Group Address Allocation Protocol (GAAP)

draft-farinacci-pim-gaap-02

IETF Yokohama
March 2023

Dino Farinacci & Mike McBride

Sun Mar 19 16:50:33 PDT 2023

What is GAAP?

- A totally decentralized multicast group address allocation protocol
- There is no central entity that allocates group addresses
- Group addresses allocated are guaranteed to be unique among all GAAP speakers
- GAAP nodes have zero configuration to run the protocol

Design Goals

- The protocol allocates both IPv4 and IPv6 group addresses
- Group addresses allocated will not collide in layer-2 IGMP/MLD snooping switches (multicast MACs unique)
- Works on a single subnet as well as over layer-3 infrastructures, including overlays
- Can coexist with other group allocation protocols by using an IANA GAAP allocation block
- When native multicast not available multicast-capable overlays are used

How Does it Work

- Multicast source & receiver nodes participate in the GAAP protocol
- There is an application specified group name that will map to a group address
- A group address is a hash of the group name
- GAAP nodes send Claim messages to a well-known IANA allocated GAAP group
- A Claim message contains the group name, group address, and timestamp of group address creation

How Does it Work

- Claim messages sent every 1 minute
- If a node is part of a group name and receives a Claim for the group name, it resets the 1 minute timer
- If a node uses a group address found in a Claim for a different group name, a collision has occurred, first creator gets to keep the group address, later creator has to rehash
- Nodes that detect a collision set a 1 second random delay timer to trigger a Claim message
- Other nodes with the same group name, suppress sending triggered Claim
- Nodes part of collided group address, will rehash with string "<group-name>+1" as input to hash, then sends Claim for new group address

Partition Repair

- When a network partition occurs, not all app nodes will see Claim messages
- During this time, collisions can occur and all or part of app nodes could use the same collided address in their partition
- When the partition heals, within less than a minute, Claim messages will be received and collisions will be detected and corrected
- App nodes that have to change their address gets an app callback (supplied in **`gaap.init()`** API call) from the GAAP library
- Allows apps to start using the new address (receivers leave old group and join new group, sources start sending to new group)

Protocol Scale

- There is at most 1 Claim per group name (regardless of the number of nodes using the group name)
- There is at most 1 Claim per collided group when detected
- Collided addresses are not used and converge quickly since collision detector triggers Claim

Protocol Security

- All messages are encrypted with Chacha20 cipher
- Default key is group name
- Can run in hybrid mode
- Protocol can detect bad actors (sending too fast, forging timestamp, etc)
- Rekeying can occur to exclude bad actors
- Overlays can help to suppress bad actors close to source
- “Re-grouping” can help even more, so input queues don’t fill

Protocol API

- The GAAP protocol is lightweight enough to run as a library in the multicast app OS process

- API calls:

```
gaap.init(callback_fn)
address = gaap.allocate(group_name)
gaap.release(group_name)
gaap.close()
```

- Apps which participate in the same group, are started with the same group-name

- Apps can use multiple group-names since they may have requirements for multiple group address use

- A lightweight app can use a lighter-weight Restful API to a GAAP proxy node that runs the protocol (app doesn’t run the protocol directly)

Implementation

- GAAP Library - first phase in `python`
- Echo-Sender & Echo-Receiver Test App

```
es <group-name> “<message>”
er <group-name>
```

