YOU'VE PROBABLY NEVER HEARD OF

THE RETICULUM NETWORK

GUY ROYSE

- @guy.dev
- @guyroyse
- github.com/guyroyse
- guy.dev
- W8GUY

CENTRALIZATION MODELS

Centralized

Everywhere

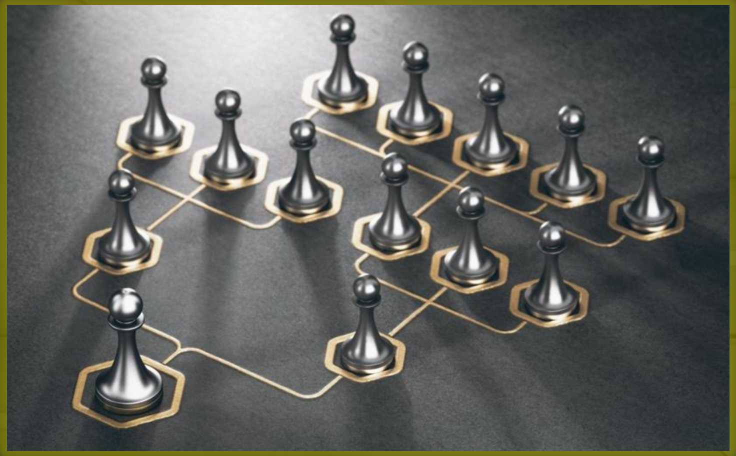
Efficient

Scalable

Brittle

Points of control

CENTRALIZATION MODELS



Everywhere

Robust

Resist control

Inefficient

Scale poorly

Centralized

Decentralized

CENTRALIZATION MODELS

Centralized

Federated

Decentralized

WHY DECENTRALIZED?

Disaster

WHY DECENTRALIZED?

Disaster

Outages

WHY DECENTRALIZED?

Disaster

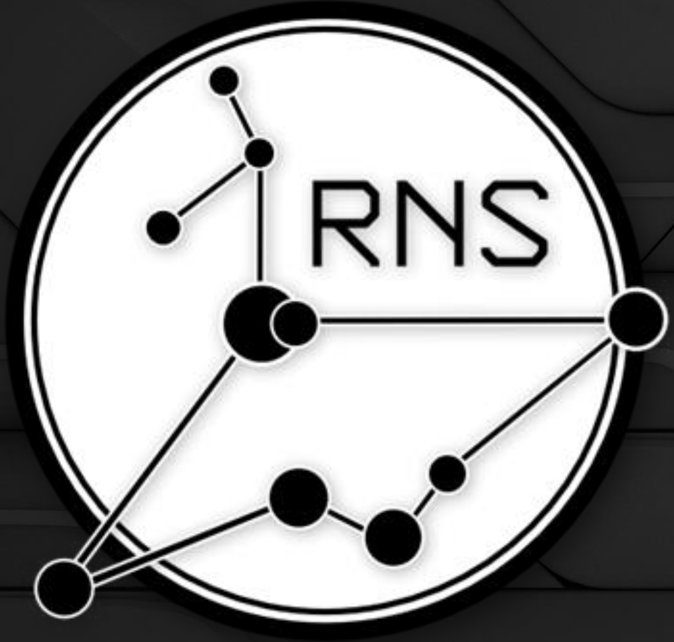
Outages

Control

RETICULUM

Cheap hardware
Low power requirements
Communicate over anything
Fully decentralized
Self-authentication
Fully encrypted

WHAT IS RETICULUM?



