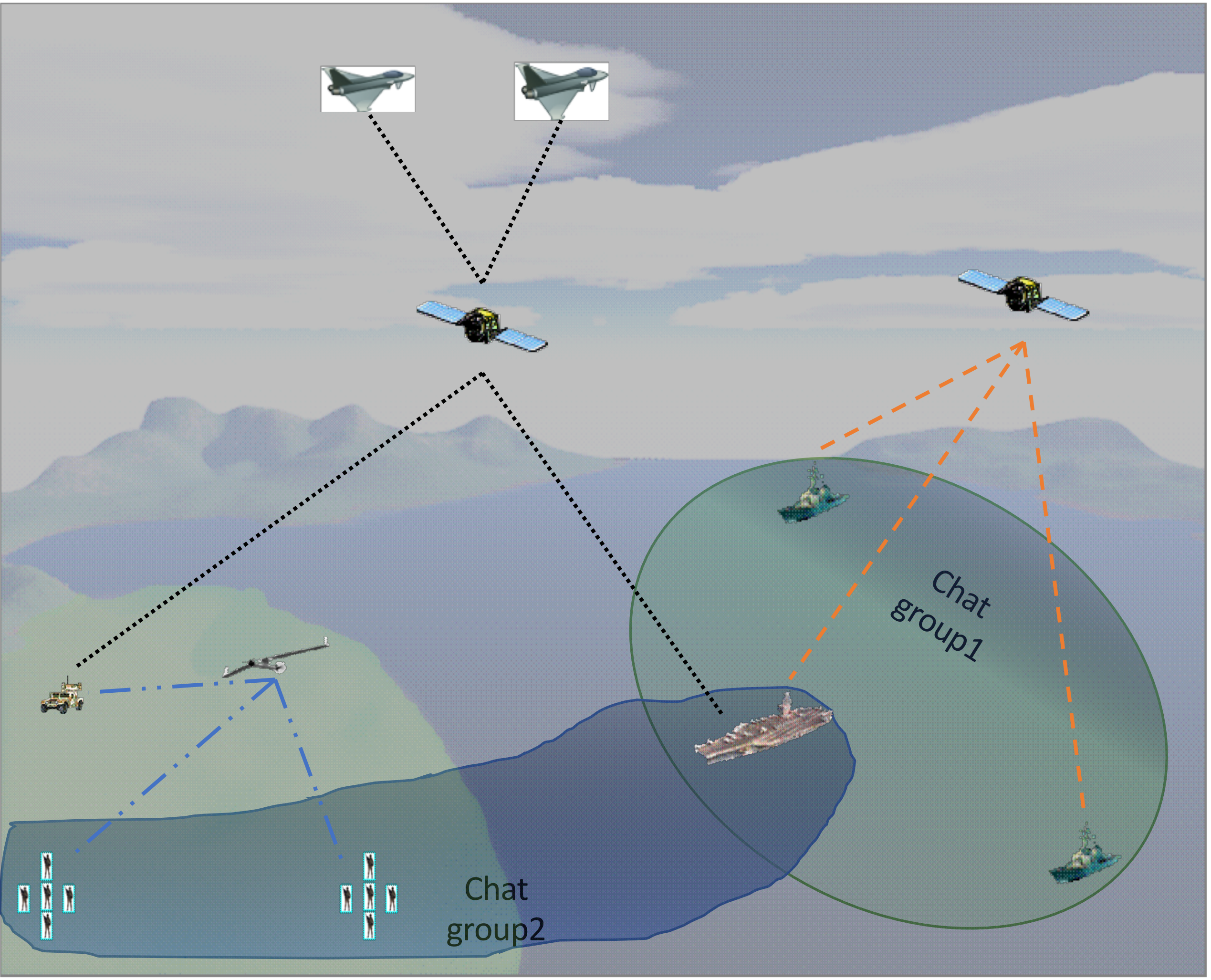
Chat Application

milcom

Military Communications for the 21st Century

LAX Marriott, Los Angeles, CA

Notional Tactical Network

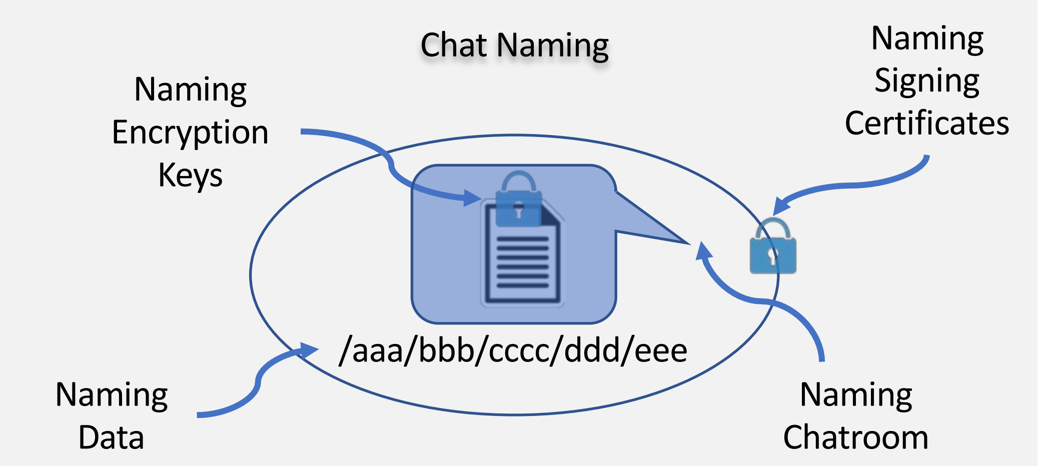


Chat

- Two different chat groups
  - Between the three ships
  - Between the carrier and the troops
- Loss cannot be tolerated
- In order delivery is important
- Can tolerate some delay

