Manifest Proposal Variations

v7: Mosko, Tshudin, Wood

2015-10-3

Revisions

- v1 (caw): Initial version without ccn-lite or PARC advanced details
- v2 (cft): Modified and improved ccn-lite details, added EBN descriptions, reorganized document
- v3 (caw): Added new PARC advanced design
- v4 (caw+cft): Added comments from 9/8/15 meeting, extended observations, and updated some figures
- v5 (cft): aligned with FLIC name (instead of ccn-lite)
- v6 (caw): Replaced the PARC advanced variant with another PARC basic variant
- v7 (caw): Integrated comments and feedback from Marc Mosko, in addition to his proposed Manifest design

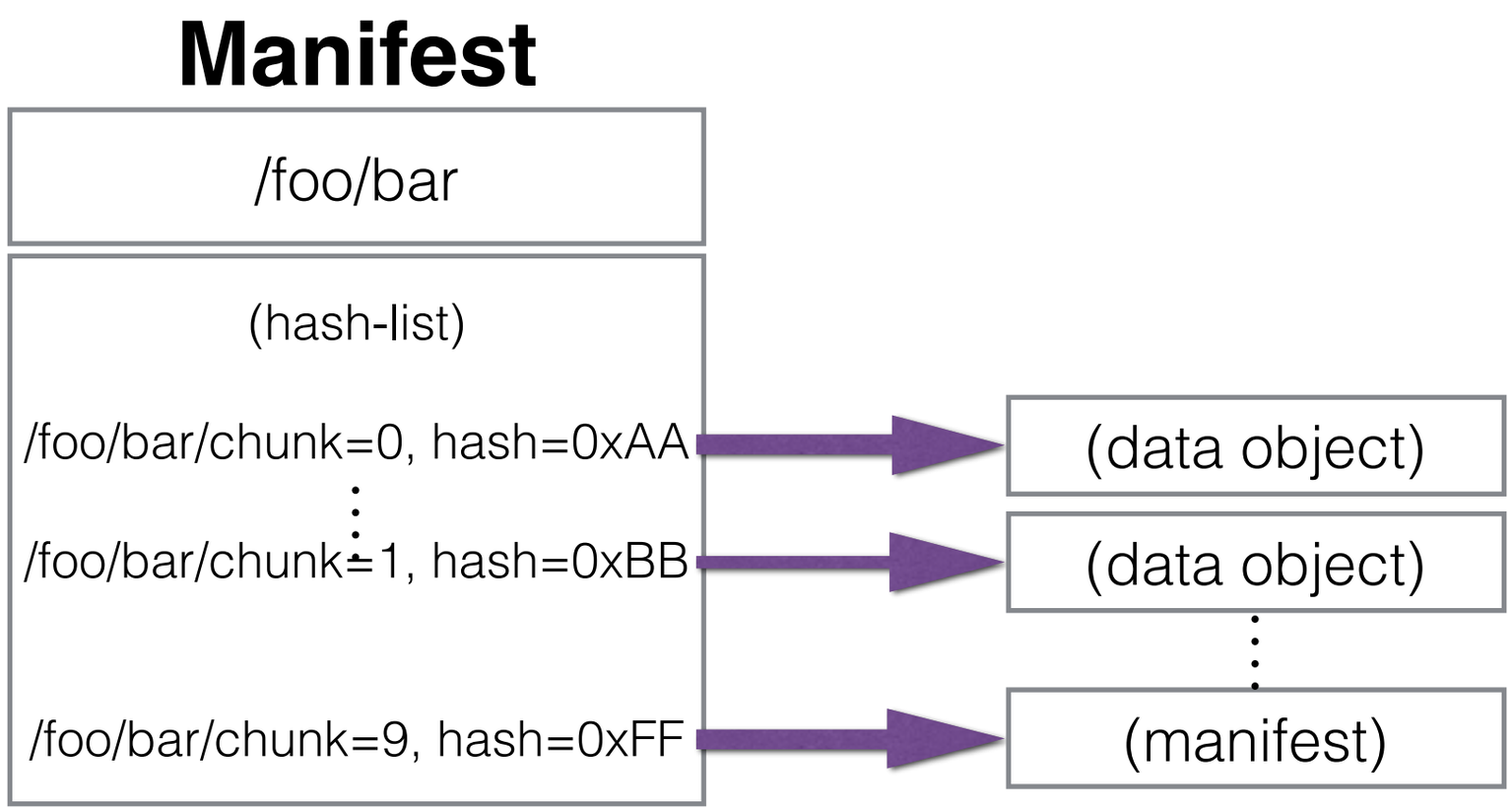
Overview

- List of Manifest proposals:
  - CCNx 0.x signed collection of links (2009)
  - NDN (Ilya's work, 2014)
  - CCNx All-in-One Streams for CCN (2014, omitted)
  - CCNx Basic V1 (spring 2015)
  - ccn-lite FLIC (summer 2015)
  - CCNx Basic V2 (summer 2015)
  - CCNx Basic V3 (fall 2015)
- Per proposal: EBN, observations

