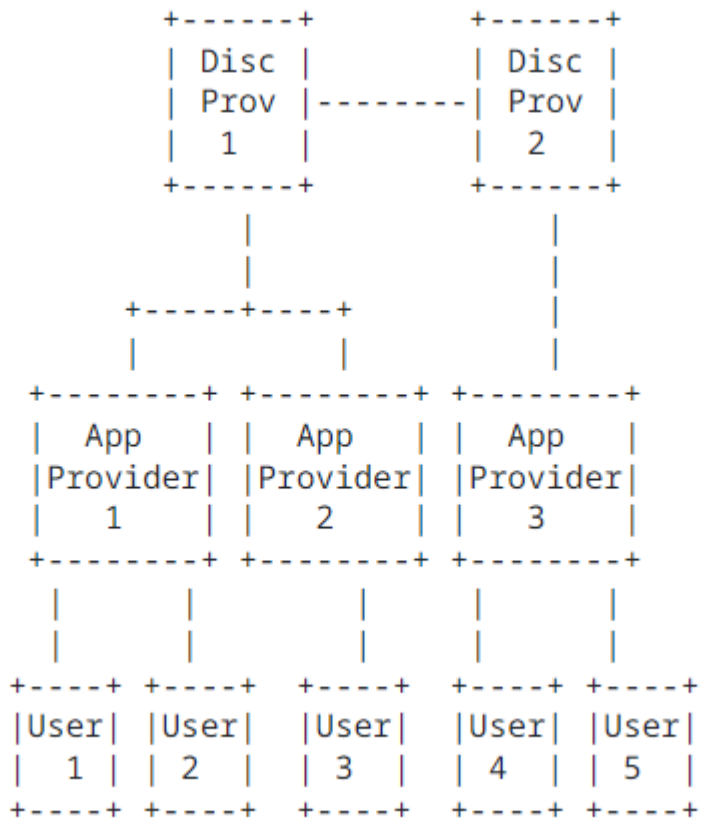
Global Lookup and Discovery of Services (GLADOS)

Jonathan Rosenberg, Five9

Requirements and Assumptions

- We need multiple mapping resolvers, to handle geopolitical and organizational boundaries
- The resolvers are trusted, and geopolitical and/or organizational processes would be put in place allowing that trust to be earned
- It is acceptable for the resolver to know the identifier (phone number or email) for a user and the service they are on, but it is not acceptable for it to know the social graph of interconnectedness between end users
- The resolver function should be invisible to end consumers – consumers should see the apps they are already using and not something new
- App providers will include gatekeepers and also small, potentially numerous, providers
- We want to protect against a malicious app provider that requests interop but uses it to send spam or scrape the system for a database of numbers
- Part of the role of the resolver is to validate the good behavior of the app provider – vetting them as a legitimate messaging app and rate limiting inquiries
- The system will not work without a solution to the bootstrap problem

Three Entities – Users, App Providers, Discovery Providers



Glados Identifier Concepts

Service Independent Identifier (SII)

+1 (732) 654-3210

joe@yahoo.com

Domain part is not their messaging provider!!