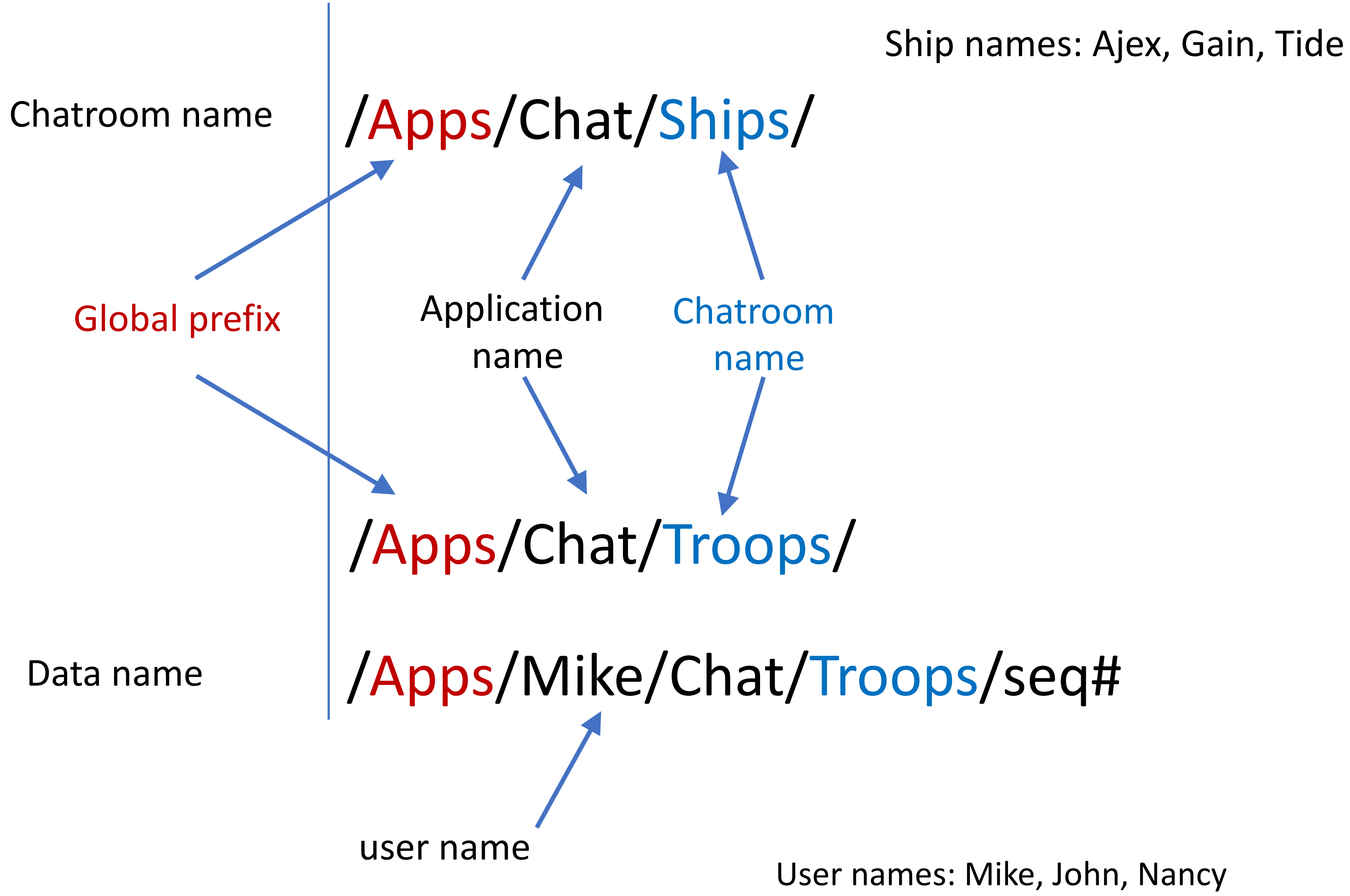
Chat Naming

- As part of the app design/configuration
- What to name:
  - Chatroom
  - Chat data produced by users
  - Encryption keys
  - Signing certificates



- App cares about who’s in the room, and what’s the latest data each has produced
  - User locations may change over time.
  - Direct end-to-end paths between producer-consumers may not exist

Chatroom and Data Naming



Ship names: Ajex, Gain, Tide

Chatroom name

/Apps/Chat/Ships/

Global prefix

Application name

Chatroom name

/Apps/Chat/Troops/

Data name

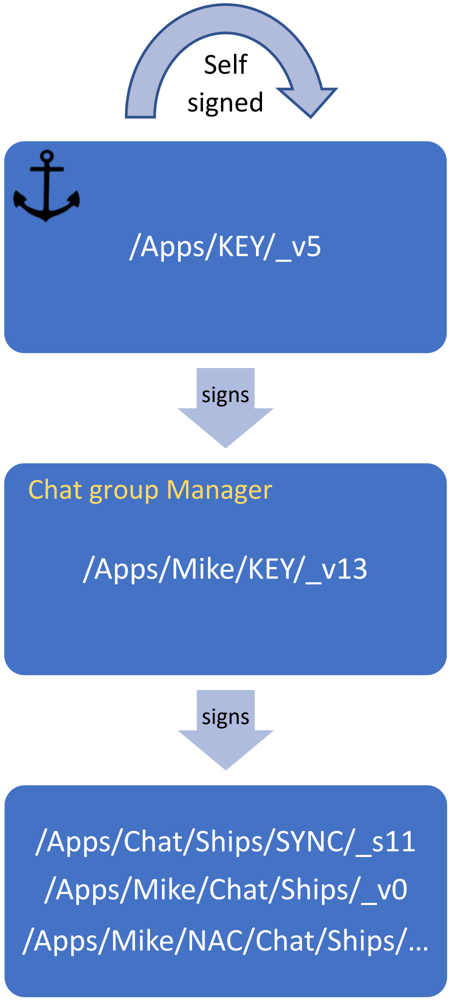
/Apps/Mike/Chat/Troops/seq#

user name

User names: Mike, John, Nancy

Trust Anchor and Keys

- /Apps as the shared trust anchor
  - Identified by a self-signed versioned certificate
  - Cert name: /Apps/KEY/_v5
  - Securely installed out-of-band into all user devices
- Every entity in the network has a cert signed by the trust anchor
  - /Apps/Mike/KEY/_v13
    - A user produces a chat-app key to sign data
- Each chat created by a room manager
  - The manager creates key encryption (KEK)/key decryption key (KDK)
  - Publishes and signs KEK
    - /Apps/Mike/NAC/Chat/Ships/KEK/_v8
  - Encrypts KDK with invited participants public keys and shares with them
    - /Apps/Mike/NAC/Chat/Ships/KDK/_v8
      /ENCRYPTED-BY/Apps/John/KEY/_v42


Self signed
⚓
/Apps/KEY/_v5
signs
Chat group Manager
/Apps/Mike/KEY/_v13
signs
/Apps/Chat/Ships/SYNC/_s11
/Apps/Mike/Chat/Ships/_v0
/Apps/Mike/NAC/Chat/Ships/...

