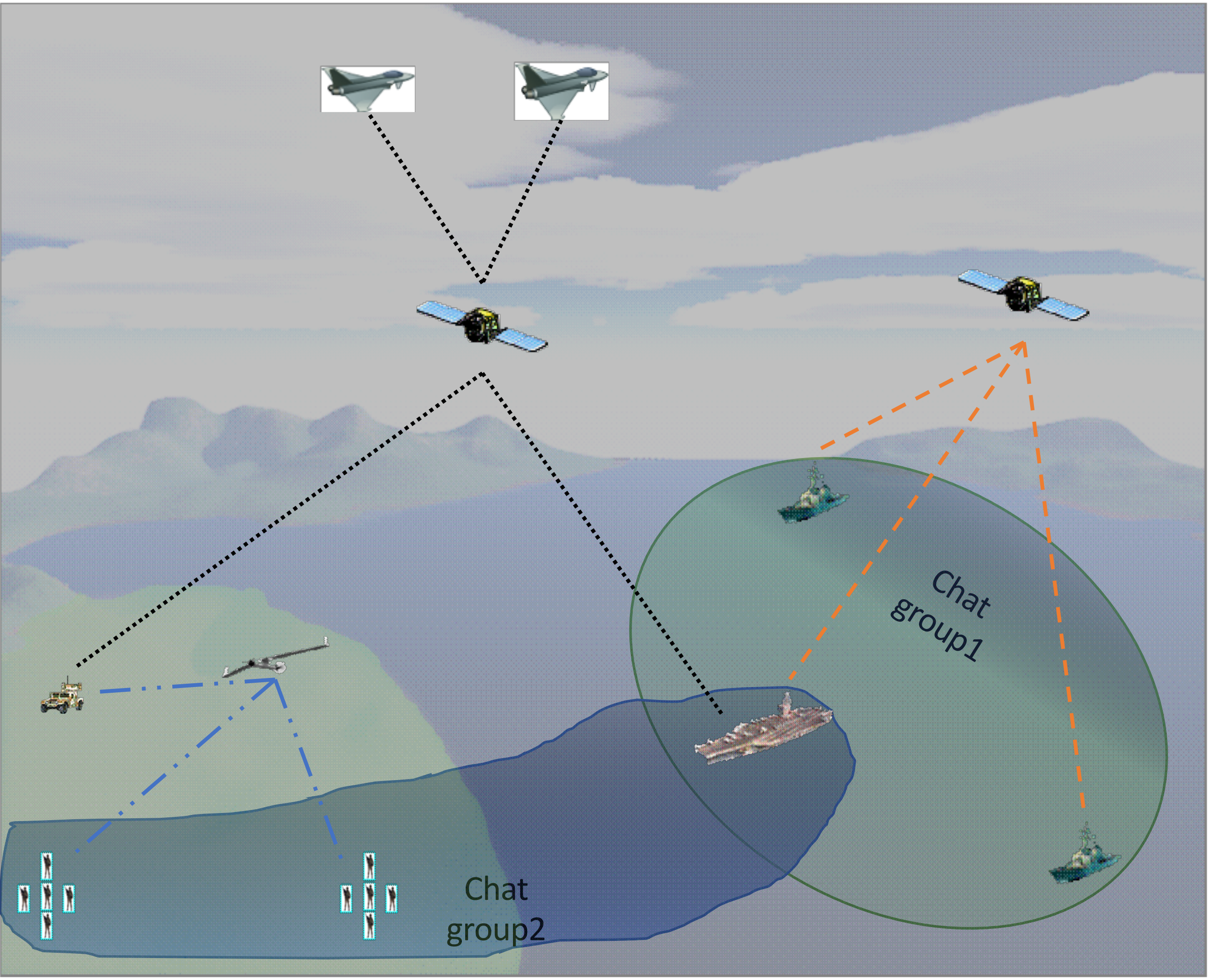
PLI Application

milcom

Military Communications for the 21st Century

LAX Marriott, Los Angeles, CA

Notional Tactical Network

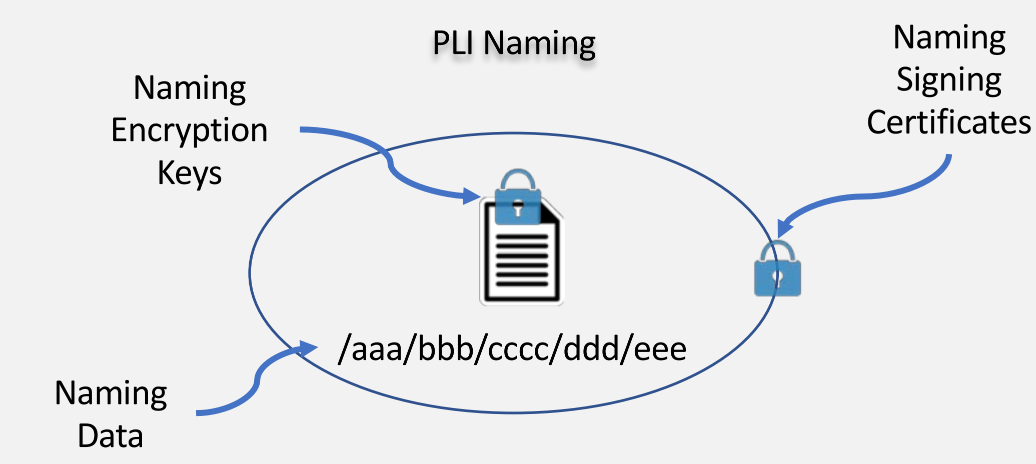


● PLI

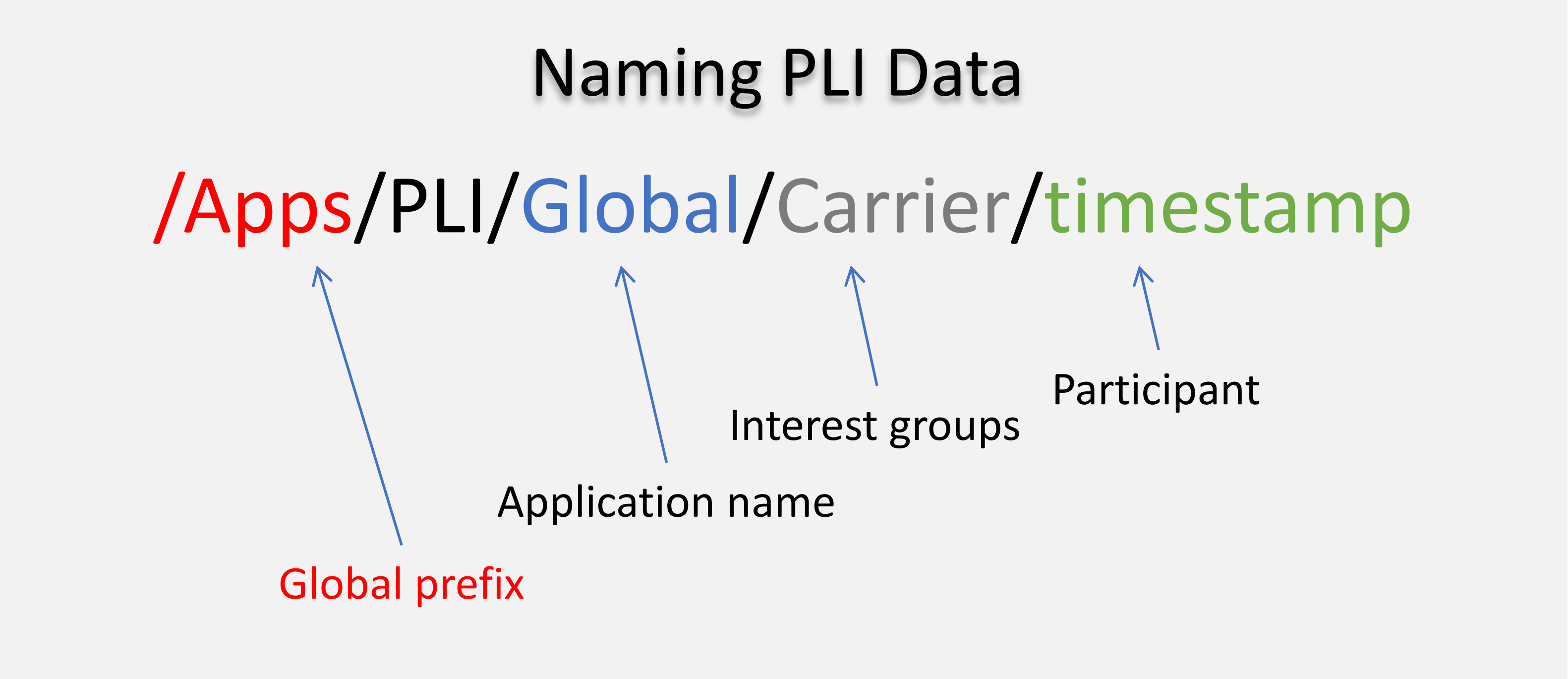
- Shared amongst all nodes. Sent every 5 seconds by each one of the troops, ships, and aircrafts
- Loss can be tolerated
- Delivery of the latest PLI data is most important