CCNx 0.x

```
ManifestPayload = PointerList
PointerList = *(Name, Hash)
```

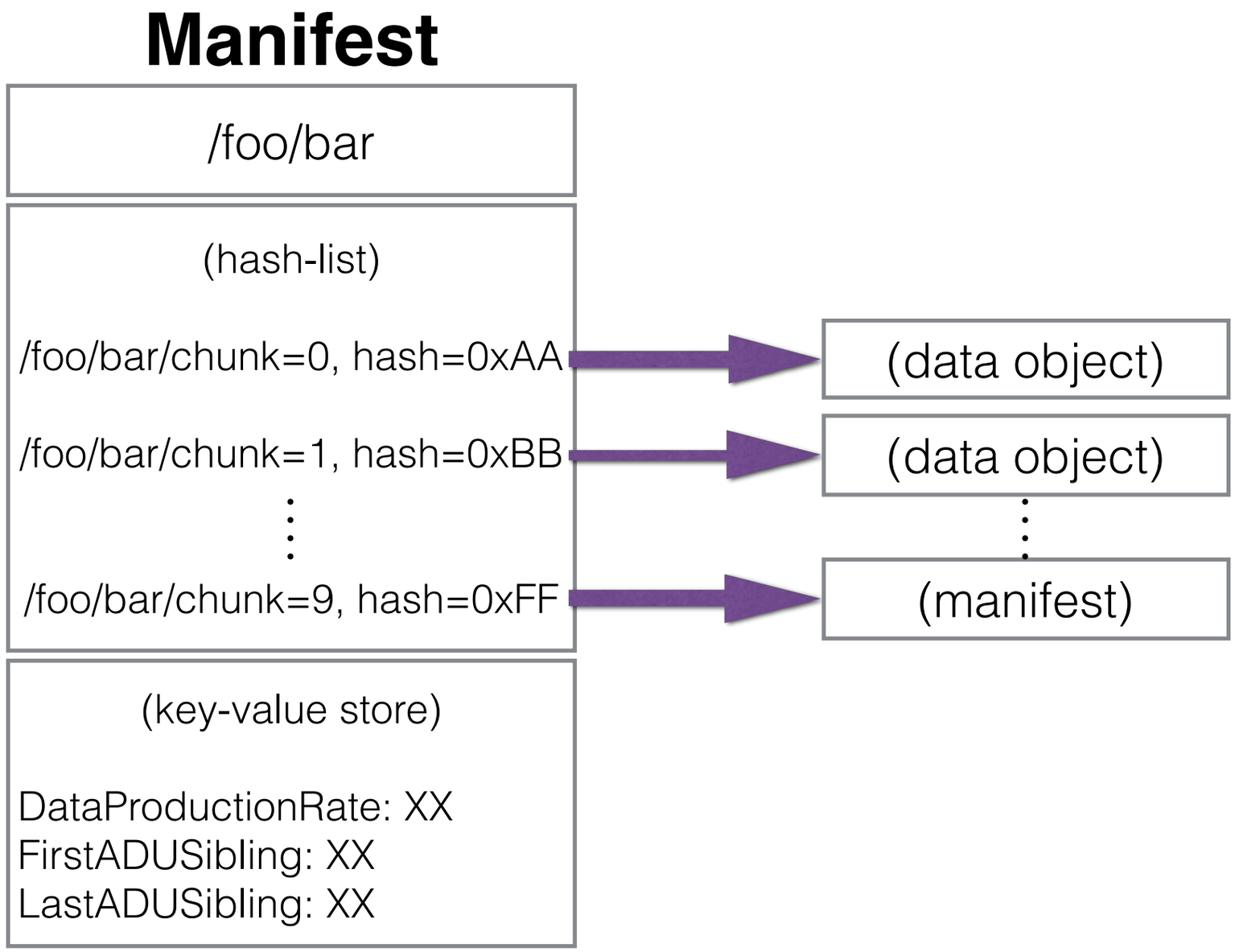
CCNx 0.x Structure



NDN Structure EBN

```
ManifestPayload = PointerList [MetaData]
PointerList = *(Name, Hash)
```

NDN Structure



Source: Consumer / Producer communication with application level framing in Named Data Networking, in ACM ICN 2015.

NDN: Properties and Observations

Flexible metadata key-store

Unclear relation to group-based encryption: Decryption metadata is stored outside of the manifest (in the KeyLocator of data packets)?