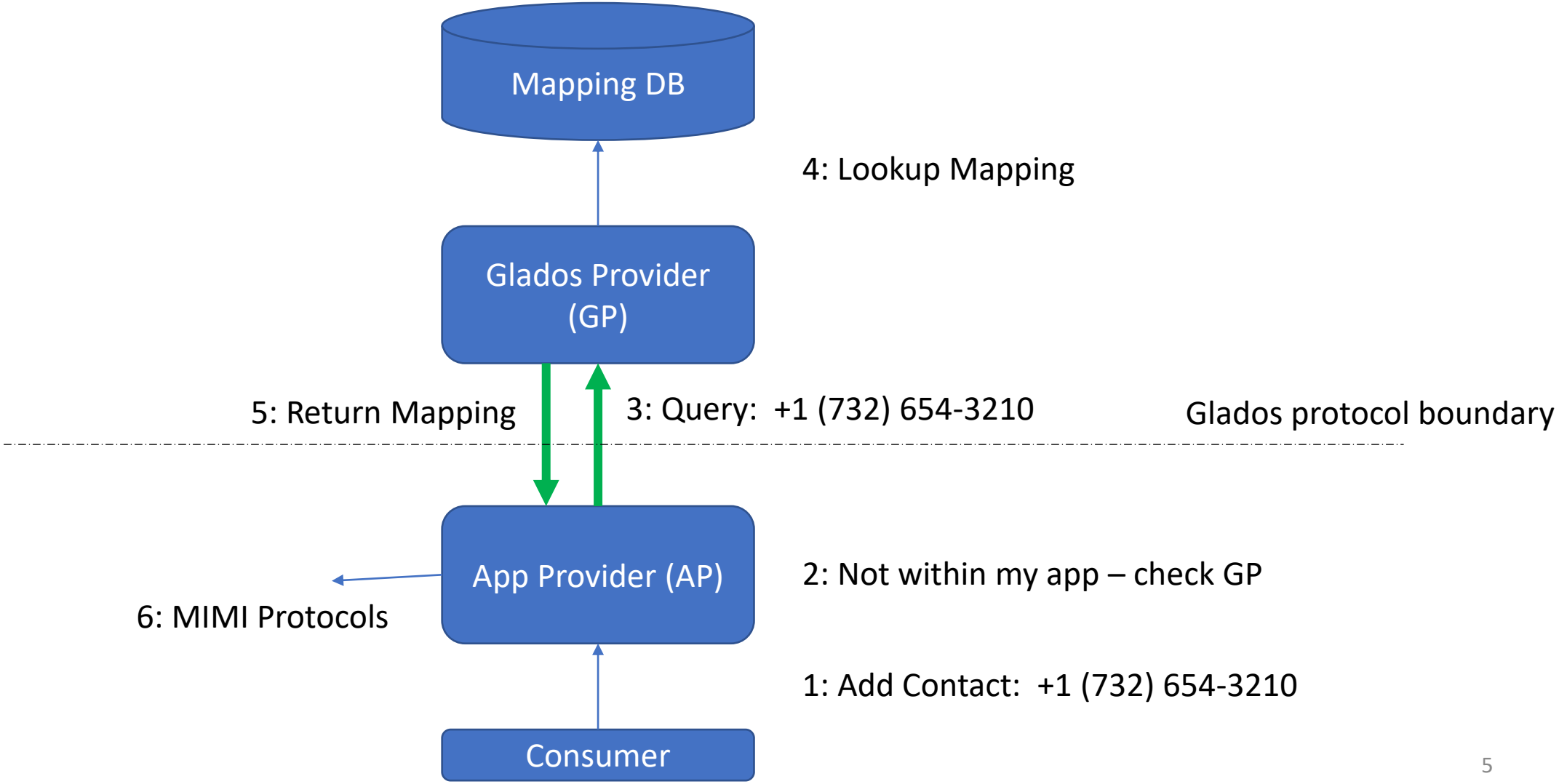
Service Specific Identifier (SSI)