PLI Naming

- Naming is part of the application design and configuration
- What to name:
  - PLI data
  - Encryption keys
  - Signing certificates
- Application cares about fetching the data from those who are authorized to participate within a given interest group



Naming PLI Data



Naming PLI Data

/Apps/PLI/Global/Carrier/timestamp

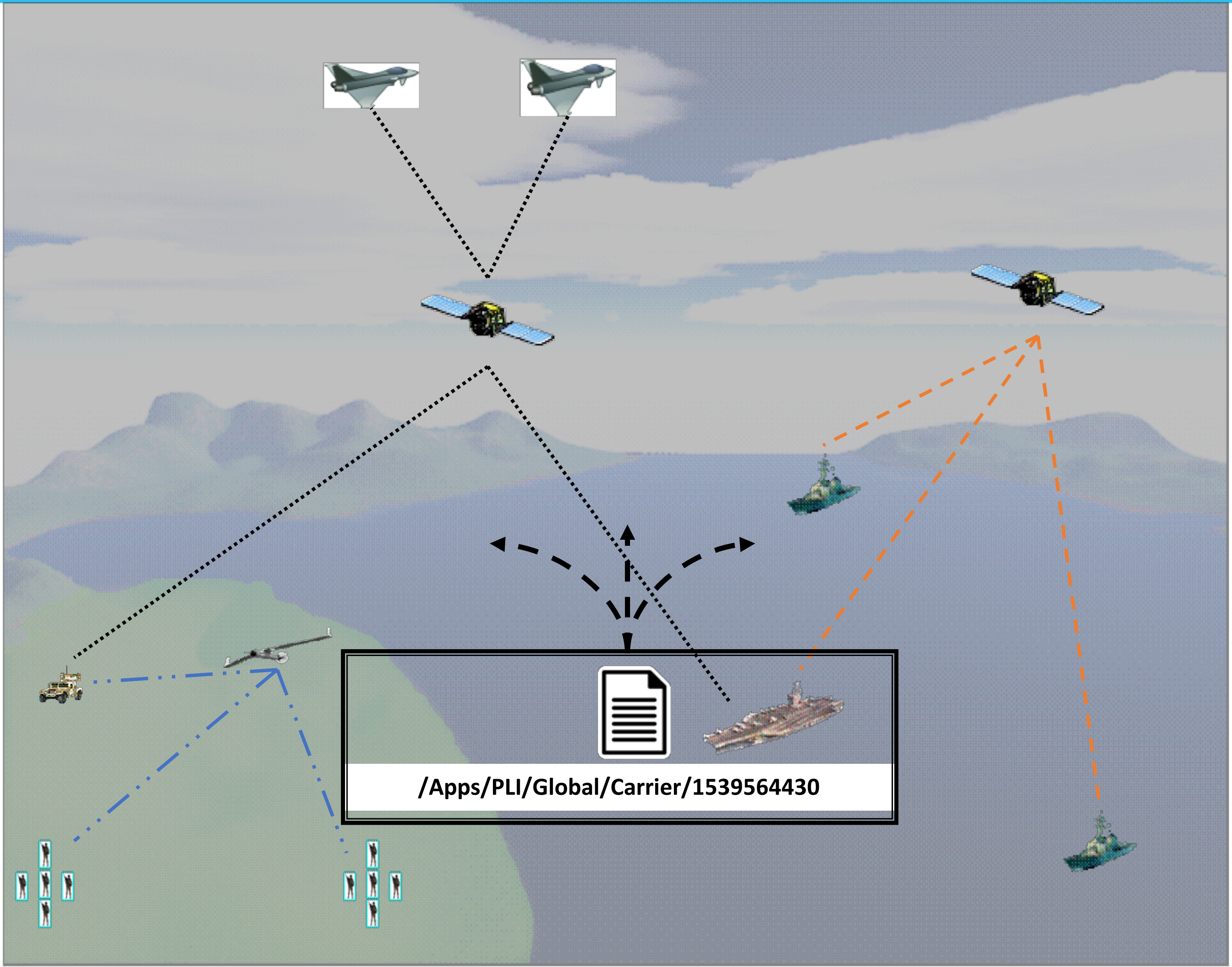
Global prefix

Application name

Interest groups

Participant

Naming PLI Data

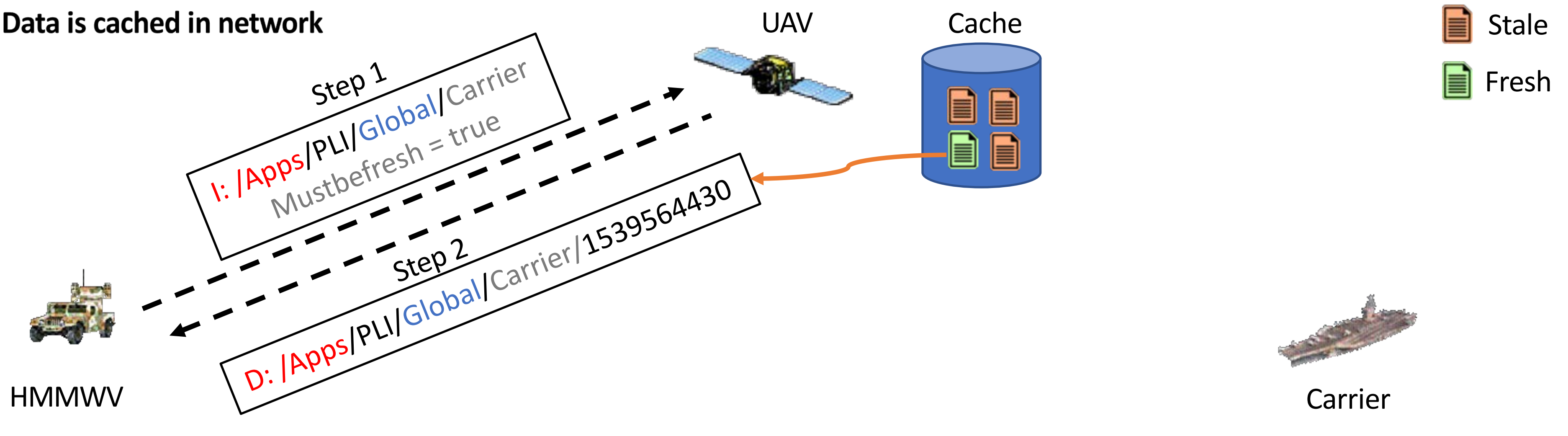


Data Retrieval

- Delivery of the latest data is more important
- Retrieve the latest PLI first
- Loss can be tolerated
- Sync not needed assuming that application is always interested in the latest PLI information
  - PLI data generated periodically with a lifetime that guarantees its expiration when a new PLI data is generated (5 seconds)
  - NDN nodes can choose to retain the latest *n* PLI Data in their caches from any given producer at any point in time (only the latest will be fresh and the rest will be stale)
  - Consumers issue interests with “must be fresh” flag to return the most recent PLI record
  - Interest packets can have the name prefix and not the full name