Mesh Networking Stack

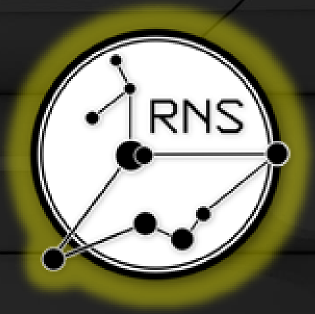
```
$ pip install rns
```

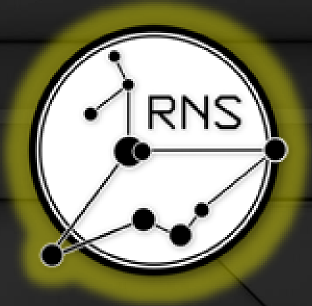
IDENTITY

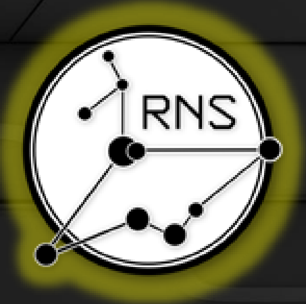
```
$ rnid
```

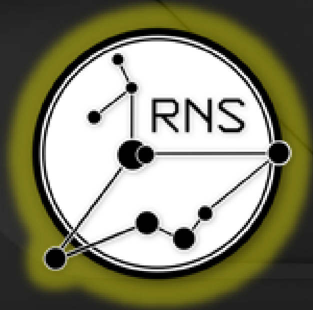
```
identity = RNS.Identity()
identity.to_file(identity_file)
identity = RNS.Identity.from_file(identity_file)
```

INSTANCES





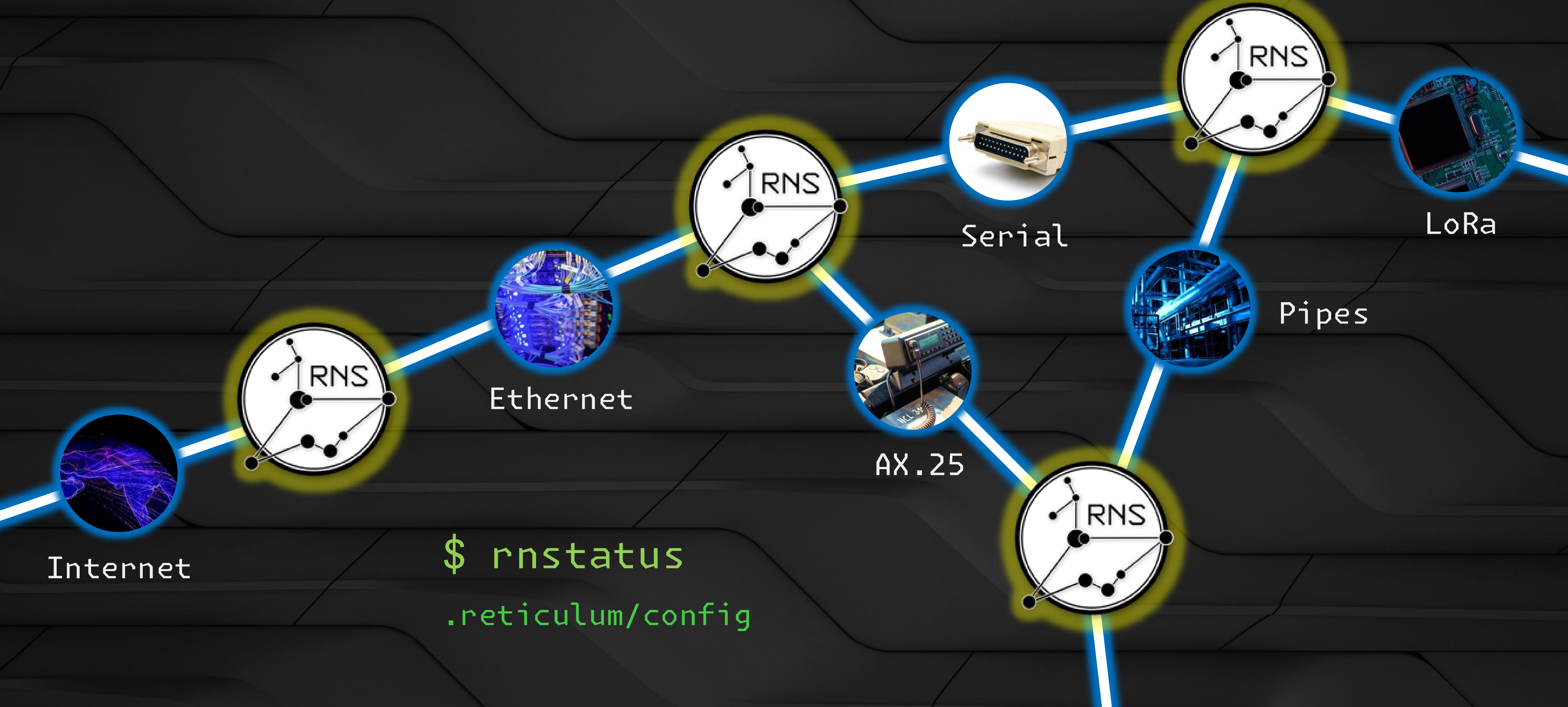




```
$ rnsd
```

```
reticulum = RNS.Reticulum()
```