{user2328371, whatsapp.com}

{user123663523, imessage.com}

User part is scoped to the provider

GLADOS Architecture – Single GP



Glados Mapping Creation: Untrusted AP

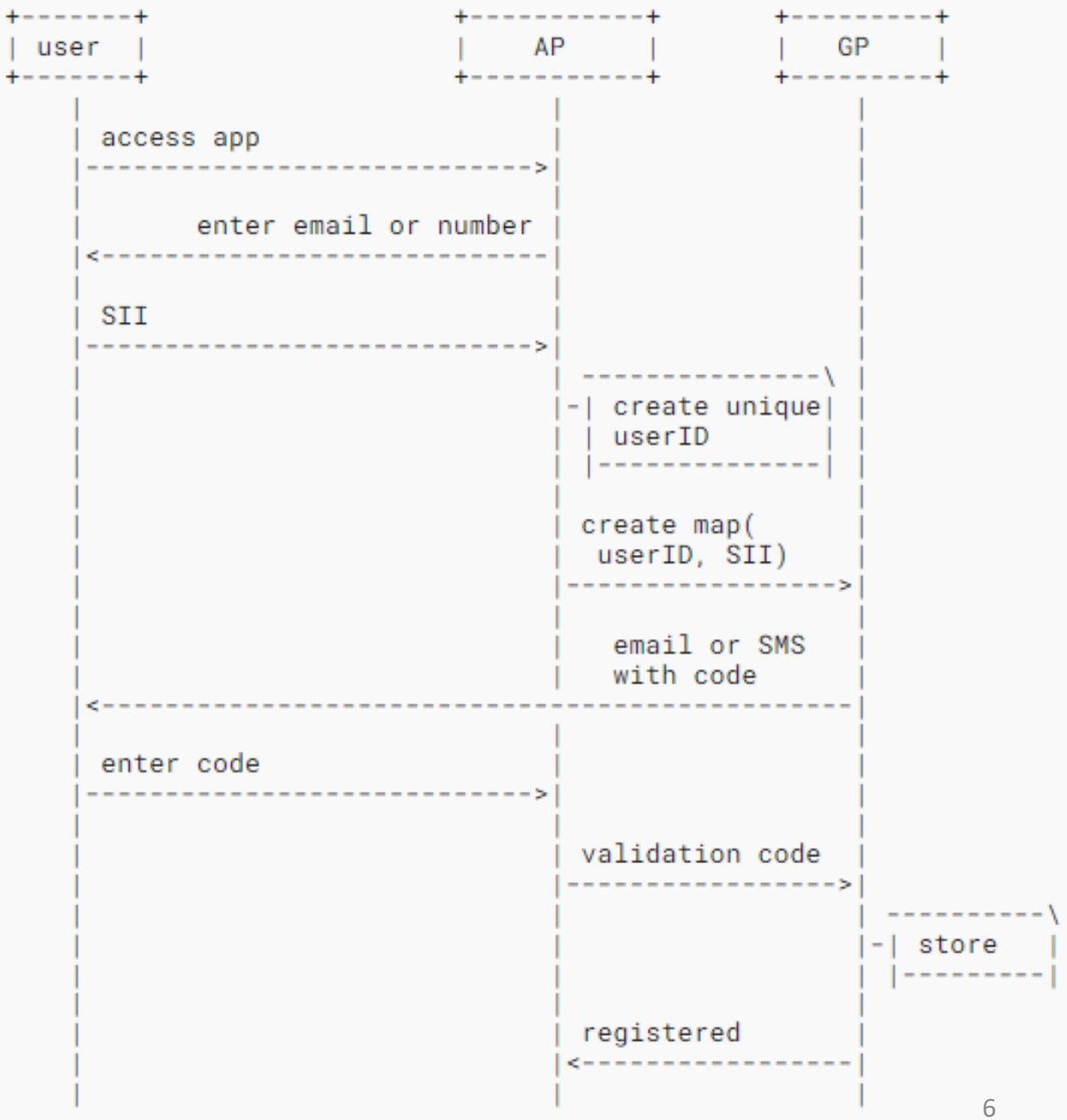
GP is trusted in its accuracy of mapping