Data Retrieval

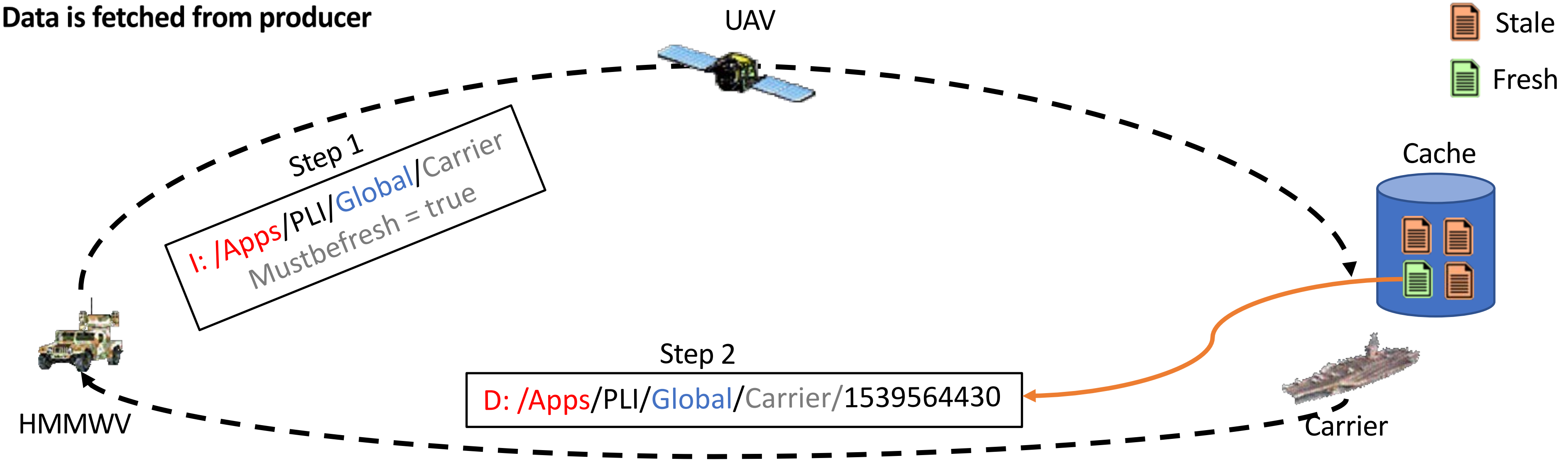
Data is cached in network



Step 1
I: /Apps/PLI/Global/Carrier
Mustbefresh = true

Step 2
D: /Apps/PLI/Global/Carrier/1539564430

Data is fetched from producer



Step 1
I: /Apps/PLI/Global/Carrier
Mustbefresh = true

Step 2
D: /Apps/PLI/Global/Carrier/1539564430

Data Retrieval

- Design tradeoffs
  - What if data generation is event driven as opposed to periodic?
    - May warrant the need for Sync (becomes analogues to chat or CoT)
  - What if we want to retrieve all PLI data?
    - Need to know exact names (can’t use name prefix only)

Retrieve latest PLI Data

I: /Apps/PLI/Global/Carrier
Mustbefresh = true

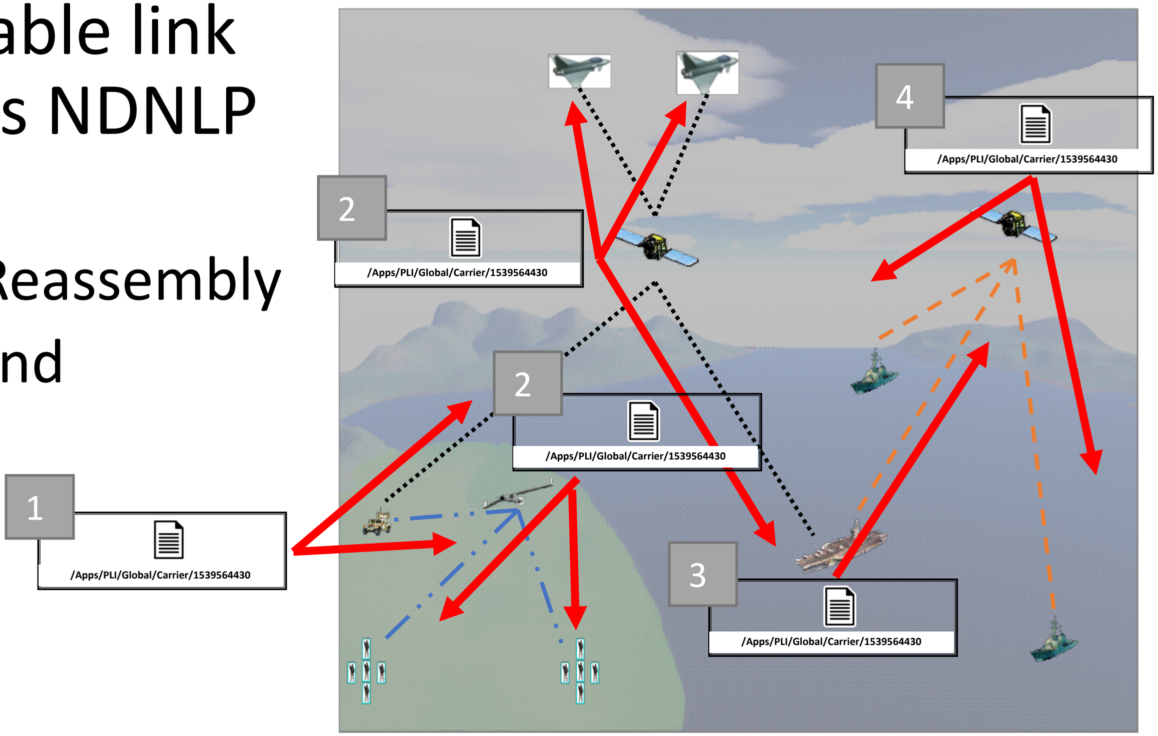
Retrieve a missed PLI Data

I: /Apps/PLI/Global/Carrier/1539564435
Mustbefresh = false

- Need to ensure caching policies retains PLI data for as long as possible

Forwarding and Routing

- Forwarding strategy can be as simple as multicast
  - /Apps/PLI/Global → /localhost/nfd/strategy/multicast
  - Fits well the nature of the application (global sharing)
- To reduce loss, a reliable link layer protocol such as NDNLP may be utilized
  - Fragmentation and Reassembly
  - Acknowledgement and Retransmission


4
/Apps/PLI/Global/Carrier/1539564430
2
/Apps/PLI/Global/Carrier/1539564430
2
/Apps/PLI/Global/Carrier/1539564430
1
/Apps/PLI/Global/Carrier/1539564430
3
/Apps/PLI/Global/Carrier/1539564430

Resilience to Disruption

- Fully utilize the broadcast nature of Wireless channels
- Fully utilize in-network storage
- Fully utilize NDN’s two-way, stateful forwarding plane

Utilizing Wireless Broadcast

- For each device which receives the signal: Does it care?
  - In IP, determined by the address
  - In NDN, determined by the name
- If one cares:
  - Receive an Interest
    - do I have data? Or
    - should I further forward?
  - Receive a data packet
    - Have a matching PIT entry? Or
    - should I buffer it anyway?