Supports coarse-grained data-deduping with nested manifests

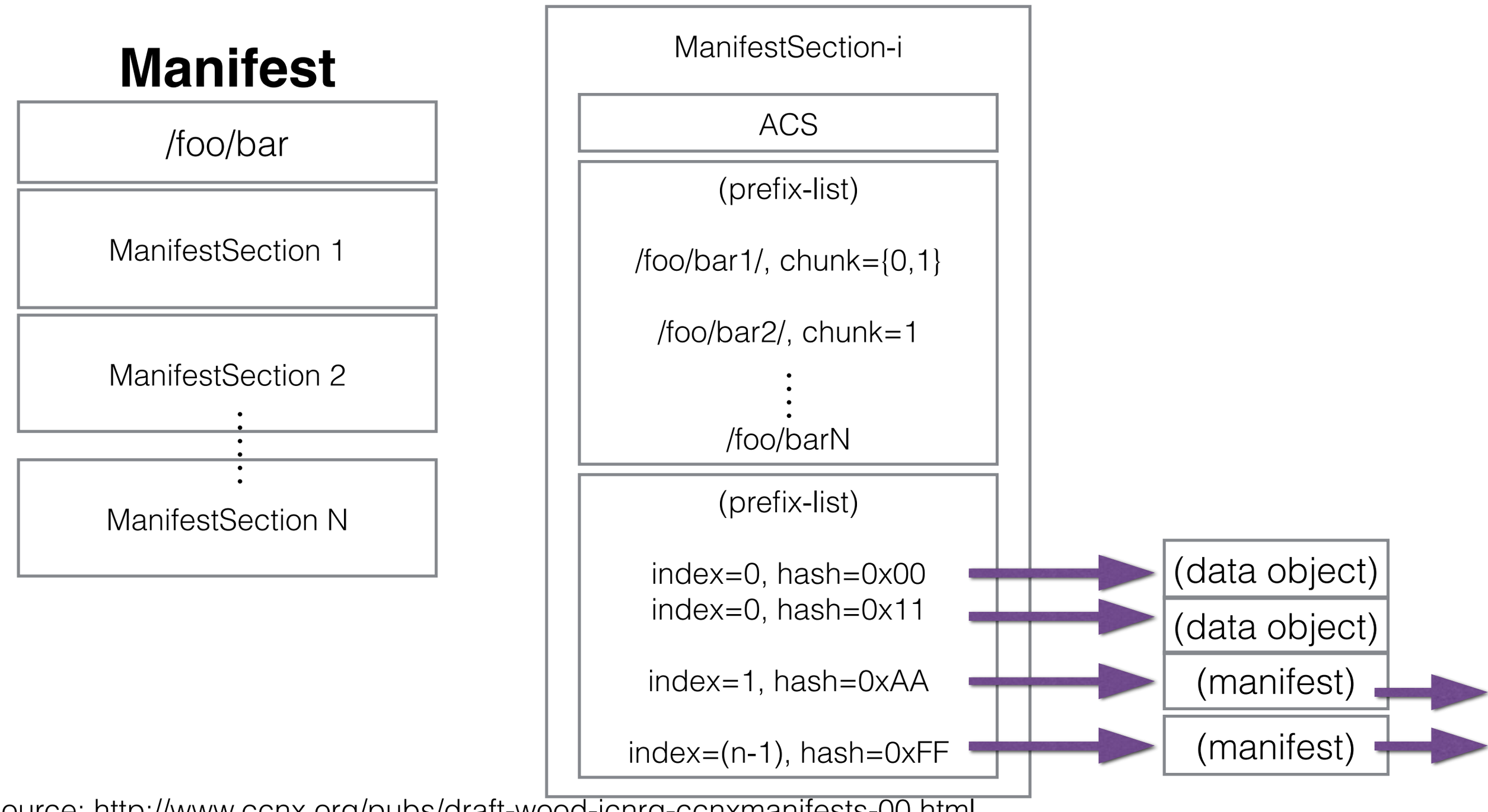
CCNx Basic Structure EBN

CCNx basic:

```
ManifestPayload = (ManifestSection)*
ManifestSection = SECTION | LINK
SECTION         = [ACS] [ListOfPrefixes] ListOfHashes

ACS             = LINK
ListOfPrefixes  = (PrefixEntry)*
NameEntry       = [StartChunk] ContentNamePrefix
ListOfHashes    = PrefixIndex HASH
PrefixIndex     = OCTET
HASH            = 32(OCTET)
LINK            = (Name, KeyIdRestr, ContObjHashRestr)
```

ACS = access control spec (= link to a list of decryption keys)

CCNx Basic Structure EBN



Source: http://www.ccnx.org/pubs/draft-wood-icnrg-ccnxmanifests-00.html

CCNx Basic: Properties and Observations

Support for data-deduping with different sections

Native decryption metadata support (with the ACS), where each section's ACS is independent

Flexible data-to-manifest encoding in each manifest

Manifest pointers may or may not use chunked names

No generic metadata structure

Potentially redundant name information in the prefix-list (why not make names derived from base name of the manifest?)