INTERFACES



Internet

Ethernet

Serial

LoRa

AX.25

Pipes

```
$ rnstatus
```

```
.reticulum/config
```

DESTINATIONS

```
destination = RNS.Destination(
  identity,
  RNS.Destination.IN,
  RNS.Destination.SINGLE,
  app_name,
  aspect
)
```

RNS
RNS
RNS
RNS
>_

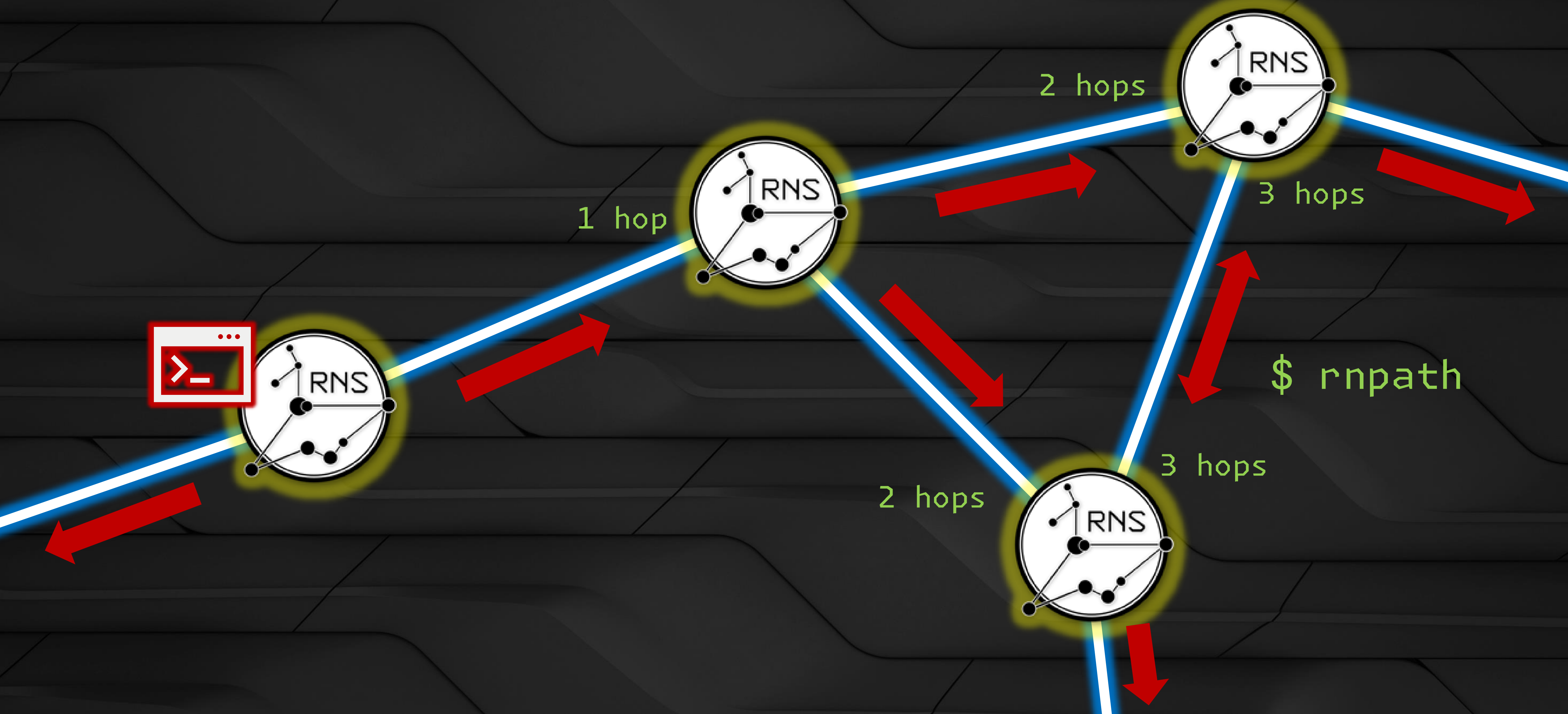
```
$ rnprobe
```

Have an Identity

Have a destination hash

```
rserver.web.<public_key>
4faf1b2e0a077e6a9d92fa051f256038
```