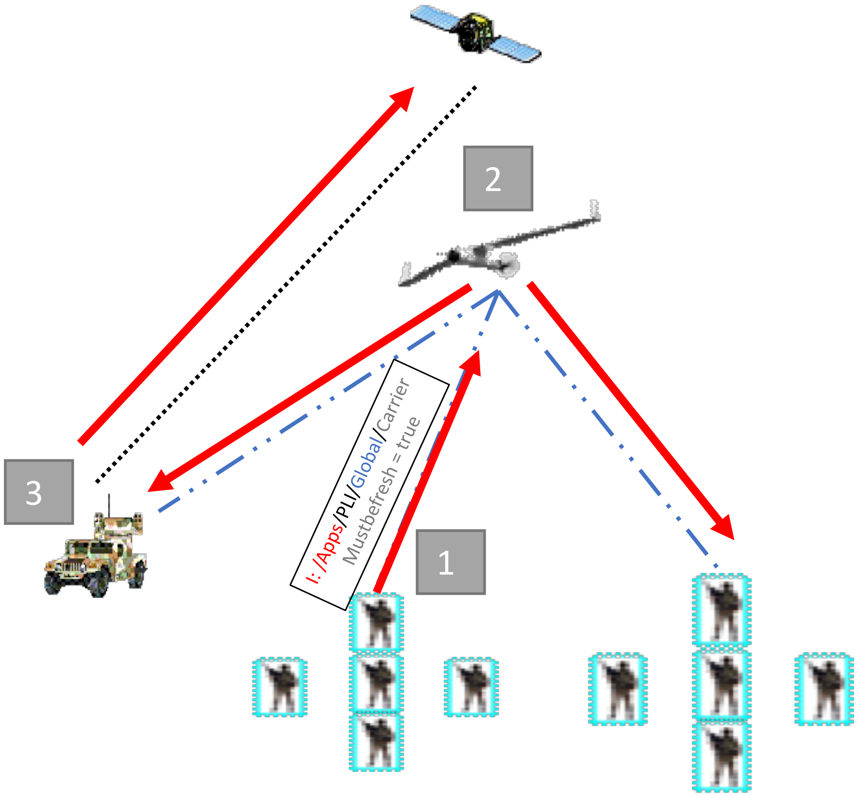
Decision by the forwarding strategy

Utilizing In-Network Storage

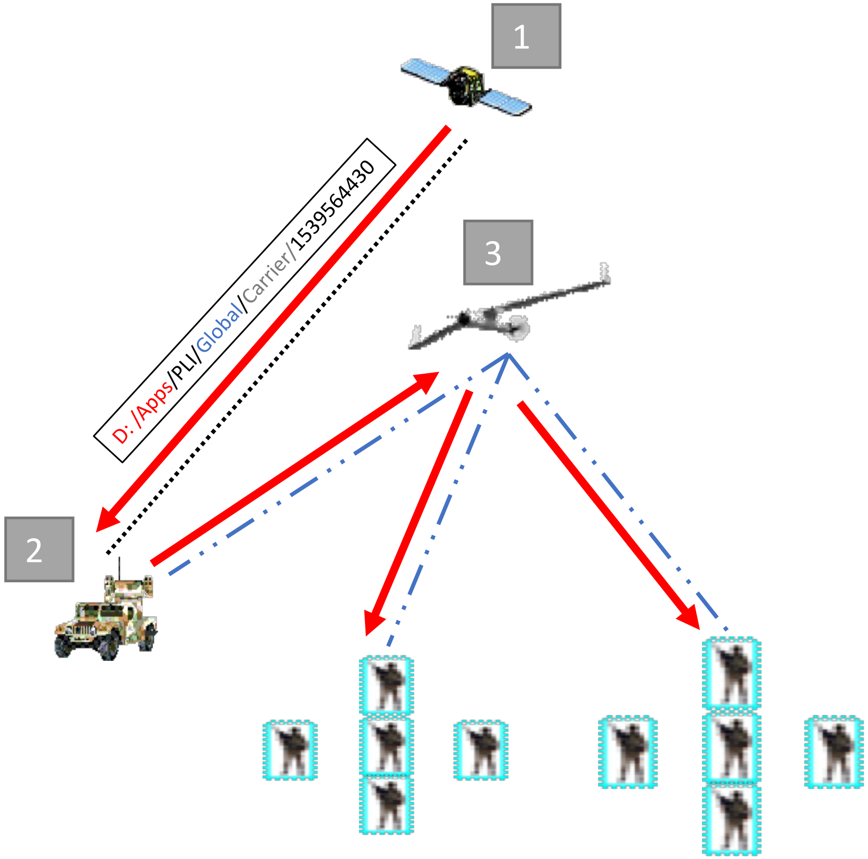
- Receive a data packet but does not have a matching PIT entry at the time
  - May buffer it for future use potential
  - May make the decision based on a filter on name prefixes
- When next time receives an Interest, either from a neighbor node, or from a local app
  - May find matching data in the cache