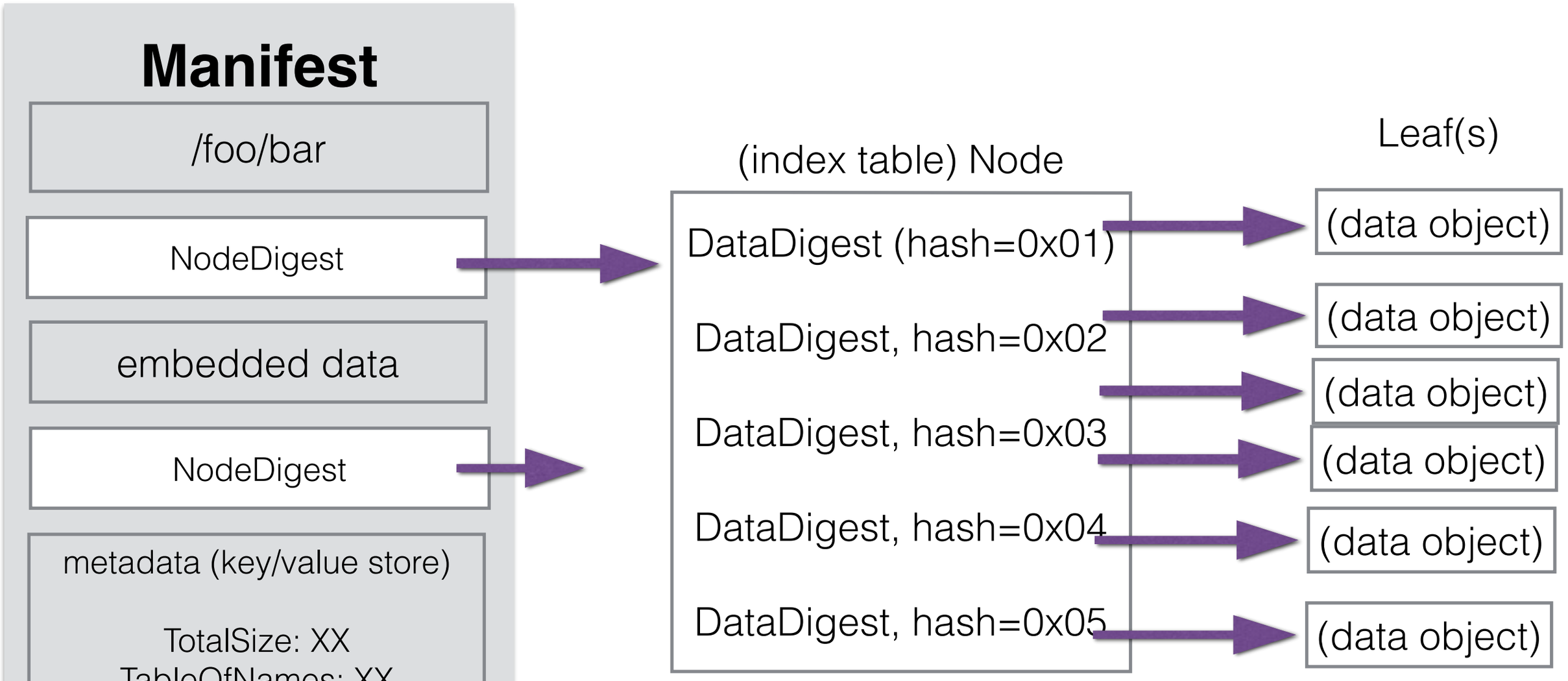
FLIC Structure EBN

FLIC = File-LIke Collection, using UNIX “index tables” as a model for the distributed data structure

```
ManifestPayload = (Node | EncrNode) [MetaData]
Node = *([RepeatCnt]
         (Pointer | NoneByte | ZeroByte | Leaf)
        )
Pointer = (LeafDigest | EncrLeafDigest|
           NodeDigest | EncrNodeDigest)

EncrNode = Blob
Leaf = Blob
RepeatCnt = INTEGER
LeafDigest = 32(OCTET)
NodeDigest = 32(OCTET)
EncrLeafDigest = 32(OCTET)
EncrNodeDigest = 32(OCTET)
Metadata = key-value store
```

Source: Personal communication with Christian Tschudin

FLIC Example



FLIC: Properties and Observations

Easily represent metadata and different traversal strategies since **manifests are only at the root-level**

Built-in compression with Zero entries; None entries for sparse files (e.g. memory dumps)

Pointers have a type: consumer knows if target block is encrypted or not, is a leaf or not

Bytes can be embedded in non-leaf nodes: self-contained manifest-plus-its-data object

Pointers are pure hashes (the manifest's name serves as default locator) — **simplicity**

Metadata is not supported at graph-level (non-root), e.g. size-of-this-subtree=XYZ

Encrypted metadata not directly possible, must introduce a key/value pair for external ref: moreEncrMetadata ="/the/name' which would point to an encrypted metadata object

Because pointers are hashes — how to point to future data? Index table not extensible to streams. And how can one mix-and-match from different prefixes/locators in each node?

FLIC (Additional Notes)

- Introduces the following object types:
  - manifest
  - leaf node (pure data)
  - index node (sequence of entries and embedded data)
  - encrypted index node
  - encrypted leaf node

- **Question:** Can one have “naked index node” objects? Yes, for symmetry reasons to EncrNodes.