AP is (mostly) NOT trusted to provide valid mappings

GP performs the validation on its own

End user experience unchanged from today

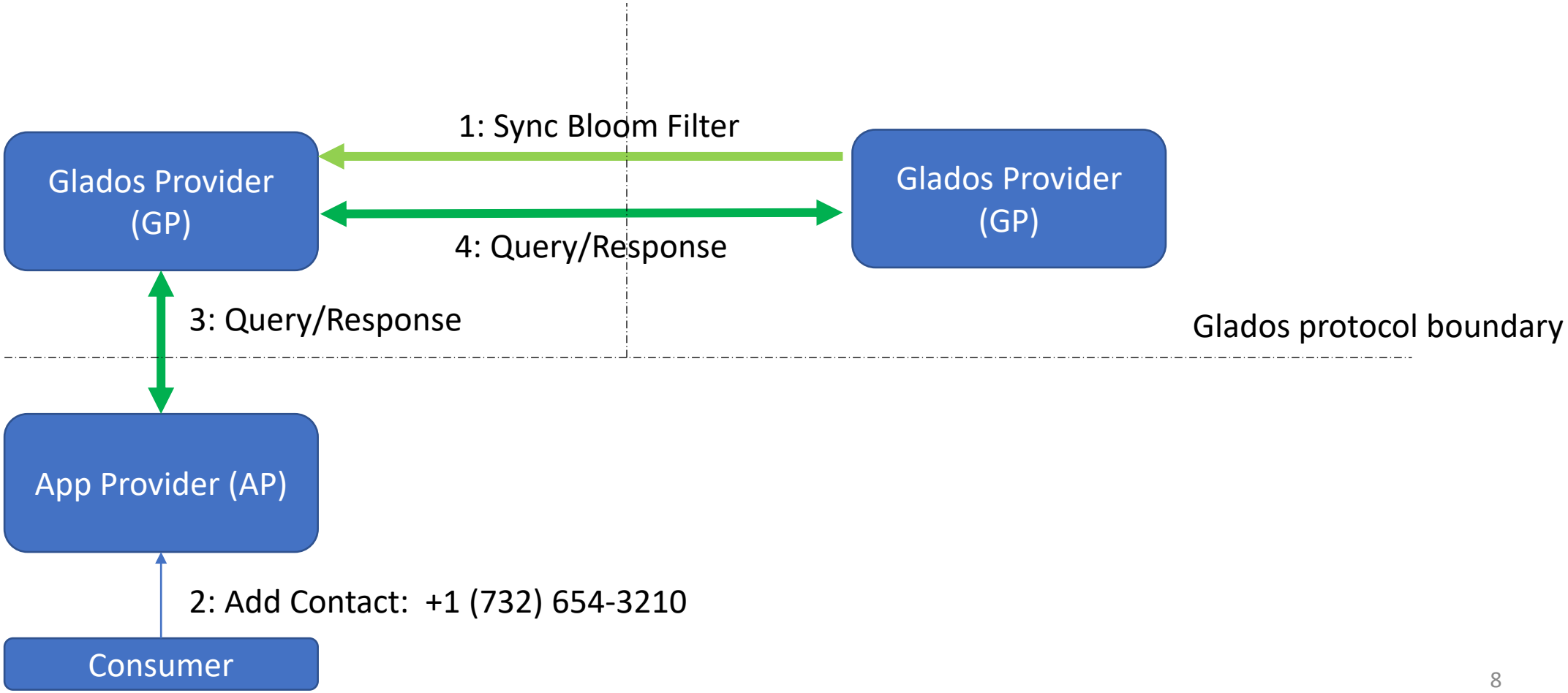
PROBLEM: How to bootstrap it?