Utilizing In-Network Storage

- Step 1: Interest sent

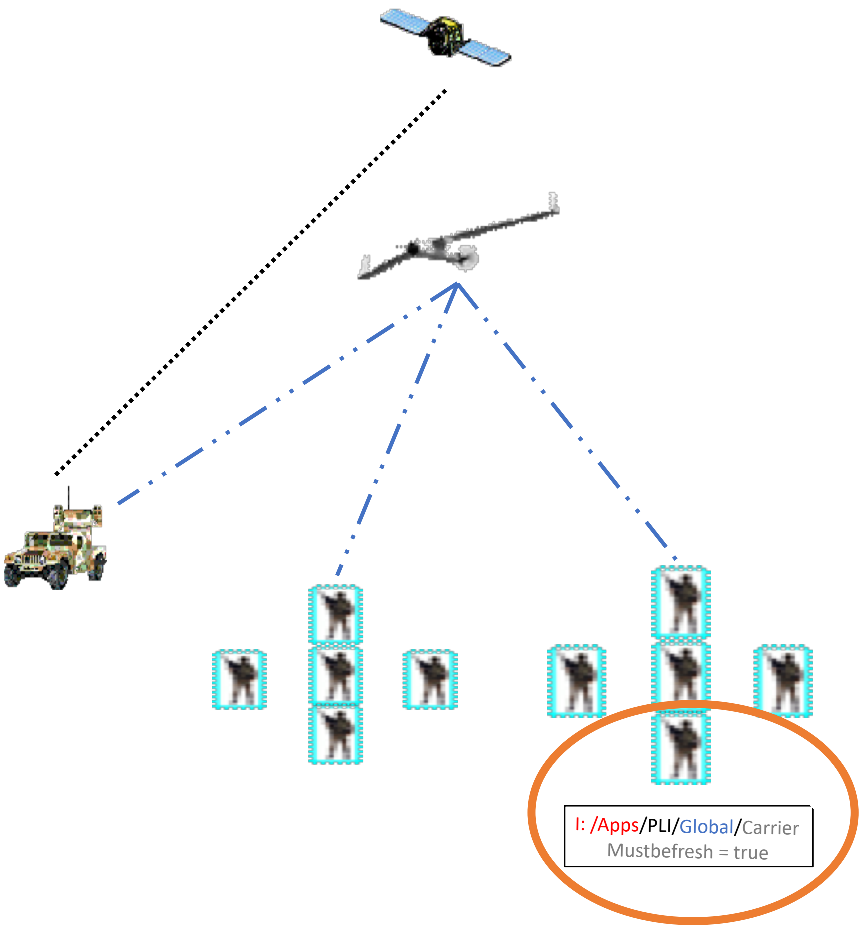


- Step 2: Data retrieved



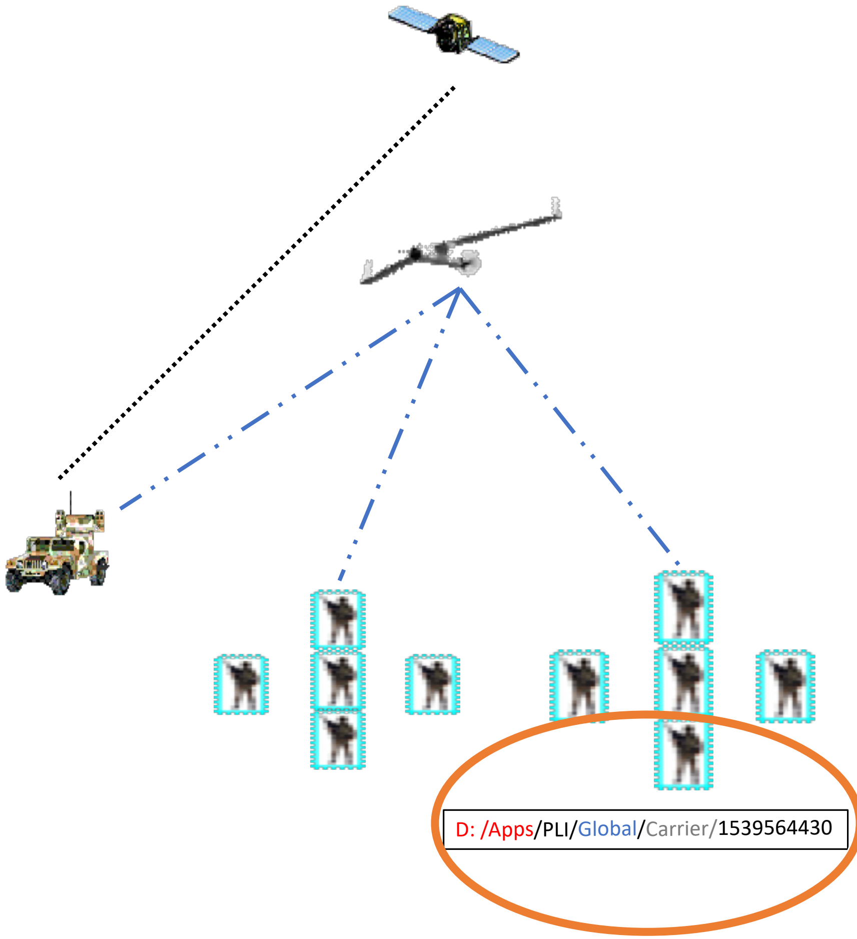
Utilizing In-Network Storage

- Step 3: Interest sent



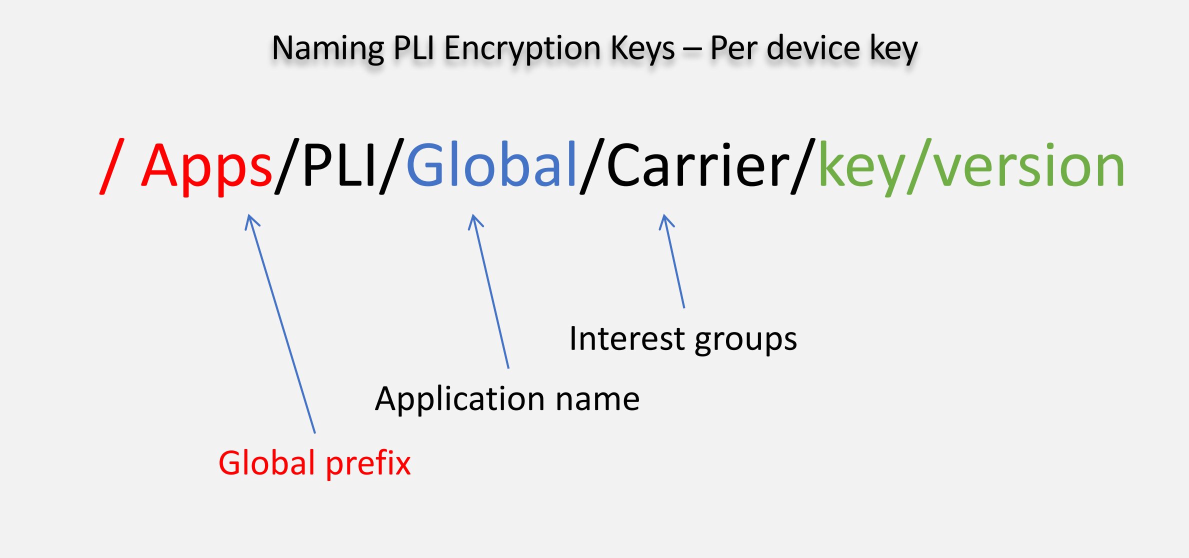
I: /Apps/PLI/Global/Carrier
Mustbefresh = true

- Step 4: Date served locally



D: /Apps/PLI/Global/Carrier/1539564430

Naming PLI Access Keys



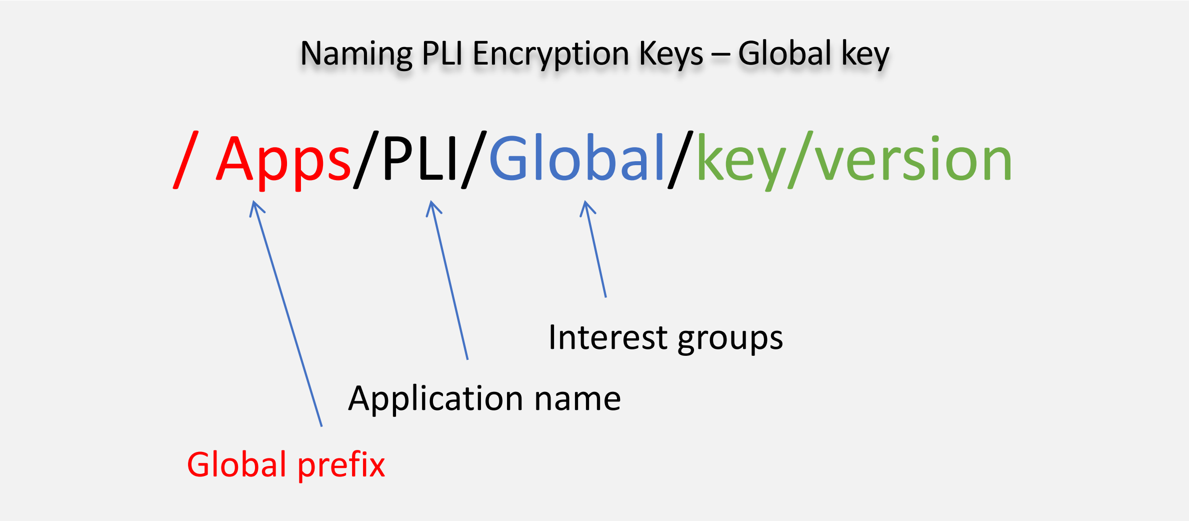
Naming PLI Encryption Keys – Per device key