- Pointers are pure hashes, the manifest’s name serves as locator. Consequences:
  - no extensible streams?
  - how to mix-and-match from different prefixes?

- Marc’s concern about keeping the fetch pipeline full by having enough hashes per object: job of the encoder.

- Manifests as envelopes: small objects can carry metadata AND the content bits in a single “manifest” object.

- **Question:** Should one add a TotalSize field to an index node (for this subtree?)

See a full description of FLIC at
`draft-tschudin-icnrg-flic-00.txt`

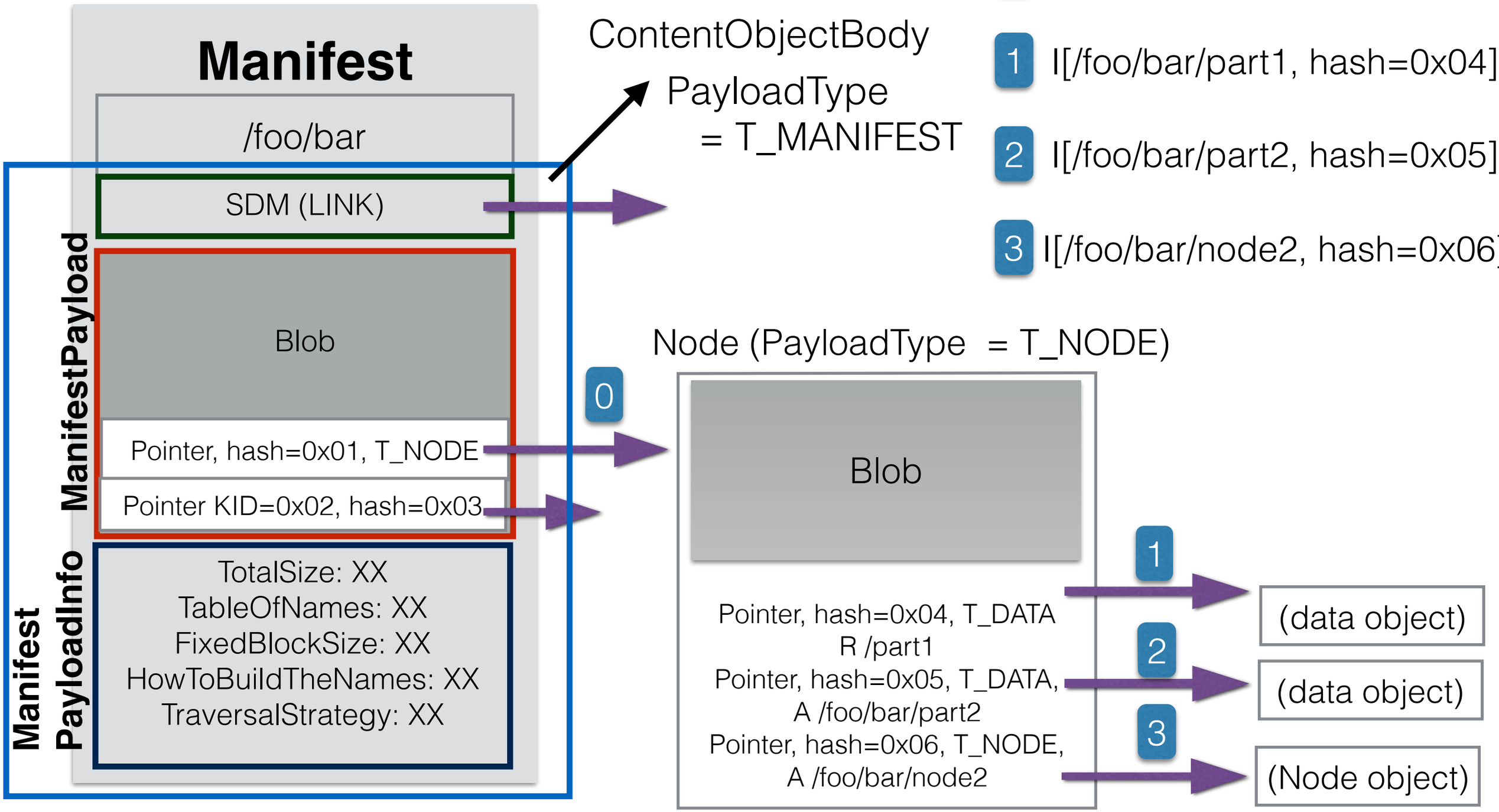
CCNx Basic V2 Structure EBN

CCNx Basic V2: A minimalist extension of the CCNx Basic V1 design with FLIC-like features