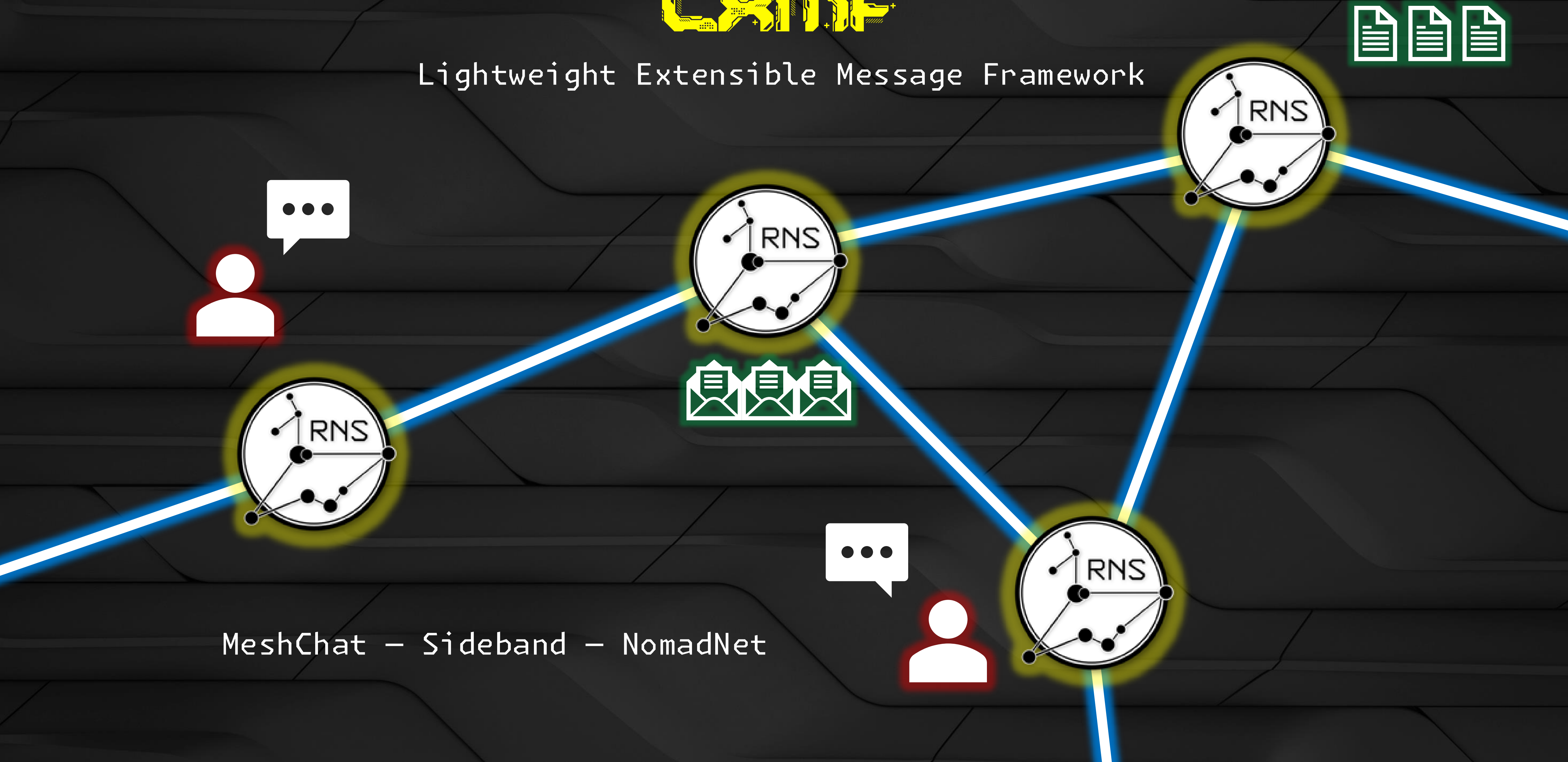
ROUTING



COMMAND LINE DEMO

LXMF

Lightweight Extensible Message Framework



MeshChat – Sideband – NomadNet

NOMADNET DEMO

A SIMPLE SERVER

```
import RNS

# start Reticulum
reticulum = RNS.Reticulum()

# get server Identity
identity = RNS.Identity.from_file(identity_file)
if identity is None:
  identity = RNS.Identity()
  identity.to_file(identity_file)

# app name and aspect
app_name = "rserver"
aspect = "web"

# create server Destination
destination = RNS.Destination(
  identity,
  RNS.Destination.IN,
  RNS.Destination.SINGLE,
  app_name,
  aspect
)
```