/ Apps/PLI/Global/Carrier/key/version

Interest groups

Application name

Global prefix



Naming PLI Encryption Keys – Global key

/ Apps/PLI/Global/key/version

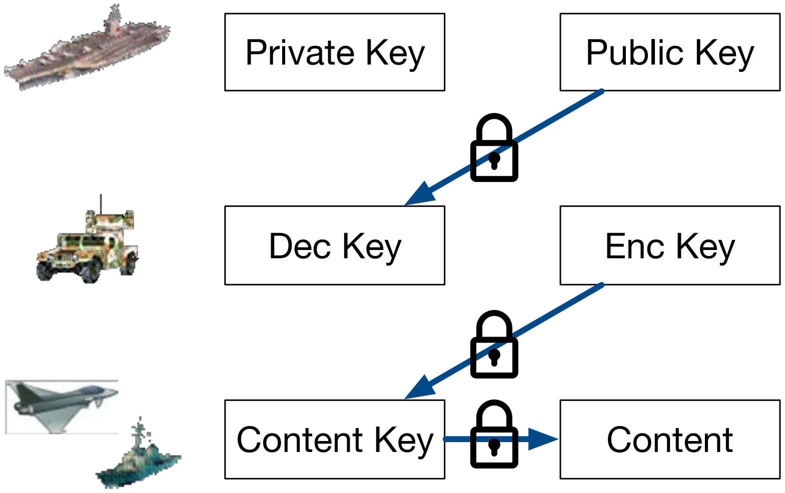
Interest groups

Application name

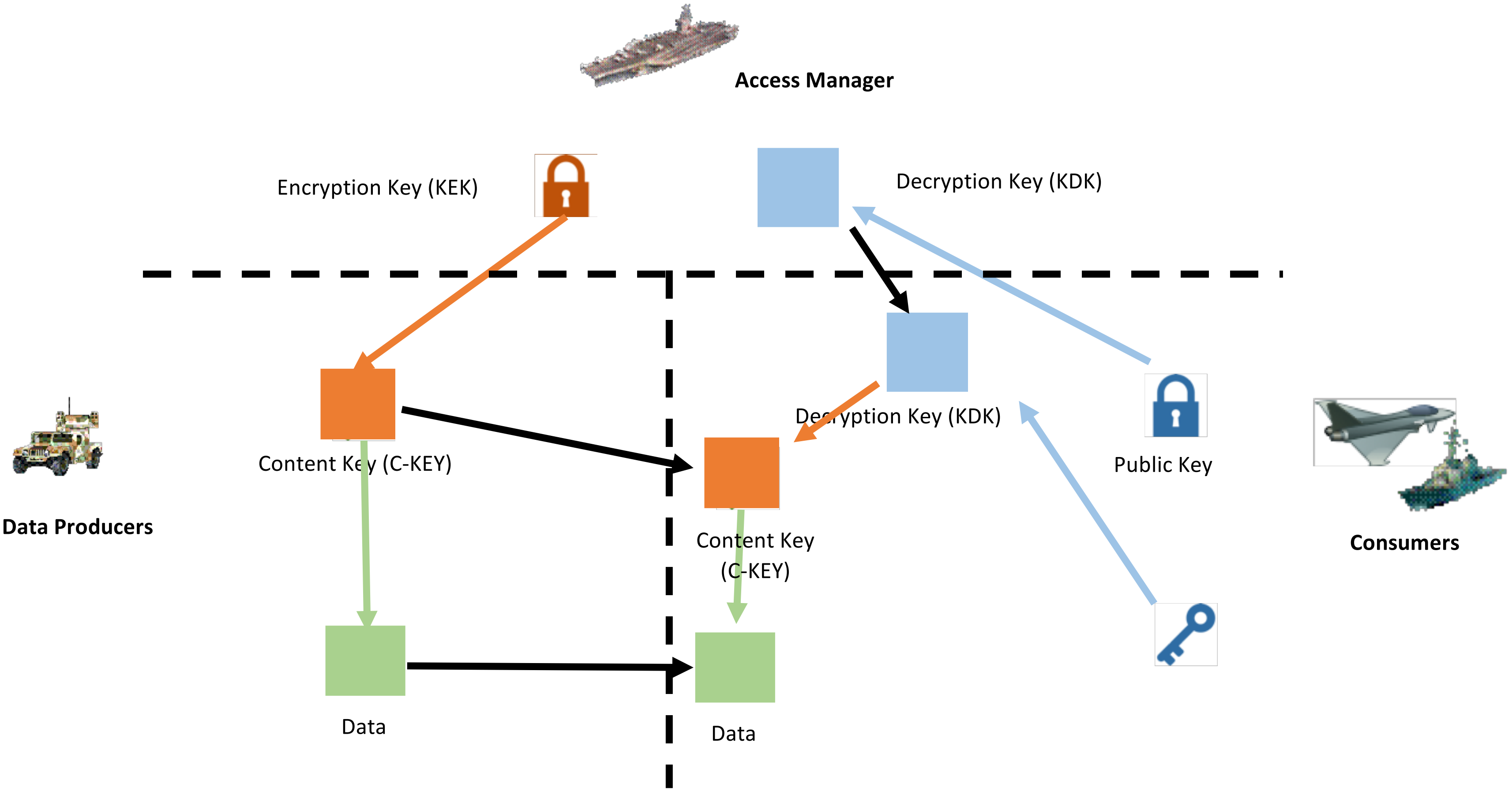
Global prefix

Name-Based Access Control (NAC) Concepts

- Access Controller – Base
  - Creates a list of encryption/decryption key pairs
  - Controls which encryption keys are used to encrypt which namespace
  - Control whom to distribute the corresponding decryption keys
- Producers (Encryptors) – HMMV
  - Fetch the right encryption keys to encrypt data
- Consumers (Decryptor)
  - Fetch the right decryption keys to decrypt data



NAC Process



Access Manager

Encryption Key (KEK)

Decryption Key (KDK)

Data Producers

Content Key (C-KEY)

Decryption Key (KDK)