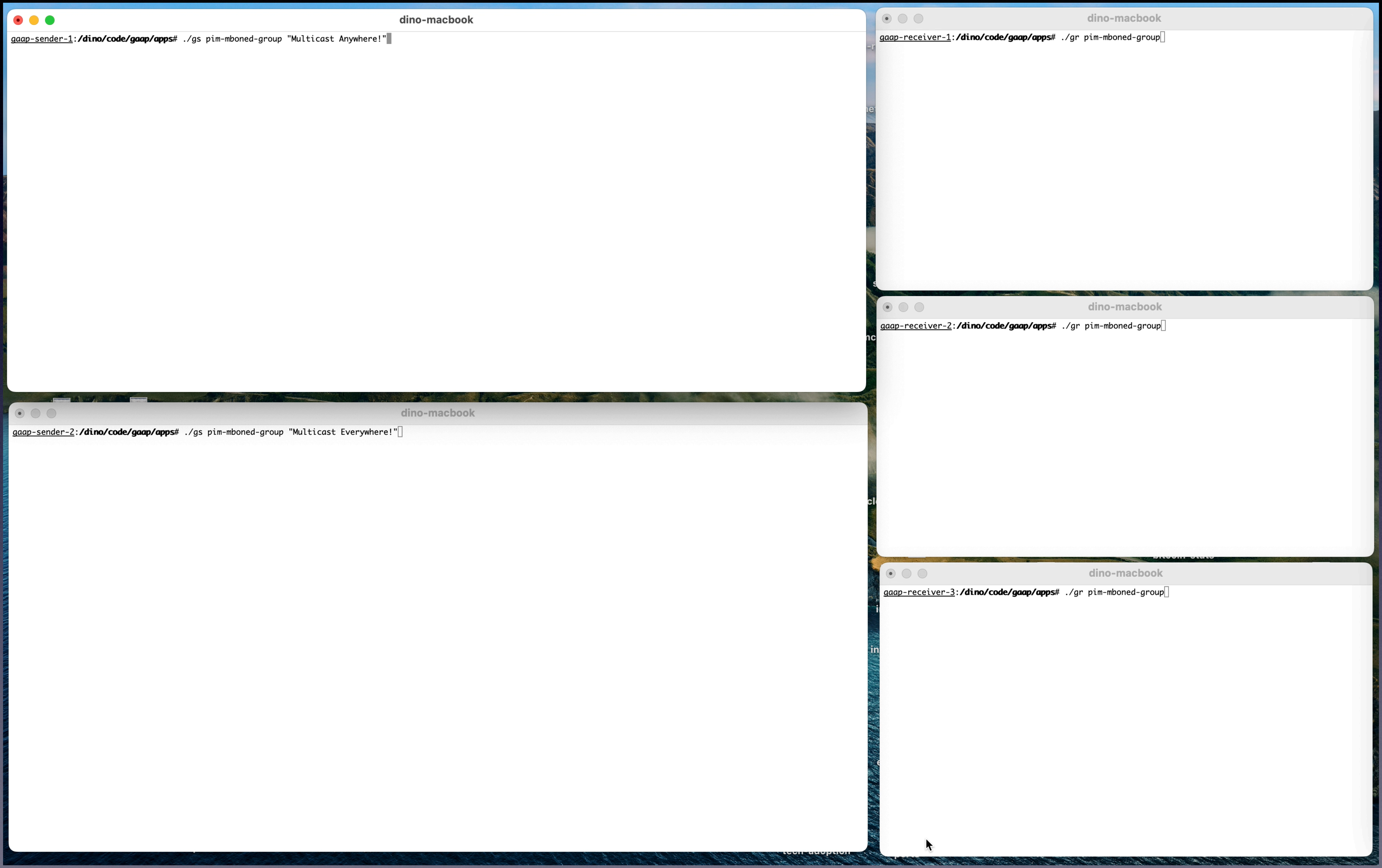
- GAAP Monitor Tool

```
gaapshark [<group-name>]
```

- Suite of GAAP Utilities

```
gaaphash, gaapscale, gaapcollide
```

App Demo - High Level



```
gaap-sender-1:/dino/code/gaap/apps# ./gs pim-mboned-group "Multicast Anywhere!"
```