Managing Access Policies

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

Access Manager

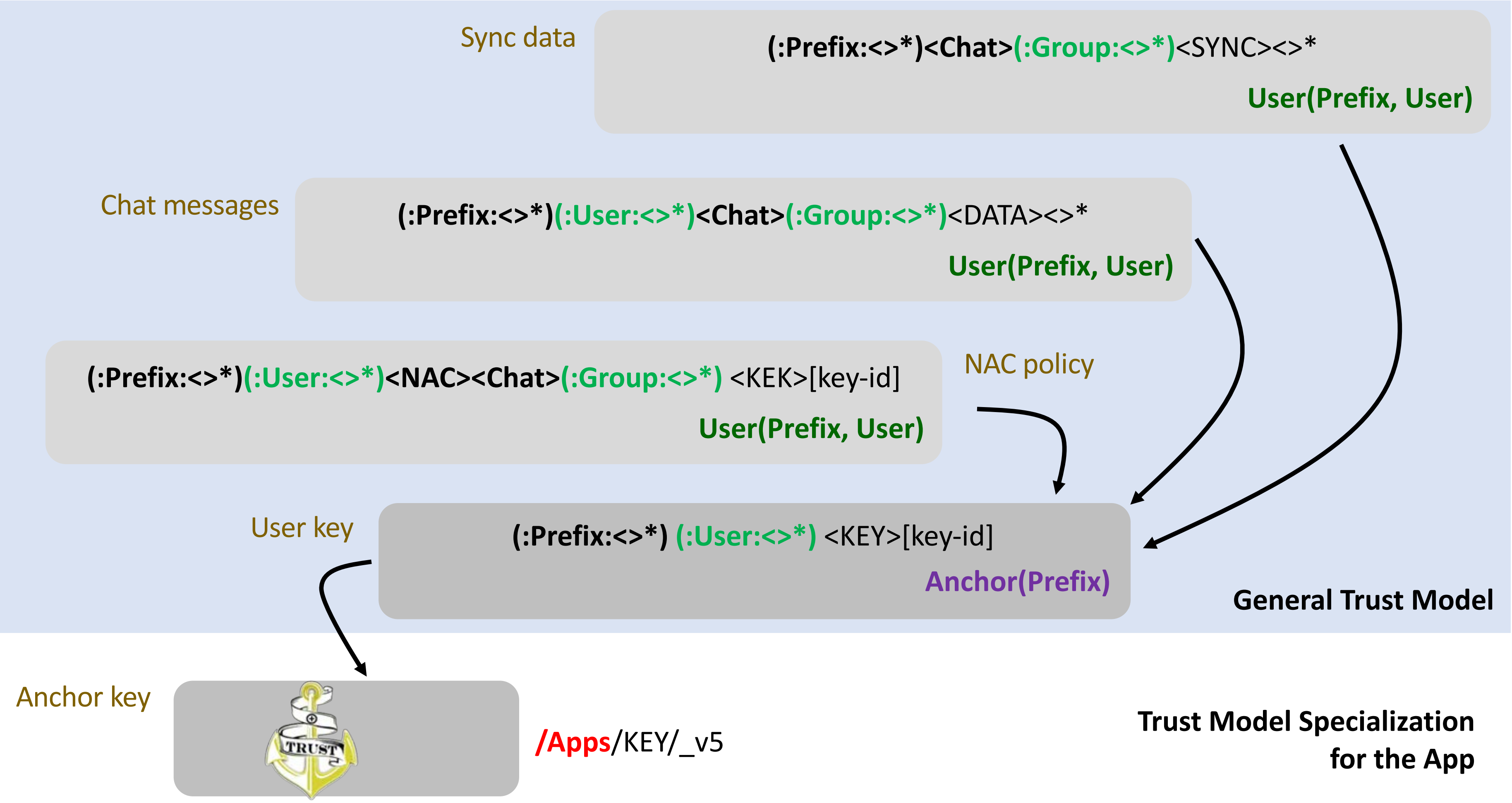
- Encryption policies using public key (KEK) – per created chat group

/Apps/Mike/NAC/**/Chat/Ships**/KEK/<version>
/Apps/Mike/NAC/**/Chat/Troops**/KEK/<version>
/Apps/Mike/NAC/**/Chat/Xyz**/KEK/<version>
…

- Authorizes participants publishing encrypted version of private key (KDK) – per group and per participant