```
ContentObject = [Name] [ExpiryTime] ContentObjectBody [Validation]
ContentObjectBody = PayloadType (Payload | ManifestBody | Node)
PayloadType = T_DATA | T_MANIFEST | T_NODE

ManifestBody = [SDM] [[ManifestPayloadInfo] Node]
SDM = LINK | <see doc>
ManifestPayloadInfo = <key-value store>

Node = [Payload] (Pointer)*
Pointer = [[T_RELATIVE | T_ABSOLUTE] Name]? KeyIdRestr? HashRestr [PointerType]
PointerType = T_NODE | T_DATA


Payload = Blob
KeyIdRestr = HASH // 32(OCTET)
HashRestr = HASH // 32(OCTET)
Name = CCNxName
```

SDM = Structured Decryption Metadata (previously the ACS)

CCNx Basic V2 Structure



CCNx Basic V2: Properties and Observations

No manifest hierarchies — there is a single root manifest which describes a tree of nodes (nodes are regular Content Objects)

Read-access metadata is contained outside of the PayloadInfo in the SDM

Pointers are typed LINKs that allow relative and absolute name composition (R/A flags)

Supports direct embedding of data in the root manifest and each node

PayloadInfo is metadata about the payload and is a generic KV-store

PayloadInfo is optional and only present if a Manifest carries a Payload

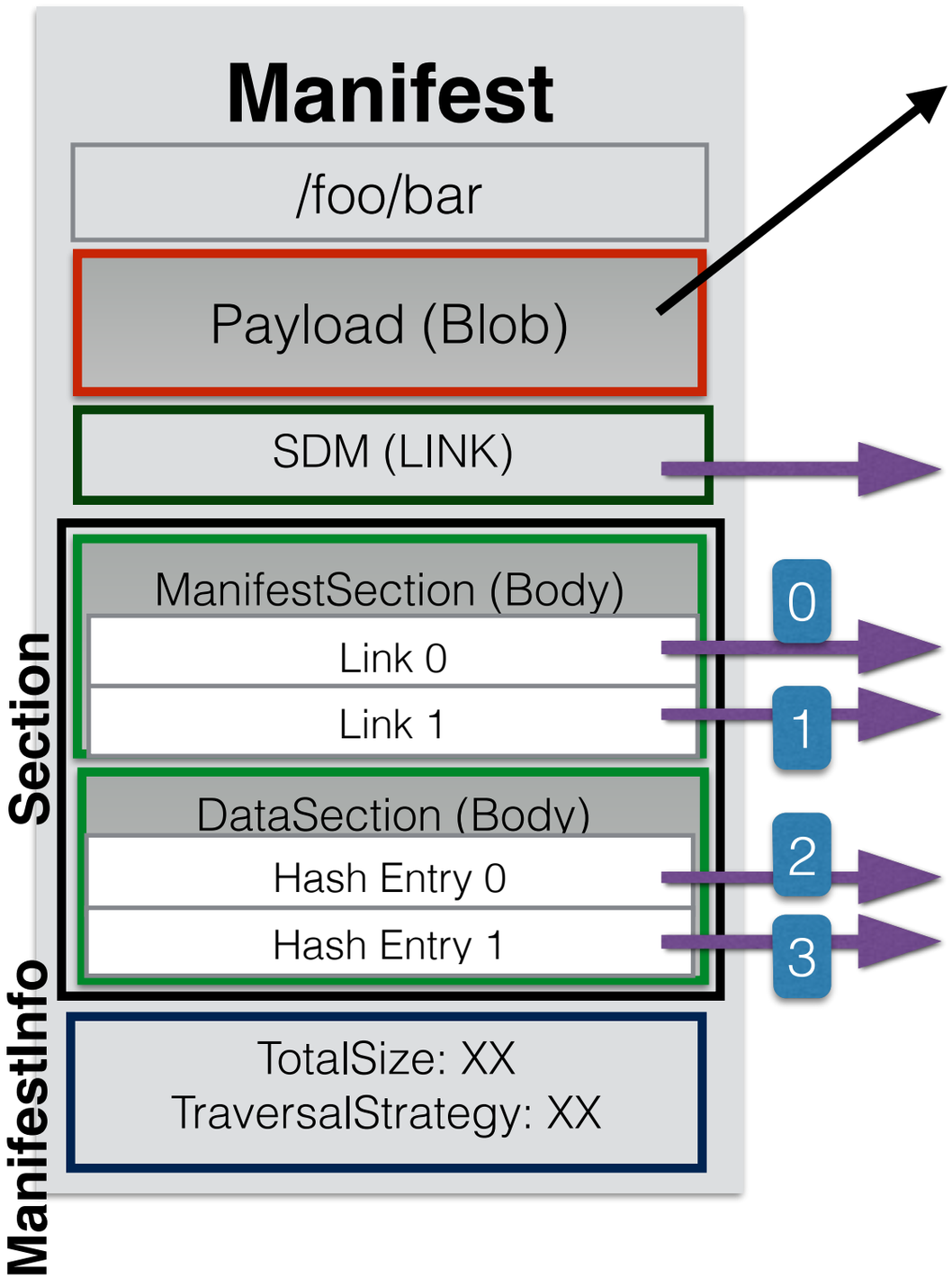
Node Payload blob could (should?) be moved into the parent Content Object payload