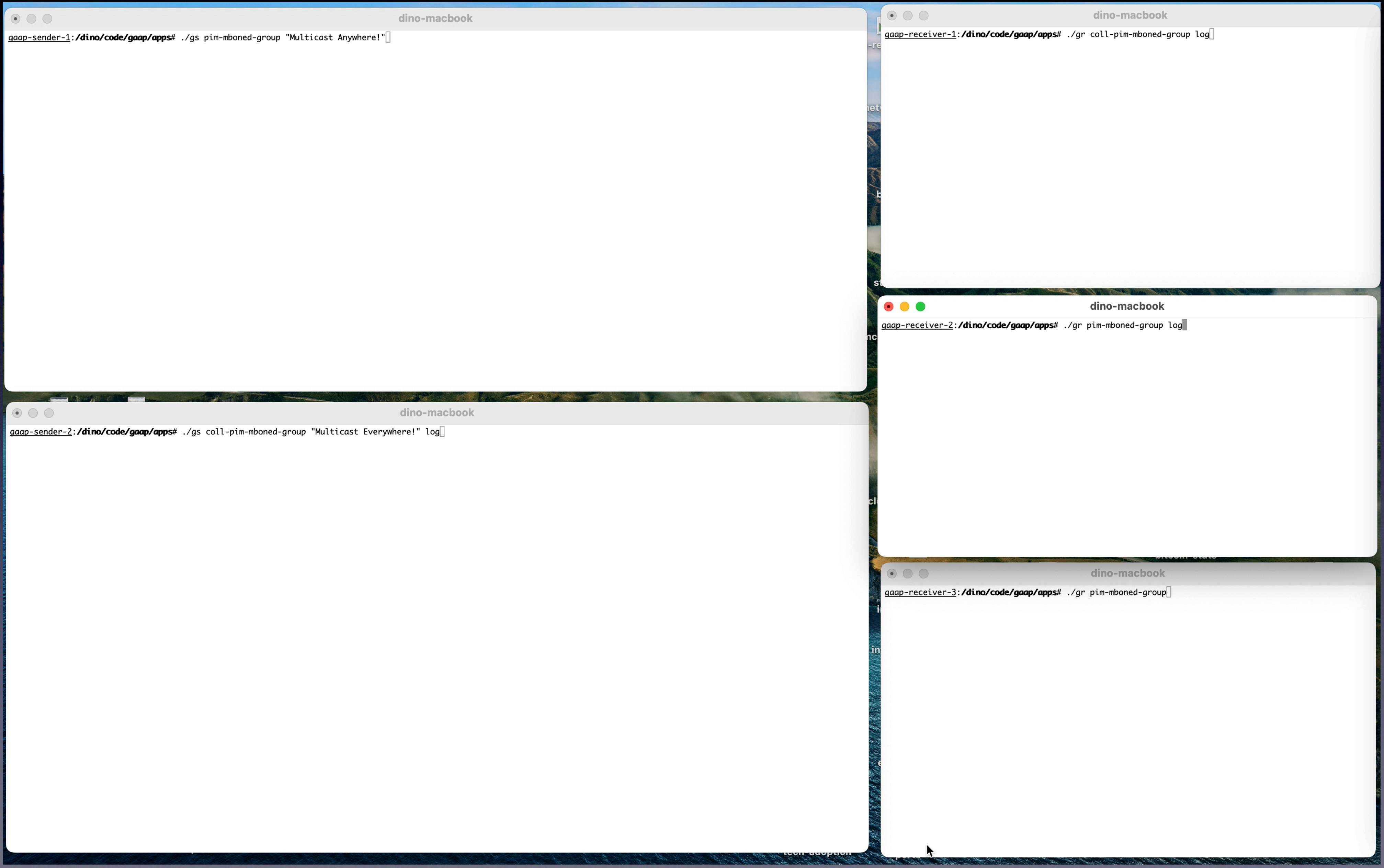
```
gaap-sender-2:/dino/code/gaap/apps# ./gs pim-mboned-group "Multicast Everywhere!"
```

```
gaap-receiver-1:/dino/code/gaap/apps# ./gr pim-mboned-group
```

```
gaap-receiver-2:/dino/code/gaap/apps# ./gr pim-mboned-group
```

```
gaap-receiver-3:/dino/code/gaap/apps# ./gr pim-mboned-group
```

Protocol Demo - Details



```
gaap-sender-1:/dino/code/gaap/apps# ./gs pim-mboned-group "Multicast Anywhere!"
```

```
gaap-receiver-1:/dino/code/gaap/apps# ./gr coll-pim-mboned-group log
```

