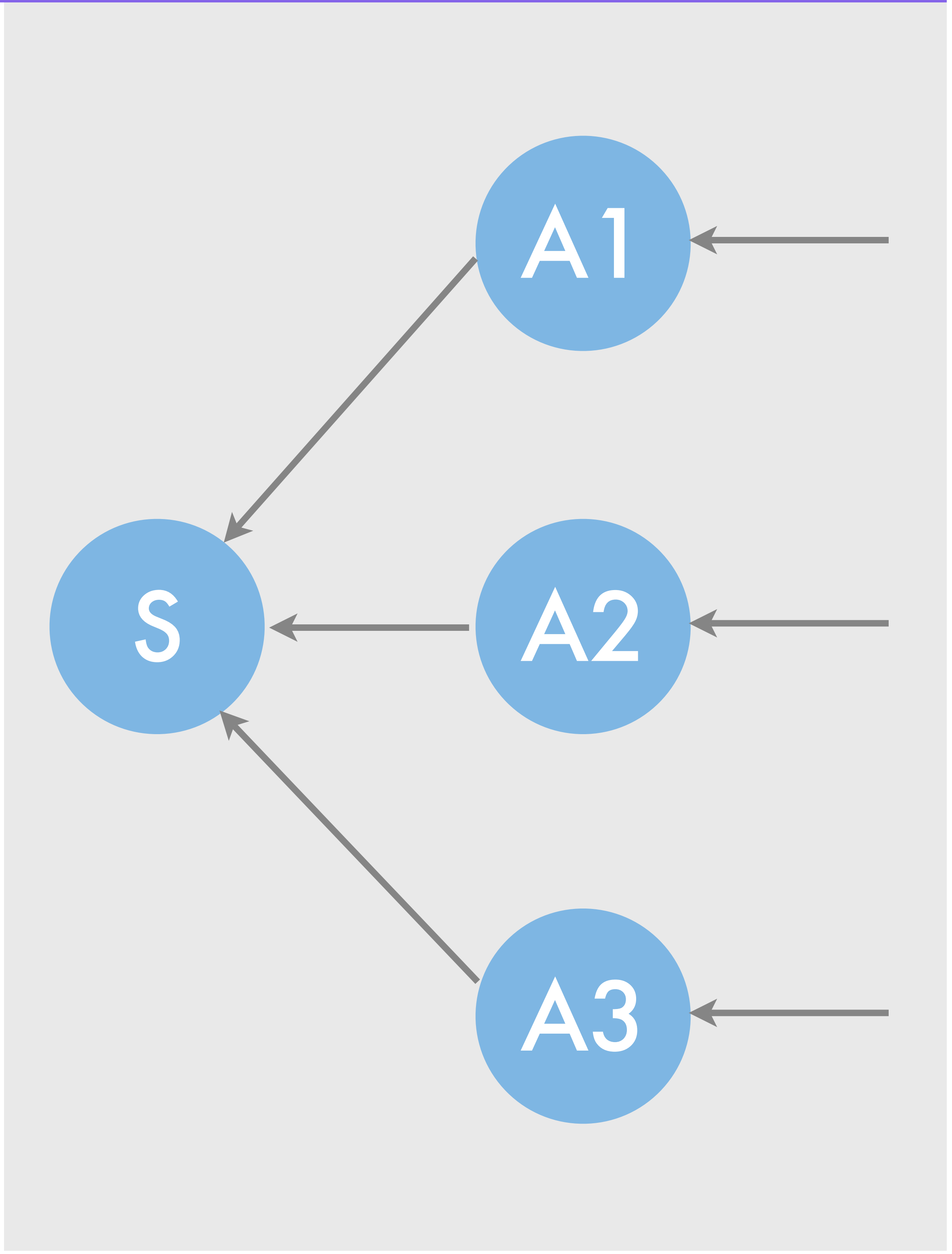
进程通讯的方法

{Pid , Nodename@Processname} ! message

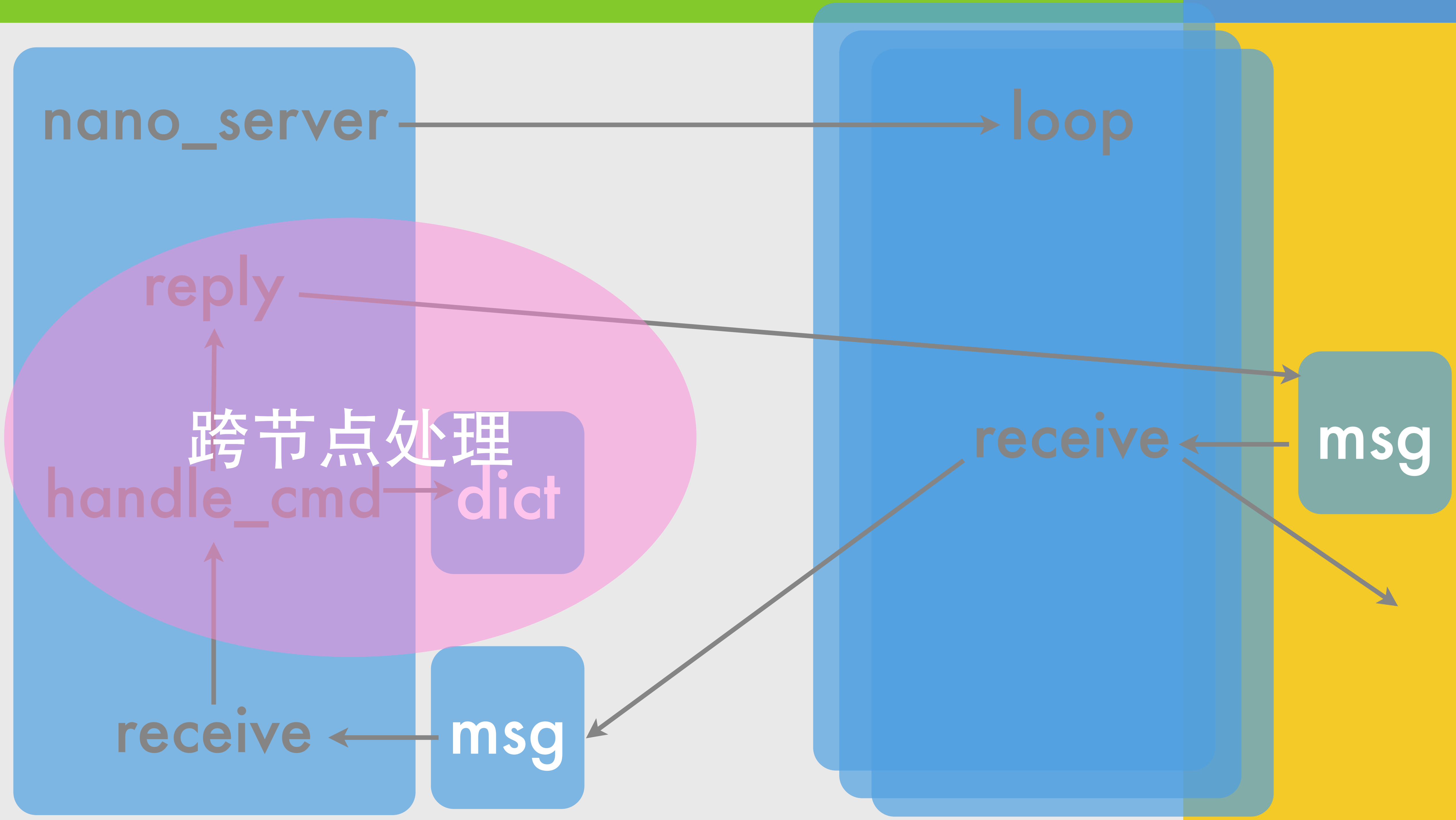
分布式的实作

需求

- 将存储放到一个点
- TCPServer分布在多个点

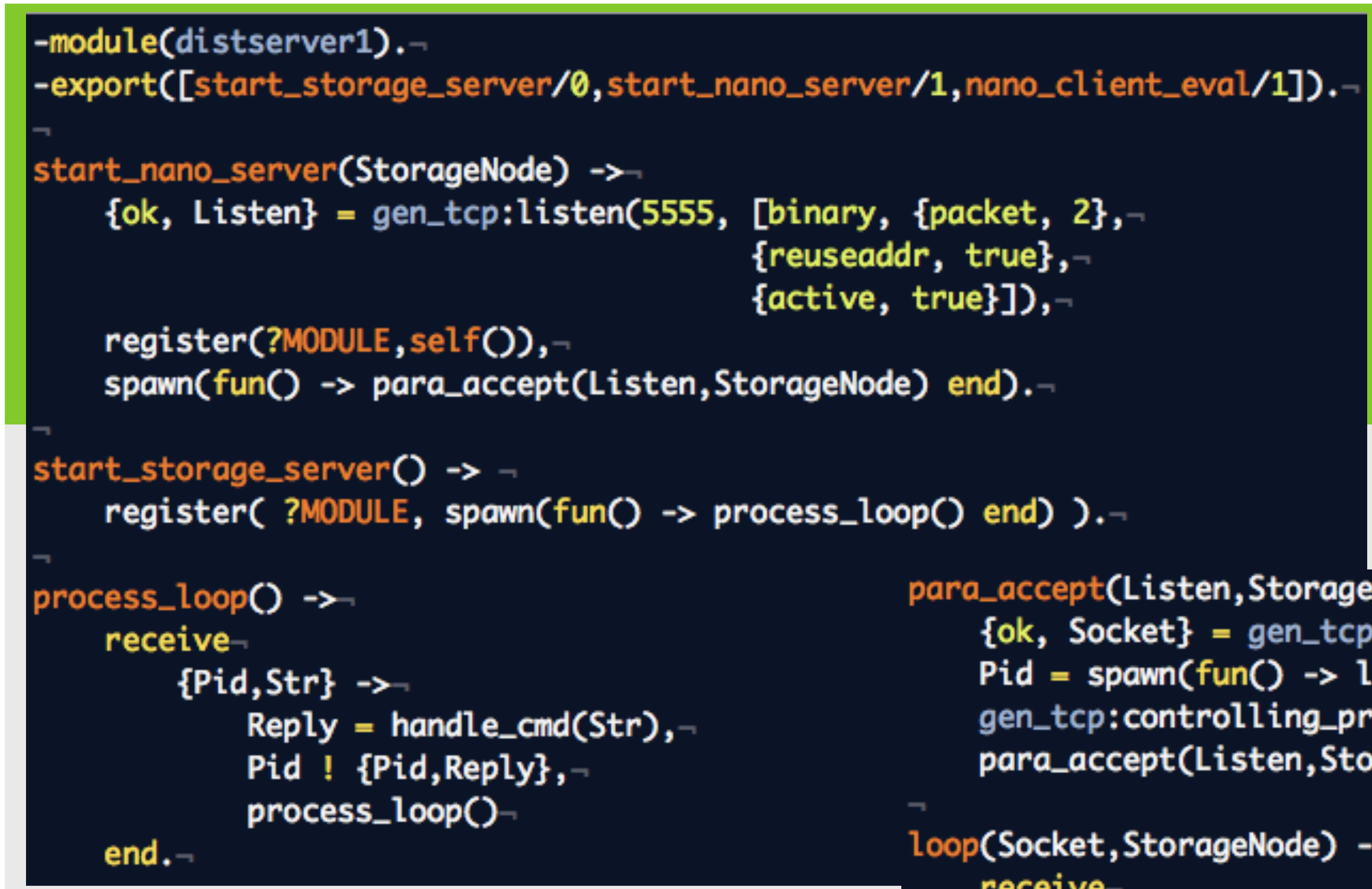


一个节点的Server



开工干活

http://xbaytable.googlecode.com/svn/trunk/erlangtut/ch4/para_server.erl

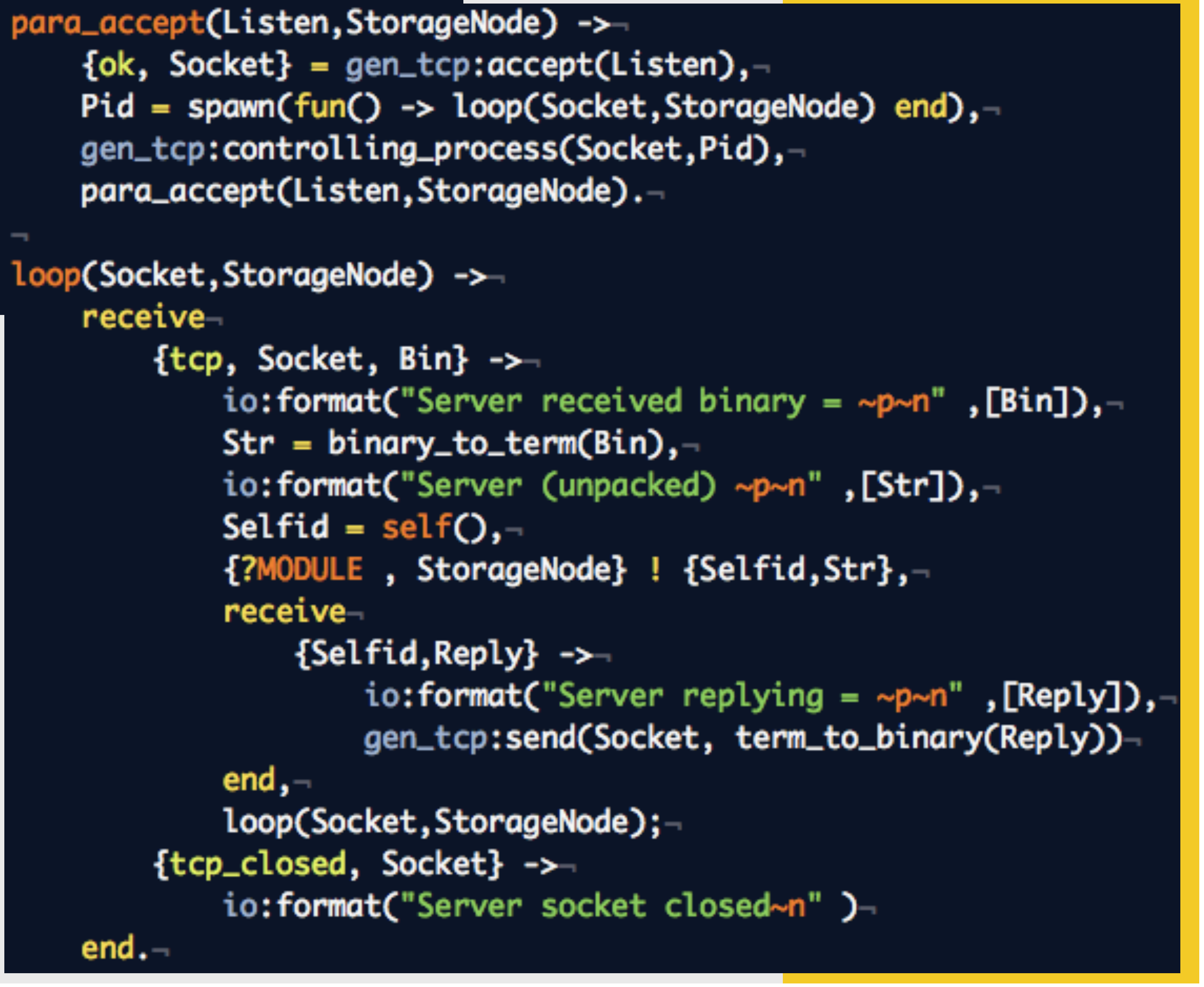


```erlang
-module(distserver1).
-export([start_storage_server/0,start_nano_server/1,nano_client_eval/1]).

start_nano_server(StorageNode) ->