Public Key

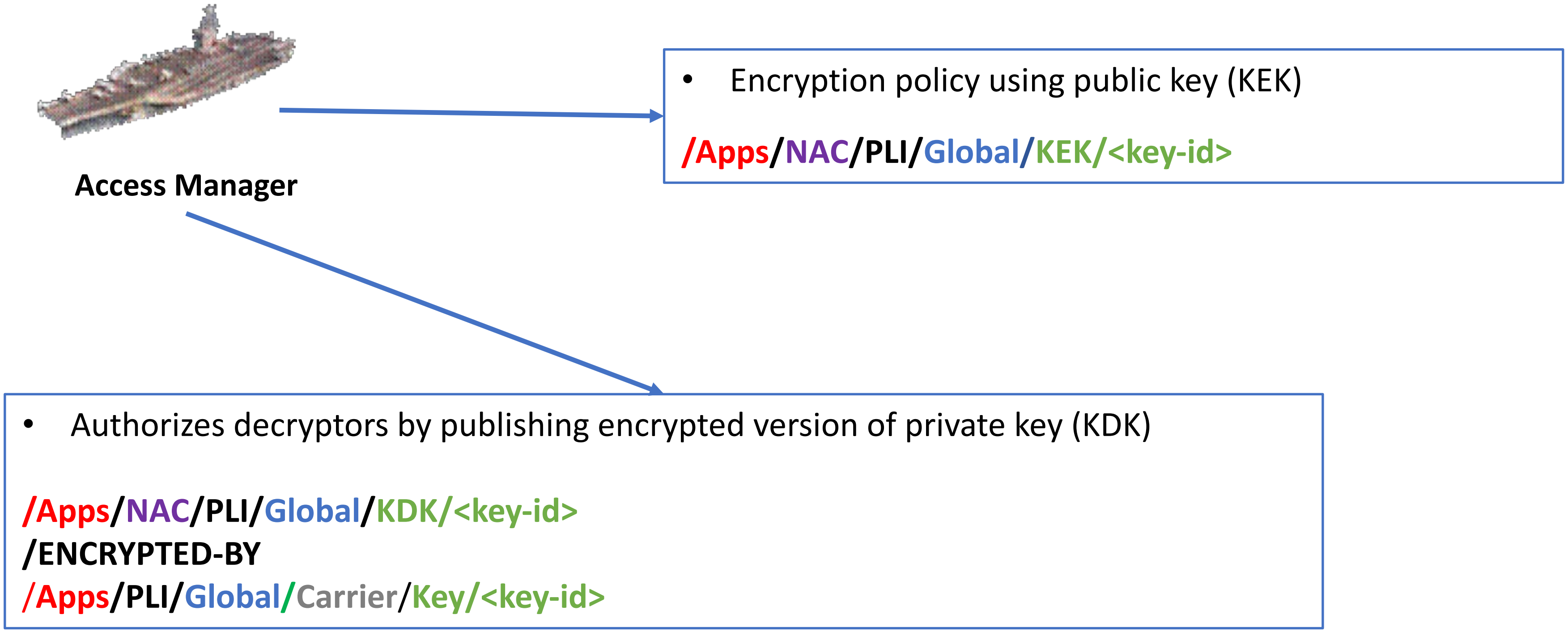
Consumers

Content Key
(C-KEY)

Data

Data

Managing Access Policies



Access Manager

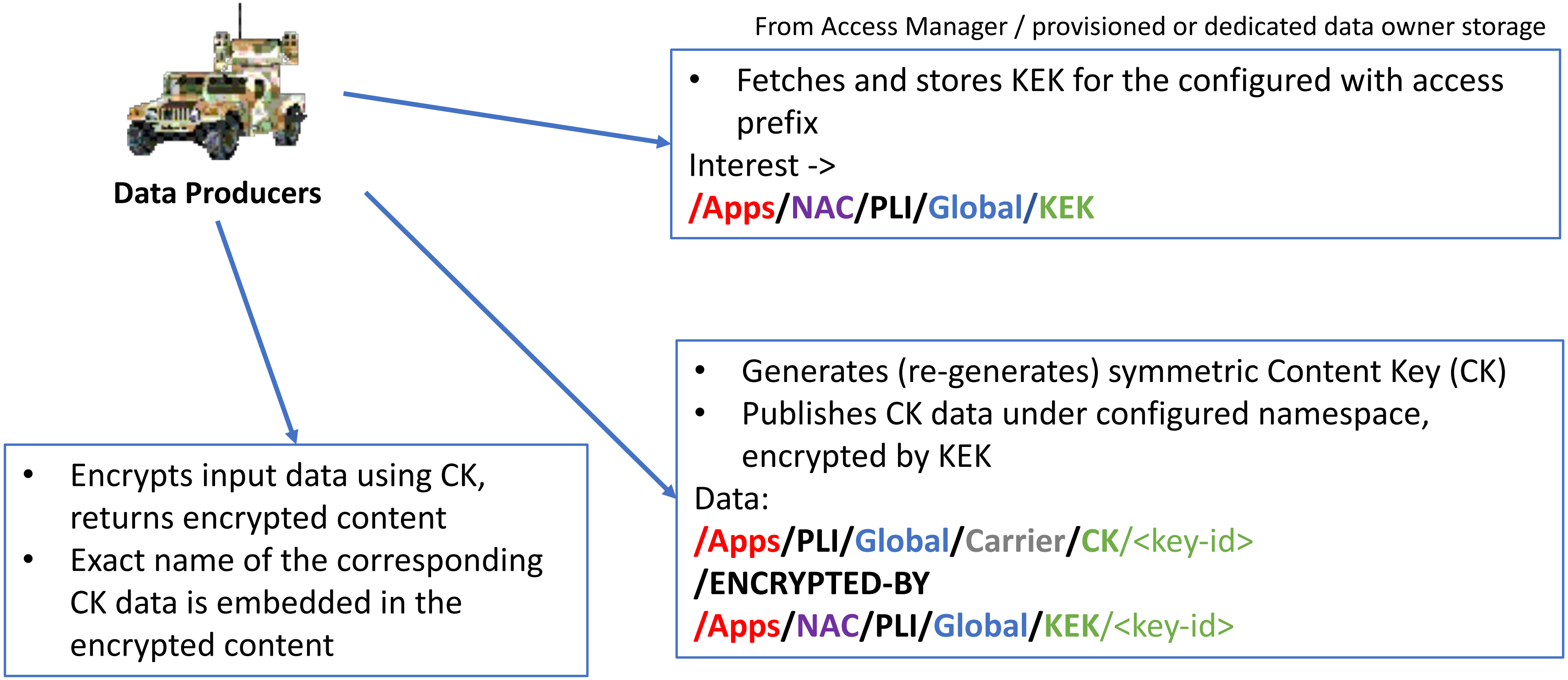
- Encryption policy using public key (KEK)

/Apps/NAC/PLI/Global/KEK/<key-id>

- Authorizes decryptors by publishing encrypted version of private key (KDK)

/Apps/NAC/PLI/Global/KDK/<key-id>
/ENCRYPTED-BY
/Apps/PLI/Global/Carrier/Key/<key-id>