/Apps/Mike/KEK/**/Chat/Ships**/KEY/<version>/ENCRYPTED-BY/…
…
/Apps/Mike/KEK/**/Chat/Troops**/KEY/<version>/ENCRYPTED-BY/…
…
/Apps/Mike/KEK/**/Chat/Xyz**/KEY/<version>/ENCRYPTED-BY/…
…

Example of Trust Schema for Chat



Sync data

(:Prefix:<>*)<Chat>(:Group:<>*)<SYNC><>*

User(Prefix, User)

Chat messages

(:Prefix:<>*)(:User:<>*)<Chat>(:Group:<>*)<DATA><>*

User(Prefix, User)

(:Prefix:<>*)(:User:<>*)<NAC><Chat>(:Group:<>*) <KEK>[key-id]

User(Prefix, User)

NAC policy

User key

(:Prefix:<>*) (:User:<>*) <KEY>[key-id]

Anchor(Prefix)

General Trust Model

Anchor key

/Apps/KEY/_v5

Trust Model Specialization for the App

Chat: Security in Packet Delivery

- False Interest packet injection: protected by the group key
  - Interest packets can be signed
- False data packet injection: mitigated by the built-in data authentication
- Signal interference: exhibited as packet losses
- Eavesdropping: mitigated by encryption as IP does today, but with automated key management
  - With IP, one can use named keys at app layer but no easy way to distribute keys

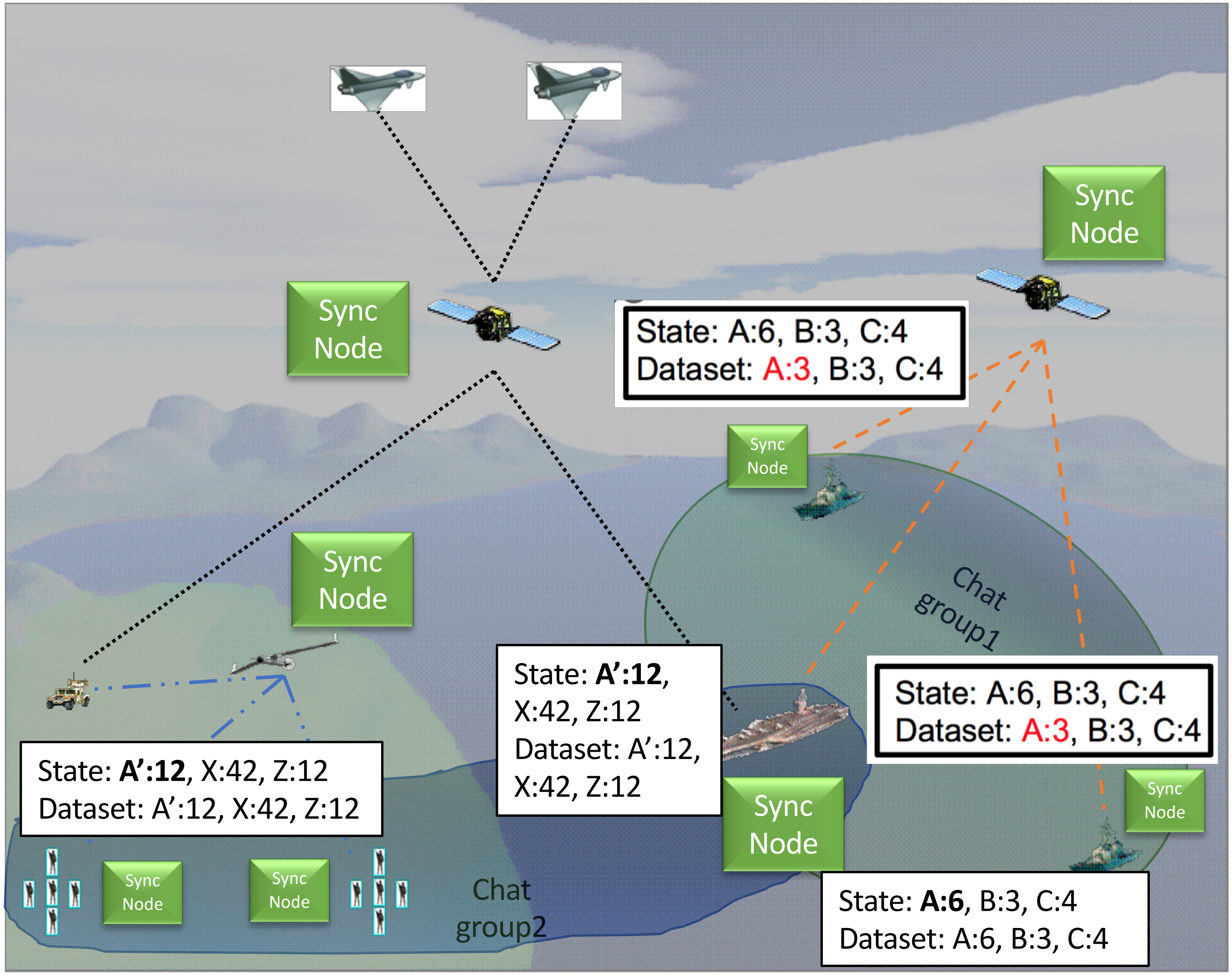
Chatroom: Simplify App Design via Dataset Synchronization