Solution: Special Case Exceptions

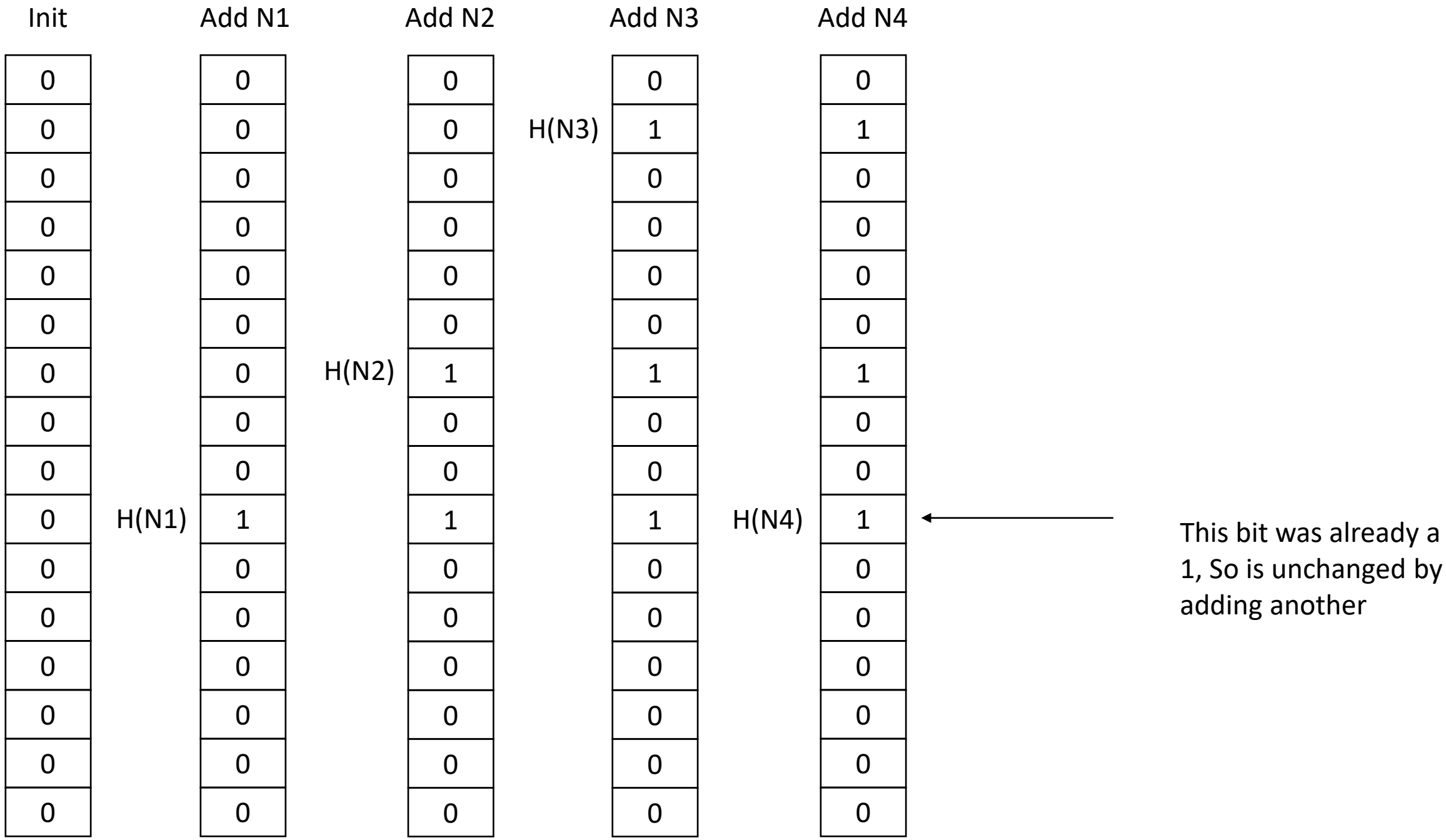
- Case 1: The Discovery Provider is the same as the Application Provider, and the Application Provider is large, and trusted to have performed valid number and email validations
  - Possibly WhatsApp, Apple iMessage
- Case 2: The Discovery Provider is already holding mappings for existing phone number routing databases
  - GSMA Pathfinder by Neustar
- Case 3: The Discovery Provider is distinct from the AP, and does a one-time import of mappings from large, trusted APs