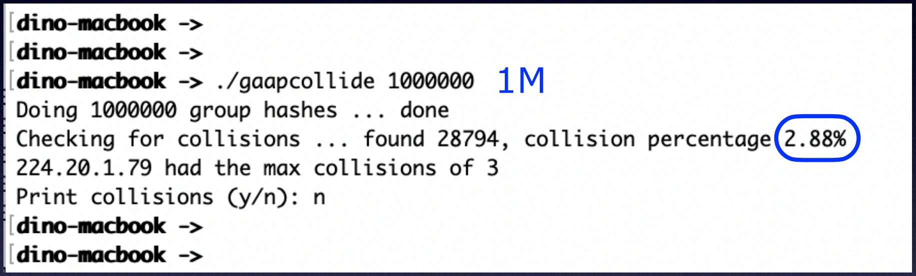
```
gaap-receiver-2:/dino/code/gaap/apps# ./gr pim-mboned-group log
```

```
gaap-sender-2:/dino/code/gaap/apps# ./gs coll-pim-mboned-group "Multicast Everywhere!" log
```

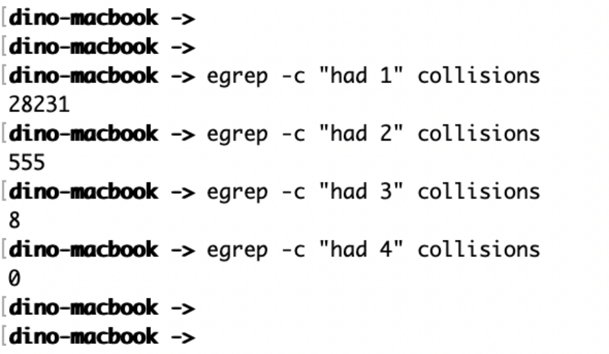
```
gaap-receiver-3:/dino/code/gaap/apps# ./gr pim-mboned-group
```

Quick Point about Collisions

- Collisions are very rare
- There is a good chance that the Collision Claim Procedures will never run

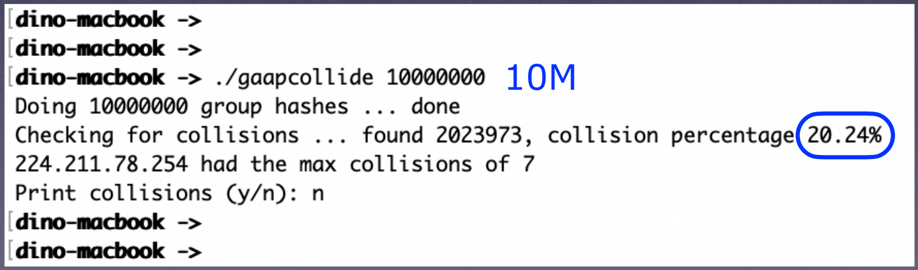


```
dino-macbook ->
dino-macbook ->
dino-macbook -> ./gaapcollide 1000000  1M
Doing 1000000 group hashes ... done
Checking for collisions ... found 28794, collision percentage 2.88%
224.20.1.79 had the max collisions of 3
Print collisions (y/n): n
dino-macbook ->
dino-macbook ->
```

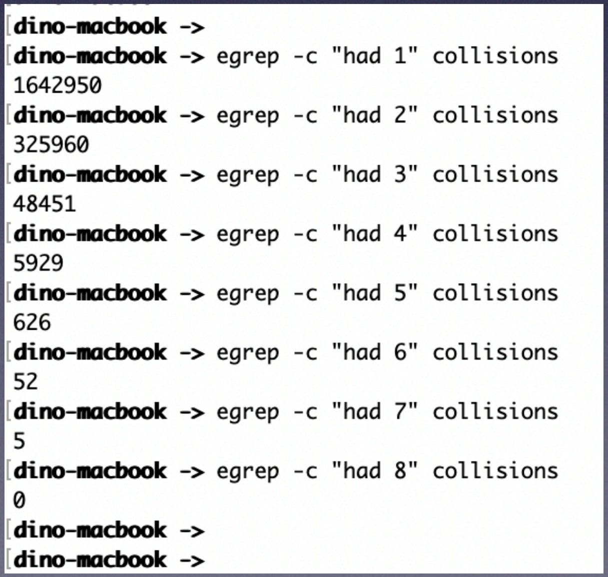


```
dino-macbook ->
dino-macbook ->
dino-macbook -> egrep -c "had 1" collisions
28231
dino-macbook -> egrep -c "had 2" collisions
555
dino-macbook -> egrep -c "had 3" collisions
8
dino-macbook -> egrep -c "had 4" collisions
0
dino-macbook ->
dino-macbook ->
```