- Each user may produce input into the chat
  - Text messages
  - Image files (each file has associated metadata)
- NDN Sync keeps every user informed of the latest input from all others in the same chatroom
  - Tracking the latest data production sequence#
- Each user decides whether/when to fetch which piece of data
  - If a new piece of data is an image file: the first returned data packet carries metadata to inform the user of the file size and other content specifics

Chatroom: Simplify App Design via ...et Synchronization

6, B:3, C:4
A:6, B:3, C:4

- Different sync nodes can be defined in the topology.
- Carrier needs to sync data from both groups (same with the satellite node connected to it).
- UAV only needs to sync data from group 2.



State: A:6, B:3, C:4
Dataset: A:3, B:3, C:4

State: **A’:12**, X:42, Z:12
Dataset: A’:12, X:42, Z:12

State: **A’:12**, X:42, Z:12
Dataset: A’:12, X:42, Z:12

State: A:6, B:3, C:4
Dataset: A:3, B:3, C:4

State: **A:6**, B:3, C:4
Dataset: A:6, B:3, C:4

Chat: 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

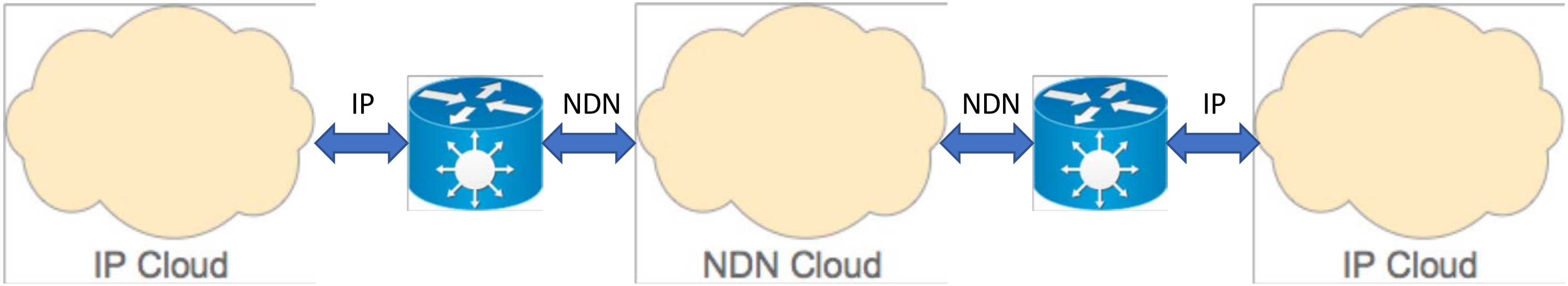
- Receive a data packet but does not have a matching PIT entry at the time
  - May buffer it for future use potential
  - May make the decision based on a filter on name prefixes
- When next time receives an Interest, either from a neighbor node, or from a local app
  - May find matching data in the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

New Chatroom Creation

- Room manager chooses a name, selects members
- Informing the members of the new chat
  - Notify each of them via a signaling Interest
    - Notification encrypted using individual's public key
  - Pub-sub: Publish the notification data (for each member) through an established notification namespace
    - Everyone can sync or periodically pull this space
- Members can learn about each other's latest data production through State-vector Sync
  - Sync Interest enumerates the member list

Integration of Existing Applications

- Integration of existing applications can be done through gateways
  - Speak both IP and NDN
- Complexity of such applications depends on the nature of the existing application


IP Cloud
IP
NDN
NDN Cloud
NDN
IP
IP Cloud