FLIC Manifests

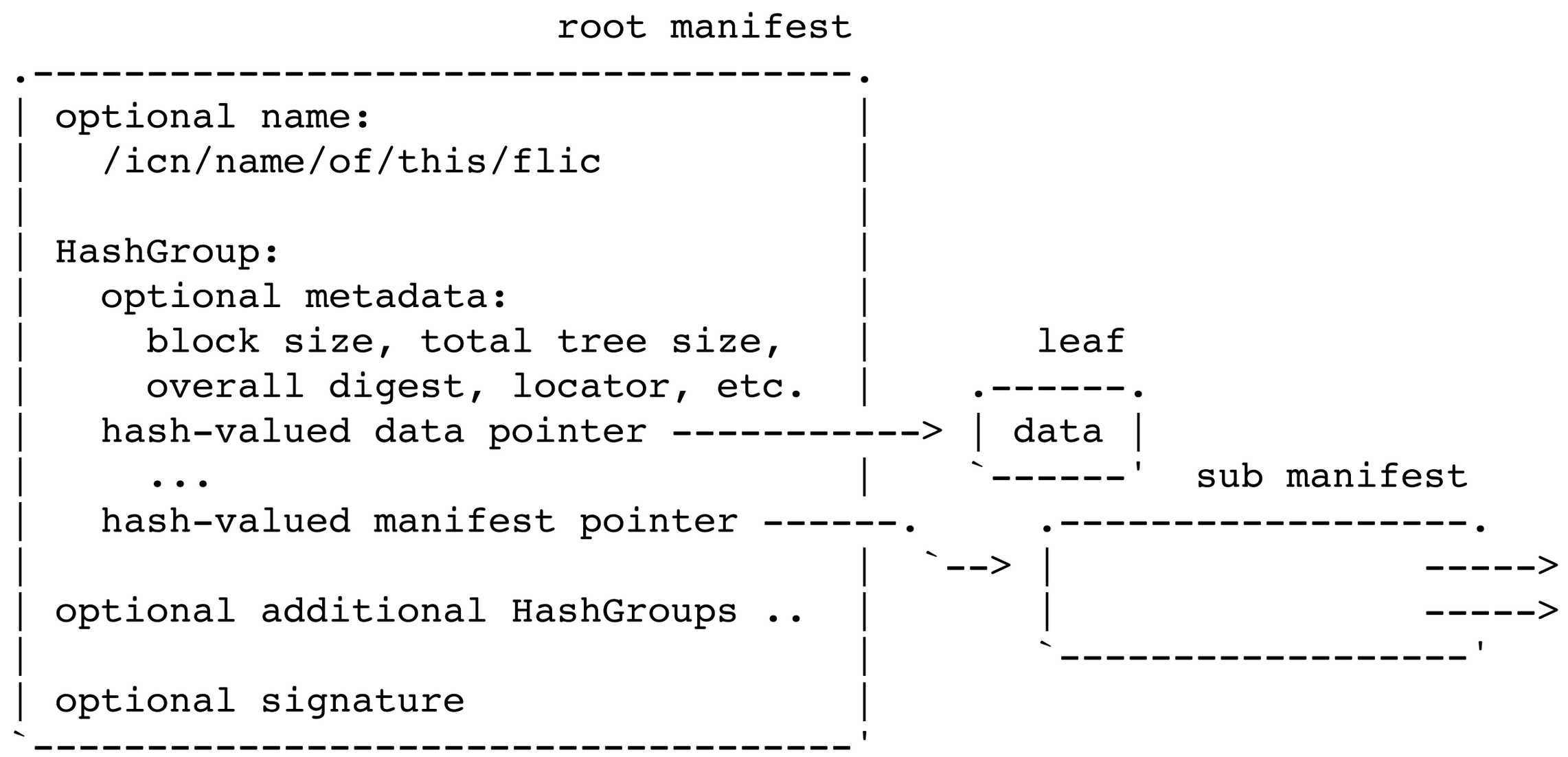
Goals

- Copy UNIX inode concept
    - Index tables and memory pointers
- Adaption to ICN
    - Hash values instead of block numbers
- Advantages
    - Single root manifest signature that covers all children nodes
    - No need for explicit chunking
    - Supports block-level de-duplification

Limitations

- All data leaves must be present at creation time

FLIC Overview



FLIC Grammar

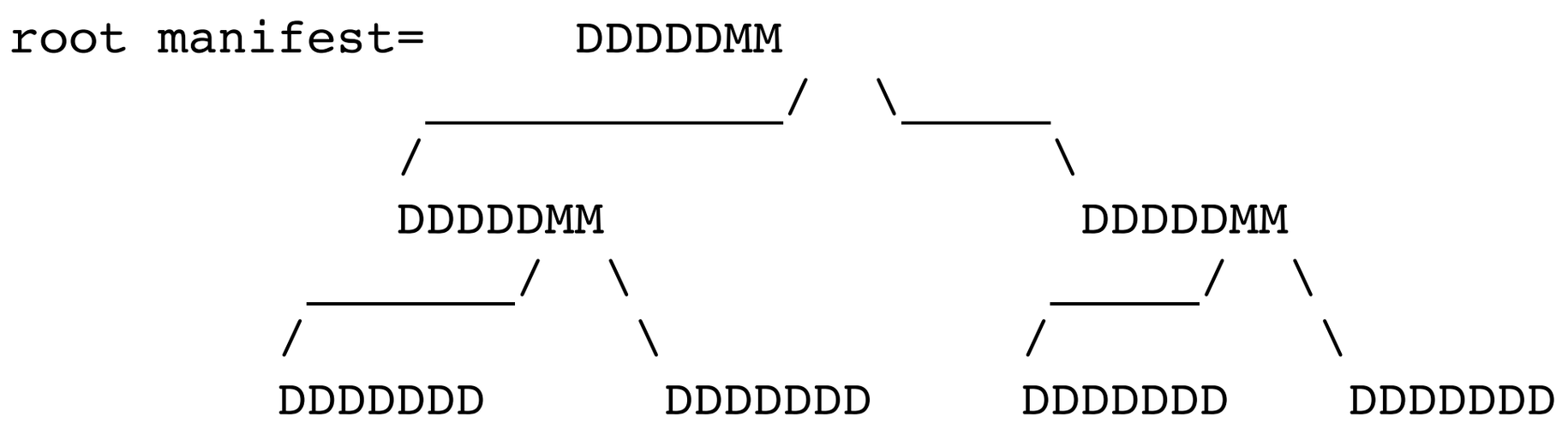
```
ManifestMsg := Name? HashGroup+
HashGroup   := MetaData? (DataPointer | ManifestPtr)+
DataPointer := HashValue
ManifestPtr := HashValue
HashValue   := OCTET[32]

MetaData    := Property*
Property    := Locator | DataBlockSize | OverallDataSize |
               OverallDataSHA256Digest | ...
```