Protection of Data During Production



Data Producers

From Access Manager / provisioned or dedicated data owner storage

- Fetches and stores KEK for the configured with access prefix

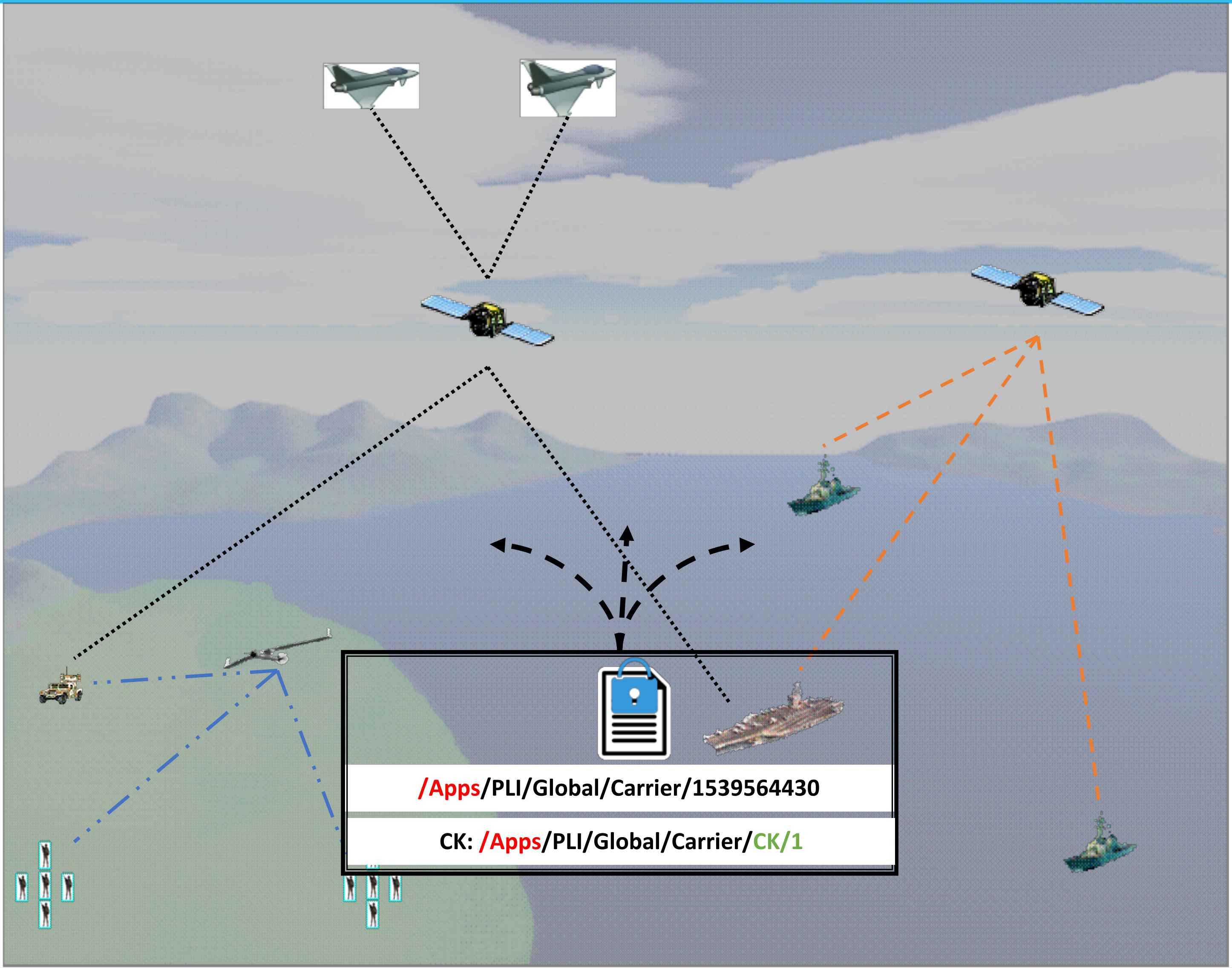
Interest ->
/Apps/NAC/PLI/Global/KEK

- Generates (re-generates) symmetric Content Key (CK)
- Publishes CK data under configured namespace, encrypted by KEK

Data:
/Apps/PLI/Global/Carrier/CK/<key-id>
/ENCRYPTED-BY
/Apps/NAC/PLI/Global/KEK/<key-id>

- Encrypts input data using CK, returns encrypted content
- Exact name of the corresponding CK data is embedded in the encrypted content

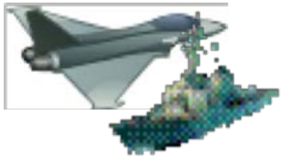
Naming PLI Enc. Keys



/Apps/PLI/Global/Carrier/1539564430

CK: /Apps/PLI/Global/Carrier/CK/1

Access to Protected Data



Data Consumers

- Fetch the encrypted Content Data
- Get the name of the corresponding CK: CK name is embedded in the encrypted content

From Encryptor / from same place as data

- Fetches CK data for the name extracted from input encrypted payload

Interest->
/Apps/PLI/Global/Carrier/CK/<key-id>

- Fetches KDK, name extracted from CK name + own configured access key name

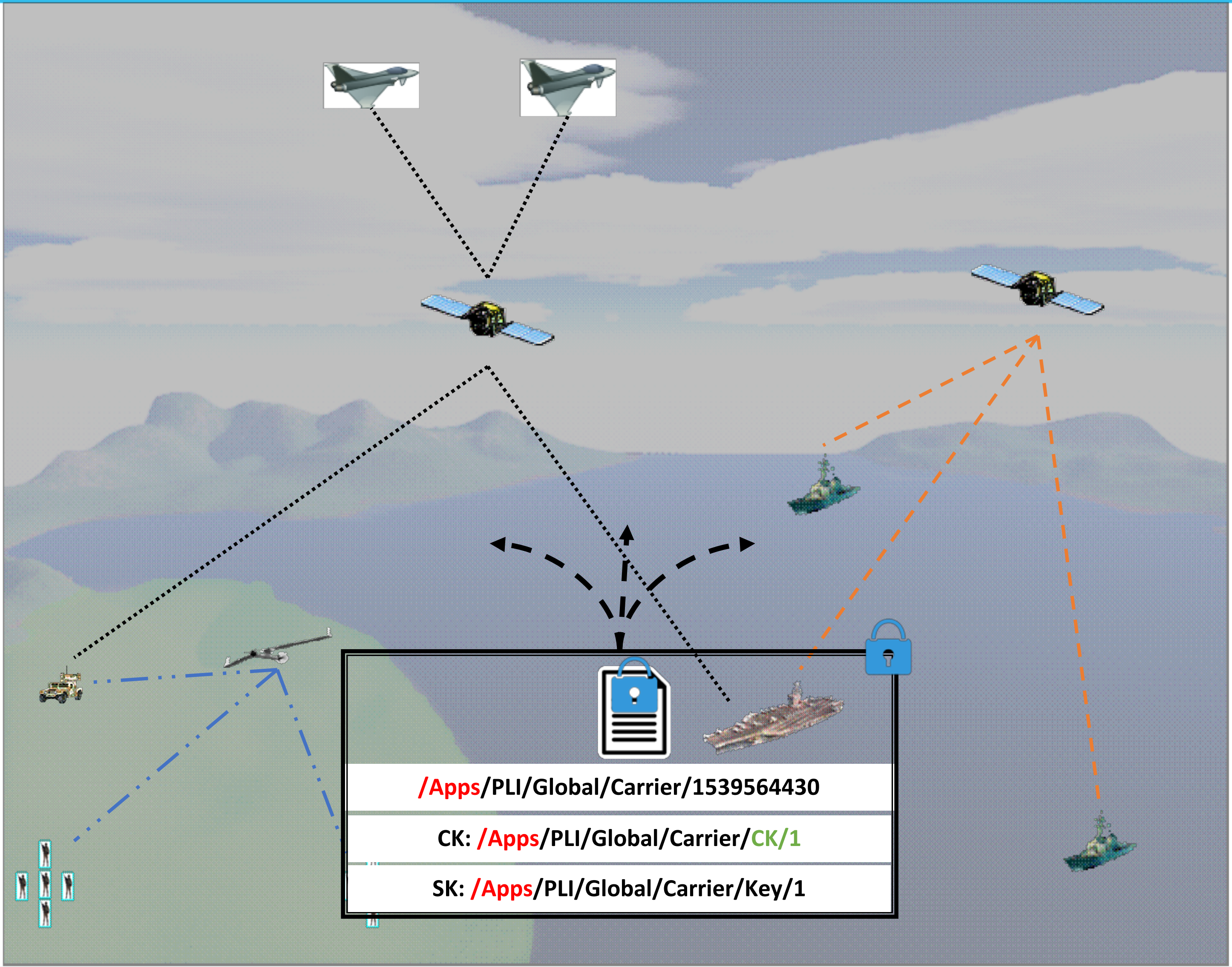
Interest->
/Apps/NAC/PLI/Global/KDK/<key-id>
/ENCRYPTED-BY
/Apps/PLI/Carrier/Key/<key-id>

From Access Manager / provisioned
or dedicated data owner storage

Naming PLI Signing Certs

- Data must be signed by producer of data:
  - /Apps/PLI/Global/Carrier/KEY/<key-id>
  - /Apps/PLI/Global/Destroyer-a/KEY/<key-id>
- Each Data packet points to the name of another Data packet that contains a certificate or public key that was used to produce the signature on it

Naming PLI Signing Certs



/Apps/PLI/Global/Carrier/1539564430

CK: /Apps/PLI/Global/Carrier/CK/1

SK: /Apps/PLI/Global/Carrier/Key/1