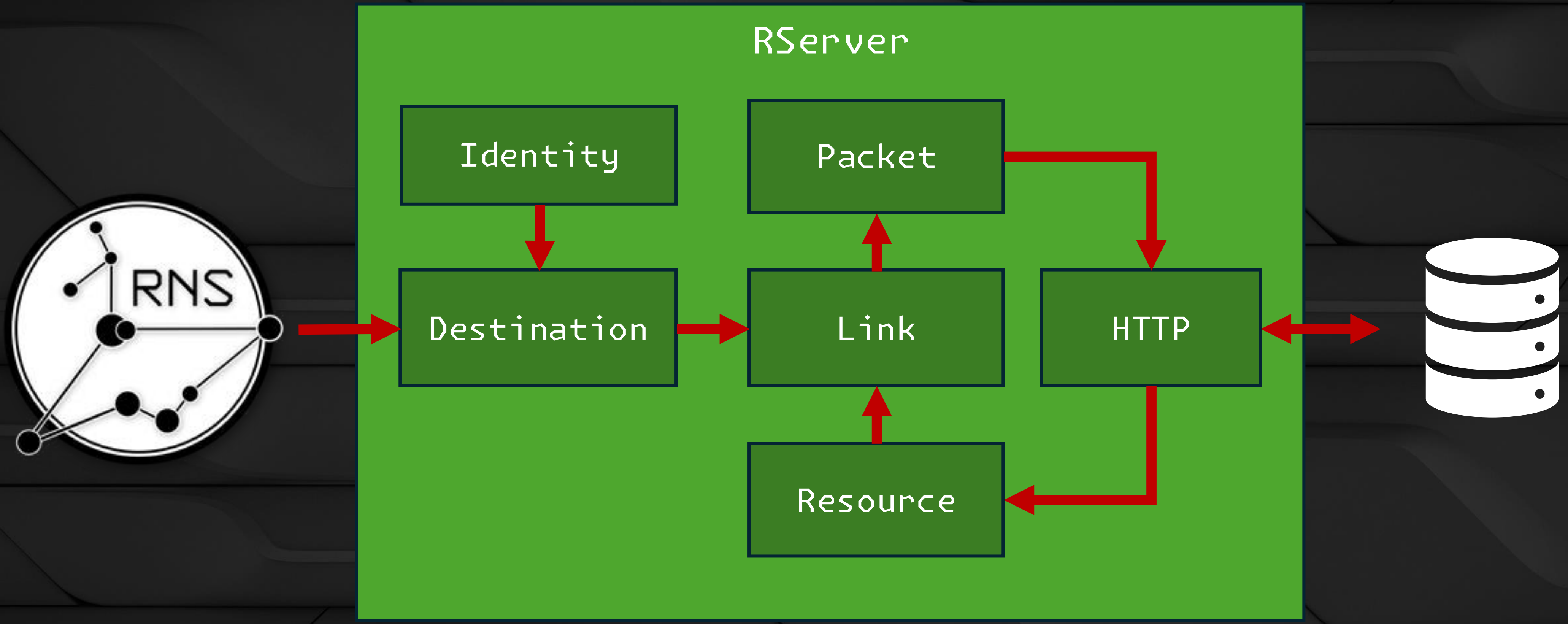
```
# announce Destination to the network
destination.announce()

# set up all the callbacks
destination.set_link_established_callback(
  on_link_established
)

def on_link_established(link):
  link.set_packet_callback(on_packet_received)
  link.set_link_closed_callback(on_link_closed)

def on_packet_received(data, packet):
  response = process_data(data)
  resource = RNS.Resource(response, packet.link)

def on_link_closed(link):
  print("Link closed")
```