CCNx Basic V3 Structure EBN

CCNx Basic V3: A different extension of the CCNx Basic V1 design

```
ContentObject = [Name] [ExpiryTime] ContentObjectBody [Validation]
ContentObjectBody = (ManifestBody | Payload | ManifestBody Payload)

ManifestBody = (SDM | [SDM] Section*) [ManifestInfo]
Section = ManifestSection | DataSection
ManifestSection = SectionBody ; entries to manifests
DataSection = SectionBody ; entries to not manifests
SectionBody = LinkBody | HashBody

LinkBody = Link+
Link = Name [KeyIdRestr] [HashRestr]
HashBody = [Name | KeyId | Name KeyId] [StartChunkNumber] EntryList
StartChunkNumber = Integer ; appended as a chunk to the name
EntryList = HashEntry+

Payload = Blob
Name = CCNx Name
KeyId = 32(OCTET)
HashEntry = 32(OCTET)
ManifestInfo = key-value store
SDM = LINK | <see doc>
```

CCNx Basic V3 Structure



This **ContentObject payload** sits parallel to the Manifest Information

0 I[/random/prefix/child-manifest1, hash=0x01

1 I[/another/one/child-manifest2, hash=0x04]

2 I[/x/y/z/chunk=2, hash=0x05]

3 I[/x/y/z/chunk=3, hash=0x06]

***ManifestSection contains a LinkBody with fully qualified Links

***DataSection contains a HashBody with name /x/y/z/ and start chunk 2

CCNx Basic V3: Properties and Observations

There are zero or more ManifestSections that point to a hierarchy of manifests.

There are zero or more DataSections that point to leaf nodes.

A SectionBody can be a list of spelled-out links or a list of hashes relative to a single name.

Each HashBody has at most one Name.

If Name is missing from a Section, it uses the ContentObject name less any chunk number.

If KeyId is present, it is used in the Interest for an Entry.

Payload is always user data, not mixed with Manifest encoding. There is no longer a "PayloadType = Manifest”.