GLADOS Architecture – Multiple GP

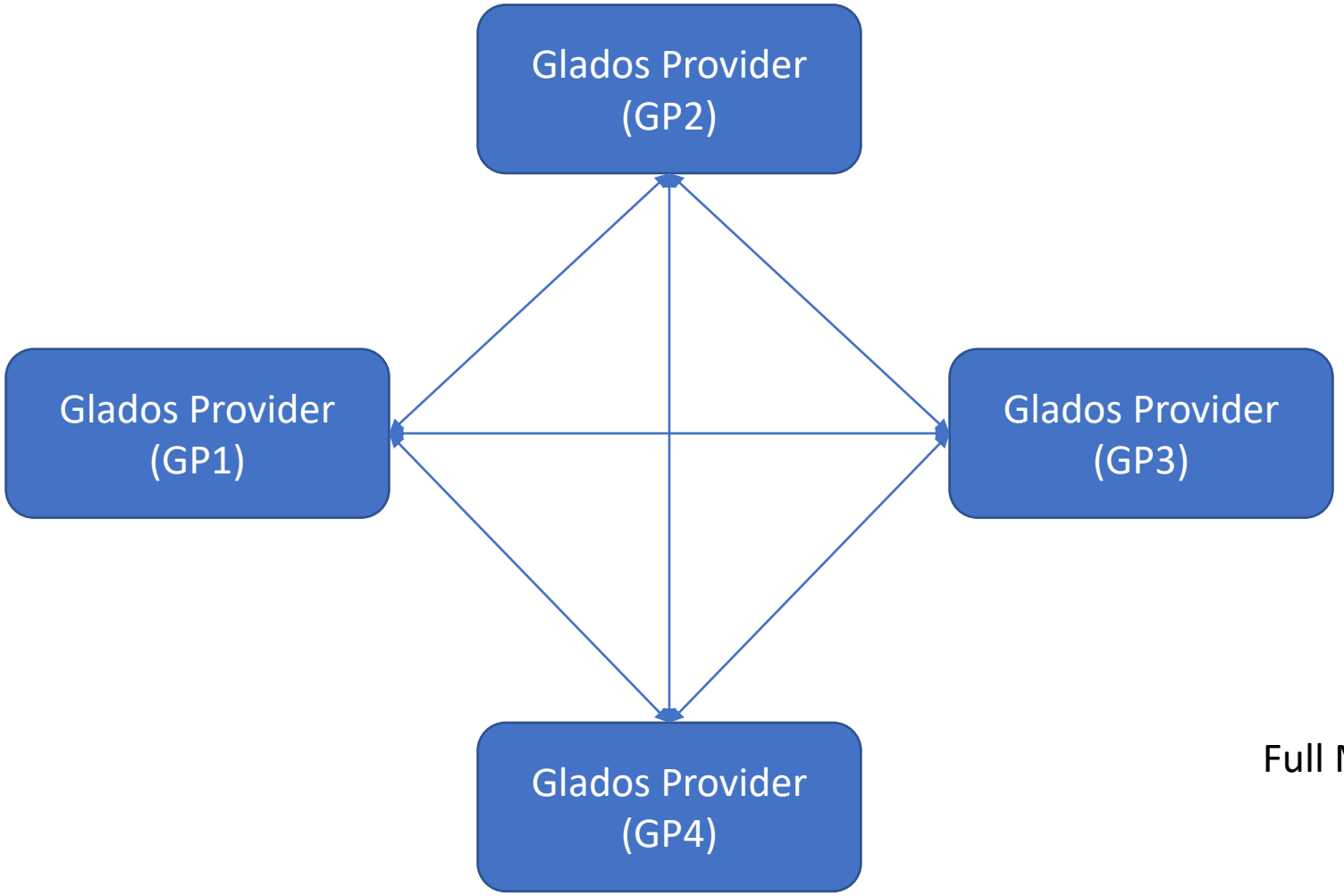


GP to GP Routing: Bloom Filters

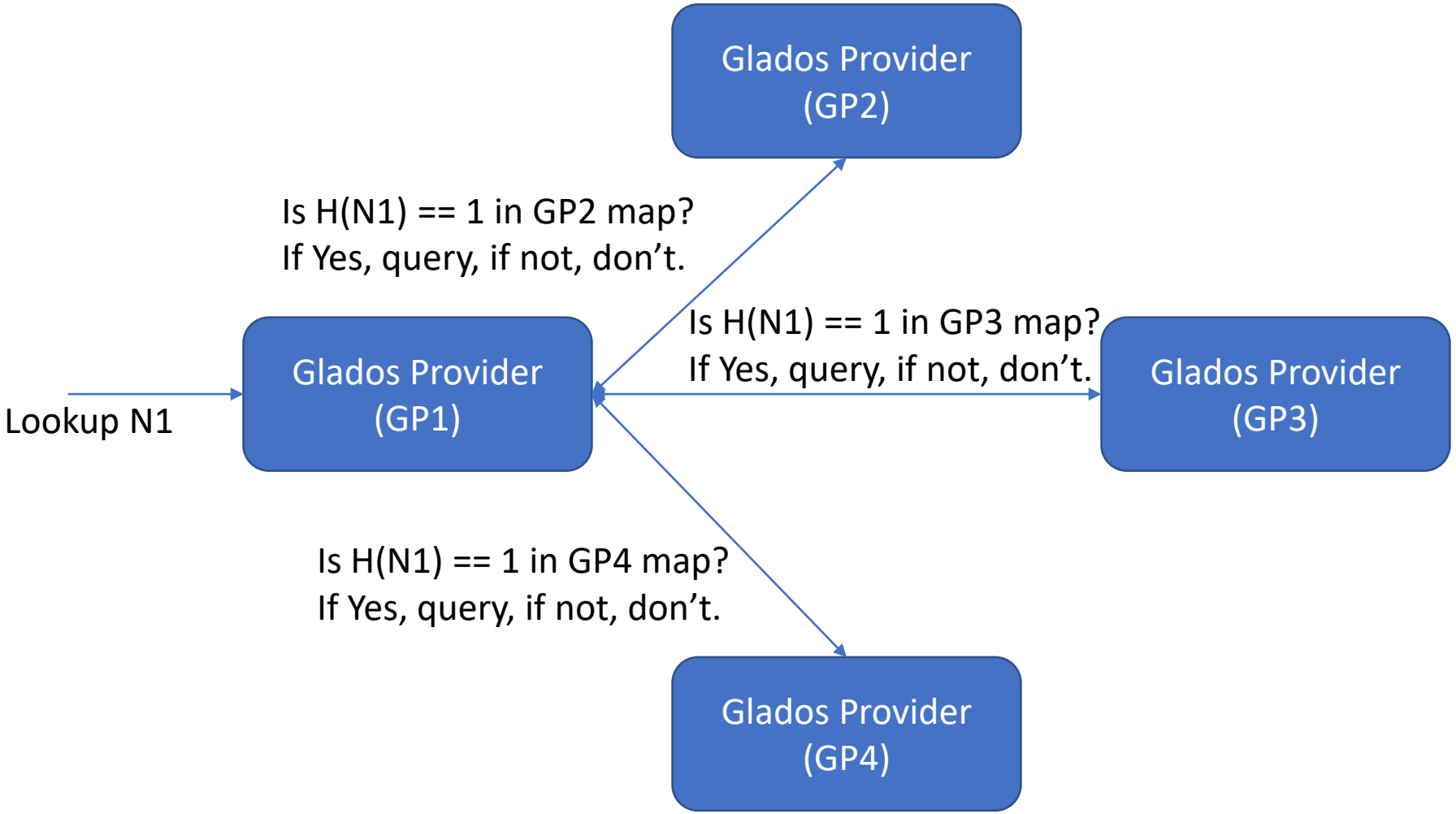
- Each GP periodically builds a bloom filter of size 2^N, initialized to zero
- Each SII is canonicalized, hashed, modulus 2^N and corresponding bit in bloom filtered is OR'd with a 1
- GP exposes API to retrieve bloom filter
- GP exposes API for additions (not removals) from bloom filter to be streamed – allowing instant updates (key use case)
- Bloom filter protocol has many benefits
  - Privacy protecting
  - Compact


Init
Add N1
Add N2
Add N3
Add N4
H(N1)
H(N2)
H(N3)
H(N4)
This bit was already a 1, So is unchanged by adding another