These simulations ran with consecutive group-names



```
dino-macbook ->
dino-macbook ->
dino-macbook -> ./gaapcollide 10000000  10M
Doing 10000000 group hashes ... done
Checking for collisions ... found 2023973, collision percentage 20.24%
224.211.78.254 had the max collisions of 7
Print collisions (y/n): n
dino-macbook ->
dino-macbook ->
```



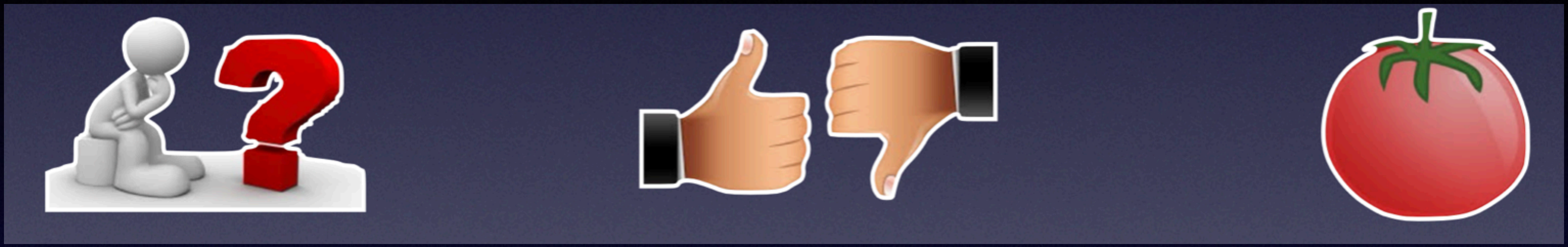
```
dino-macbook ->
dino-macbook -> egrep -c "had 1" collisions
1642950
dino-macbook -> egrep -c "had 2" collisions
325960
dino-macbook -> egrep -c "had 3" collisions
48451
dino-macbook -> egrep -c "had 4" collisions
5929
dino-macbook -> egrep -c "had 5" collisions
626
dino-macbook -> egrep -c "had 6" collisions
52
dino-macbook -> egrep -c "had 7" collisions
5
dino-macbook -> egrep -c "had 8" collisions
0
dino-macbook ->
dino-macbook ->
```

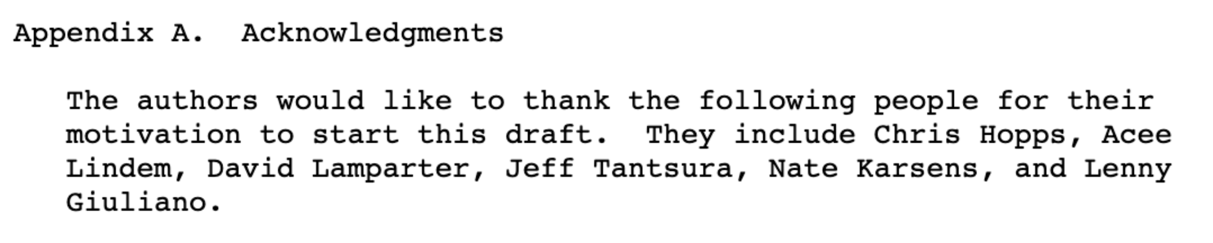
Using sha256() hash function to produce unique 24-bits

Next Steps

- More testing
- Add more security features (Shamir’s MPC Algorithm)
- Test on an overlay when no native multicast exists
- Code lightweight Restful API library
- Write more apps, suggestions?
- Seek more app developers

Questions/Reactions/Tomatoes?





```
Appendix A.  Acknowledgments

   The authors would like to thank the following people for their
   motivation to start this draft.  They include Chris Hopps, Acee
   Lindem, David Lamparter, Jeff Tantsura, Nate Karsens, and Lenny
   Giuliano.
```

App Demo - High Level

Protocol Demo - Details