FLIC Trees

- Skewed (similar to a list)

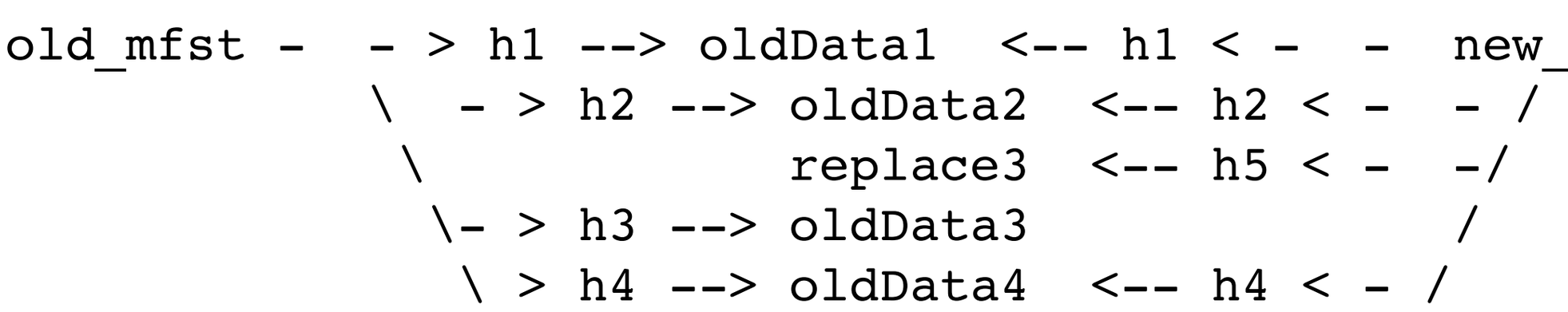
- Balanced



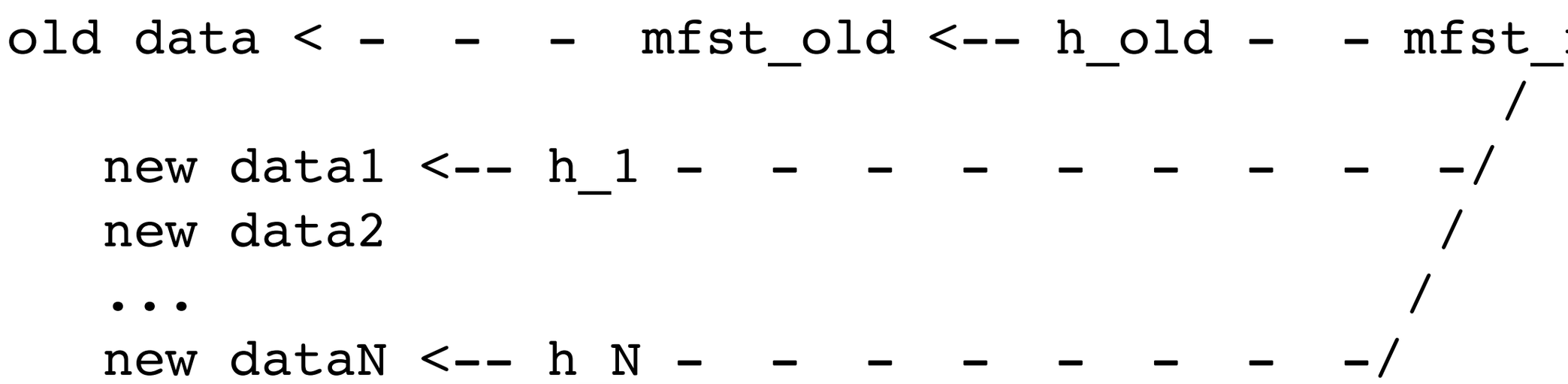
Use Cases

- Block-level reduplication
- Growing ICN collections
- Republishing under a new name
- Data chunks of variable size

Deduplification



Growing Collections



Republishing

- Create nameless manifest trees.
- Use “named manifest” that points to the nameless root.

```
data < - nameless_mfst() <-- h  < - mfst(/com/parc/east/the/flic)
                                < - mfst(/com/parc/west/old/the/flic)
                                < - mfst(/internet/archive/flic234)
```

Variable Size Blocks

```
M = (manifest
        (hashgroup((metadata(blocksize=12)) (dataptr=h1))
        (hashgroup((metadata(blocksize=1)) (dataptr=h2))
        ...
      )
```

Some Questions

- Should each pointer have an associated size?
- What are the mechanics for handling manifests with multiple hash groups?
- How should encryption be handled?