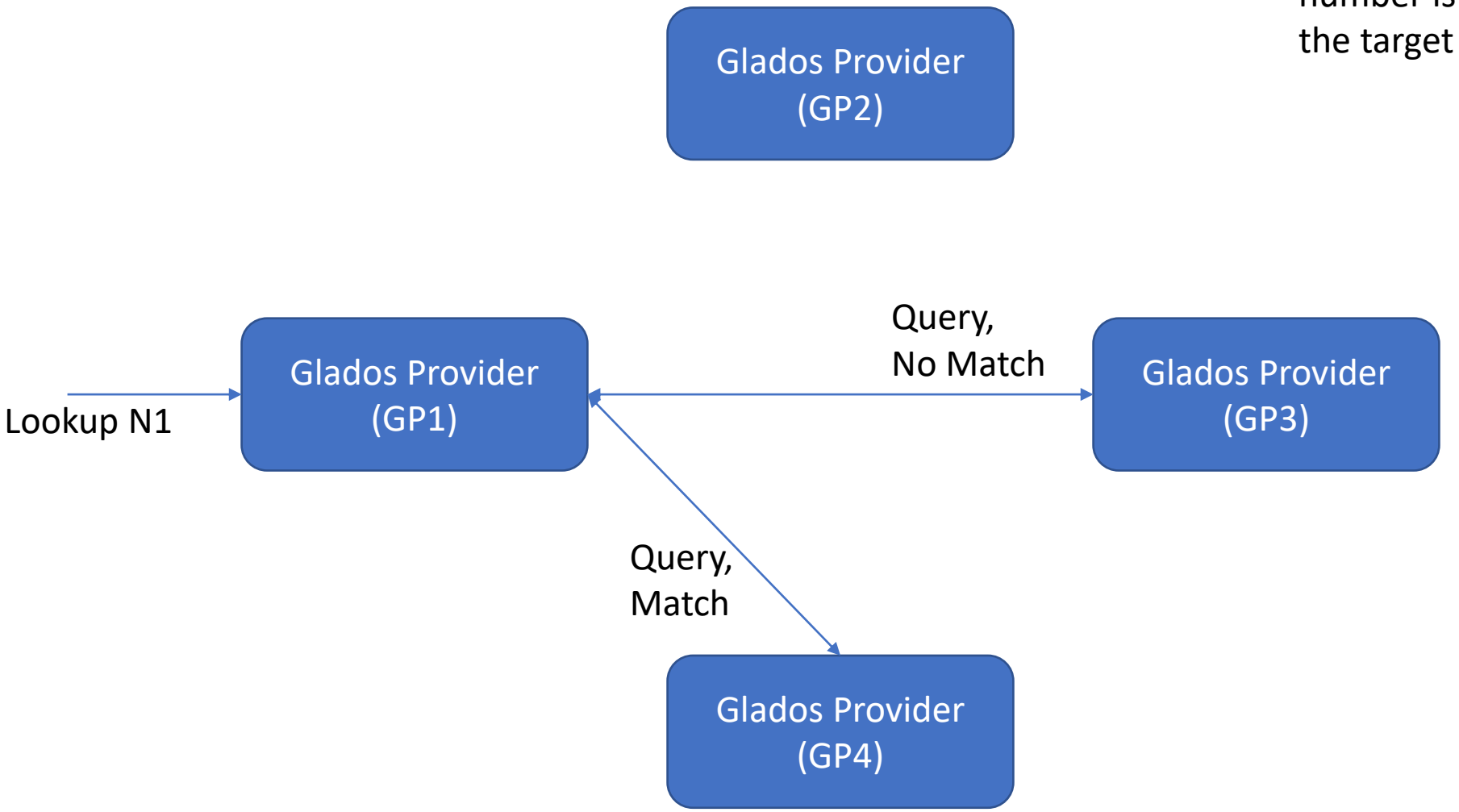
Using the Bloom Filters



Full Mesh Exchanges


Glados Provider
(GP2)
Is H(N1) == 1 in GP2 map?
If Yes, query, if not, don't.
Is H(N1) == 1 in GP3 map?
If Yes, query, if not, don't.
Lookup N1
Glados Provider
(GP1)
Glados Provider
(GP3)
Is H(N1) == 1 in GP4 map?
If Yes, query, if not, don't.
Glados Provider
(GP4)

The Bloom Filter acts as an optimization to reduce query traffic when it is known that the number is definitively NOT at the target GP



Tradeoffs in value of N

- Increased efficacy of filtering, with increasing value of N
- Increased privacy of number ownership, with decreasing value of N
- Increased compression of information, with decreasing value of N

Other Observations on Bloom Filters

- The technique works for any routing against a flat namespace, independent of the type of the identifier – number, email
- The protocol is not tightly bound to MIMI and could be used to facilitate other cases where routing is needed – e.g., in MOQ for finding the caching provider for a named object