A SIMPLE CLIENT

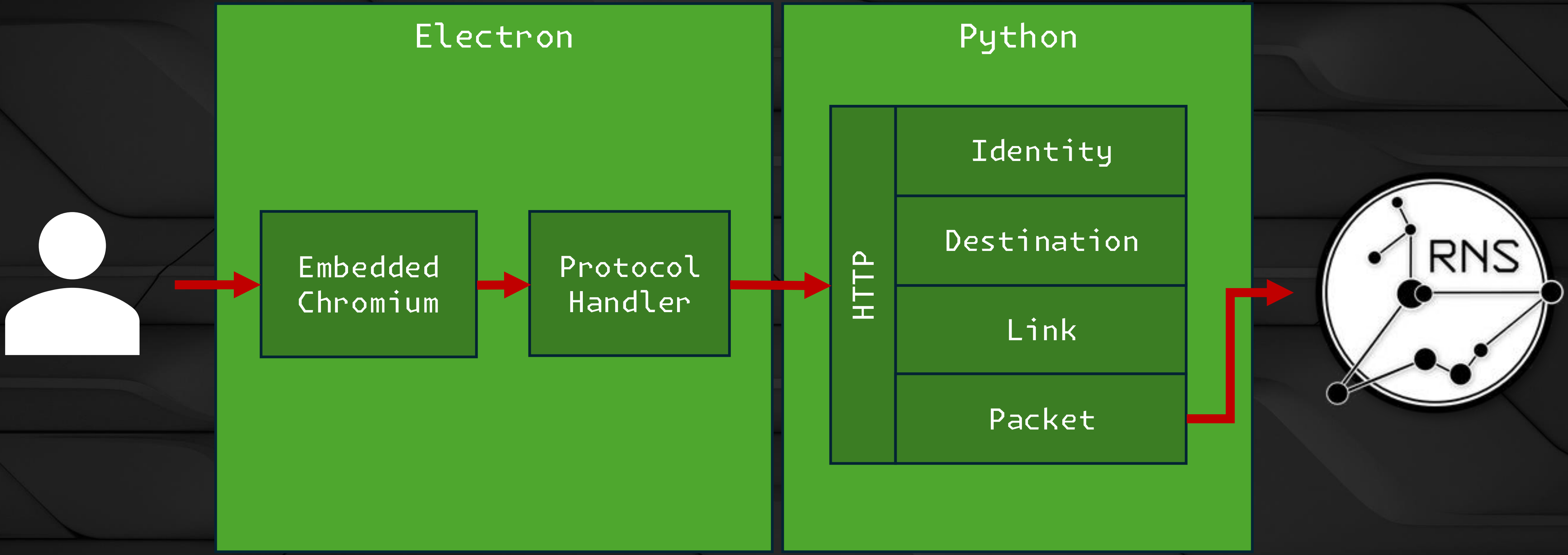
```python
import RNS

# start Reticulum
reticulum = RNS.Reticulum()

# get the destination hash
server_desination_hash = load_destination_hash()
server_identity = RNS.Identity.recall(
  server_desination_hash
)

# query the path from the network
RNS.Transport.request_path(server_desination_hash)
# sleep to wait for path to be resolved

# set up the Destination
server_destination = RNS.Destination(
  server_identity,
  RNS.Destination.OUT,
  RNS.Destination.SINGLE,
  "rserver",
  "web"
)
```

```python
# initiate Link to the Destination
link = RNS.Link(server_destination)
# sleep to wait for Link to be established

# configure the Link
link.set_resource_strategy(RNS.Link.ACCEPT_ALL)
link.set_resource_concluded_callback(
  resource_concluded_callback
)

# send a Packet of data to the Destination
# over the Link
request = build_request_data()
packet = RNS.Packet(link, request)
packet.send()


def resource_concluded_callback(resource):
  if resource.status == RNS.Resource.COMPLETE:
    response_data['content'] = resource.data.read()
  else:
    handle_error()

# teardown the link once we're done
link.teardown()
```

RSERVER


RNS
RServer
Identity
Packet
Destination
Link
HTTP
Resource

MESHBROWSER



RSERVER & MESHBROWSER

DEMO

RESOURCES

Reticulum	reticulum.network
MeshChat	github.com/liamcottle/reticulum-meshchat
RServer	github.com/guyroyse/rserver
MeshBrowser	github.com/guyroyse/mesh-browser

THANKS