    {ok, Listen} = gen_tcp:listen(5555, [binary, {packet, 2},
                                         {reuseaddr, true},
                                         {active, true}]),
    register(?MODULE,self()),
    spawn(fun() -> para_accept(Listen,StorageNode) end).

start_storage_server() ->
    register( ?MODULE, spawn(fun() -> process_loop() end) ).

process_loop() ->
    receive
        {Pid,Str} ->
            Reply = handle_cmd(Str),
            Pid ! {Pid,Reply},
            process_loop()
    end.
```



```erlang
para_accept(Listen,StorageNode) ->
    {ok, Socket} = gen_tcp:accept(Listen),
    Pid = spawn(fun() -> loop(Socket,StorageNode) end),
    gen_tcp:controlling_process(Socket,Pid),
    para_accept(Listen,StorageNode).

loop(Socket,StorageNode) ->
    receive
        {tcp, Socket, Bin} ->
            io:format("Server received binary = ~p~n" ,[Bin]),
            Str = binary_to_term(Bin),
            io:format("Server (unpacked) ~p~n" ,[Str]),
            Selfid = self(),
            {?MODULE , StorageNode} ! {Selfid,Str},
            receive
                {Selfid,Reply} ->
                    io:format("Server replying = ~p~n" ,[Reply]),
                    gen_tcp:send(Socket, term_to_binary(Reply))
            end,
            loop(Socket,StorageNode);
        {tcp_closed, Socket} ->
            io:format("Server socket closed~n" )
    end.
```

总结过去的四步

erlang语法

- module
- export
- 方法定义和基本语法
- erlang doc

进程、消息

- 进程的产生和支持
- 消息的通讯
- TCP的应用和使用，对TCP消息的基本处理
- 分布式处理和进程处理

这是一个新起点…