假设通过cleos –u url push action发送一个action。

## Local

Cleos里对push action子命令的处理会一路调到其main.cpp的push\_transaction->call->do\_http\_call，在其中会把该封装好，签好名的action发送给url

## Rpc node

在rpc服务端可以是自己，也可以是其他rpc节点。就看-u url里的url是怎样设的，默认是自己。

Rpc node 需要启用http plugin才能运行rpc 服务。(http\_plugin负责处理来自http,https,socket的请求。需要其他plugin调用其add\_api接口将url和handler注册过去)

在chain\_api\_plugin中，会把chain\_plugin的rw\_api->push\_transaction接口注册给<https://xxxxxxx/chain/v1/push_transaction/>，于是该action最终会引起调用:

Chain\_plugin->read\_write->push\_transaction->

app().get\_method<incoming::methods::transaction\_async>()

该method定义在producer\_plugin. on\_incoming\_transaction\_async

-> process\_incoming\_transaction\_async

**chain.push\_transaction** //调用chain\_plugin,将该trx应用到本地db

-> emit( self.accepted\_transaction, trx);

（即accepted\_transaction\_channel.publish）

->**net\_plugin\_impl.dispatcher.bcast\_transaction(results.second);**

//会将该trx发送出去

emit(self.applied\_transaction, trace);

（即applied\_transaction\_channel.publish）

->history\_plugin. on\_applied\_transaction

send\_response(trace);

-> \_transaction\_ack\_channel.publish

## BP

该trx会在net\_plugin.handle\_message(packed\_transaction\_ptr)中被收到和处理。

如果本地已有该trx，就return。否则调用：

**chain\_plug->accept\_transaction**

->producer\_plugin. on\_incoming\_transaction\_async

后续的操作就跟在RPC的一样了，会调用chain.push\_transaction将其应用到本地db。

不同之处在于：

1. 由于本地已经有该trx，（而且该trx集成到本地db后的回调里会向其他peer发送该trx）所以net\_plugin.dispatcher不会在收到chain\_plugin.emit时再次向外广播该块（所有node都是如此，不止bp）
2. 后续出块后，会将块发送出去，其他node同步该块。普通的node不会广播自己的块。

if(chain接受该trx)

dispatcher->bcast\_transaction //继续向外发送该trx

else

dispatcher->rejected\_transaction