CCN

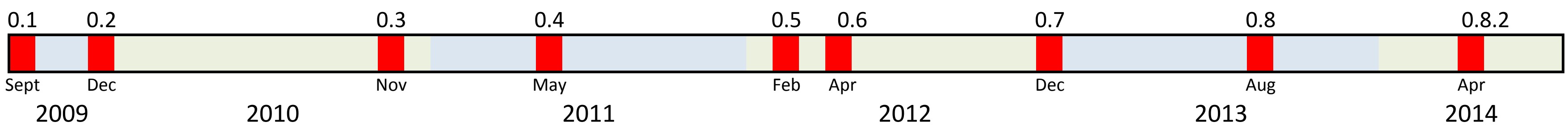
CCNx 1.0 Evolution From Experiments

Computer Science Laboratory
Networking & Distributed Systems

March 2014

parc
A Xerox Company

CCNx 0.x Releases



CCNx 0.x Open Source
5 Years of Releases

Pioneering work in practical
Information Centric Networking

Basis of NSF Future Network Architecture
Named Data Networking

CCN 1.0

Cleaned API

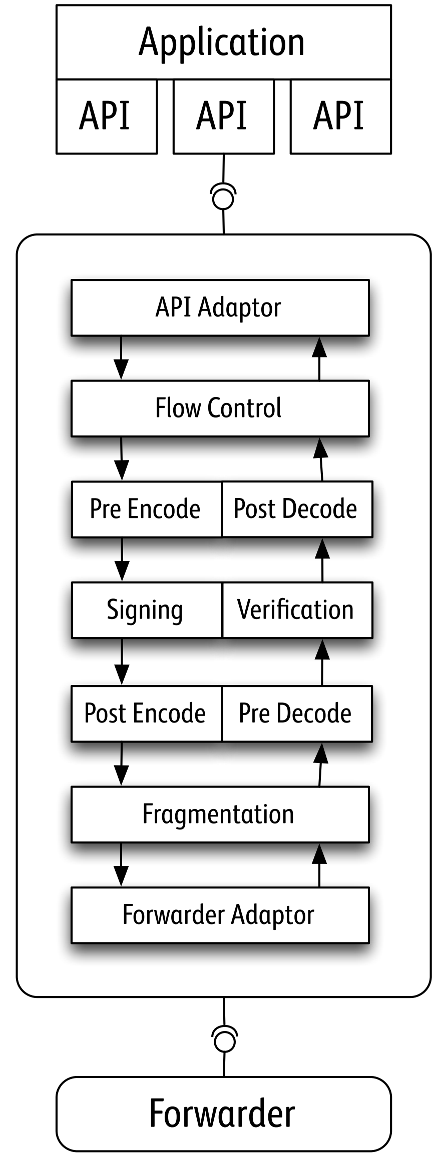
Lowered the learning curve
Separated concerns

Cleaned Code

Improved maintainability
Increased modularity

Cleaned Protocol

Defined minimum protocol
Specified auxiliary protocols



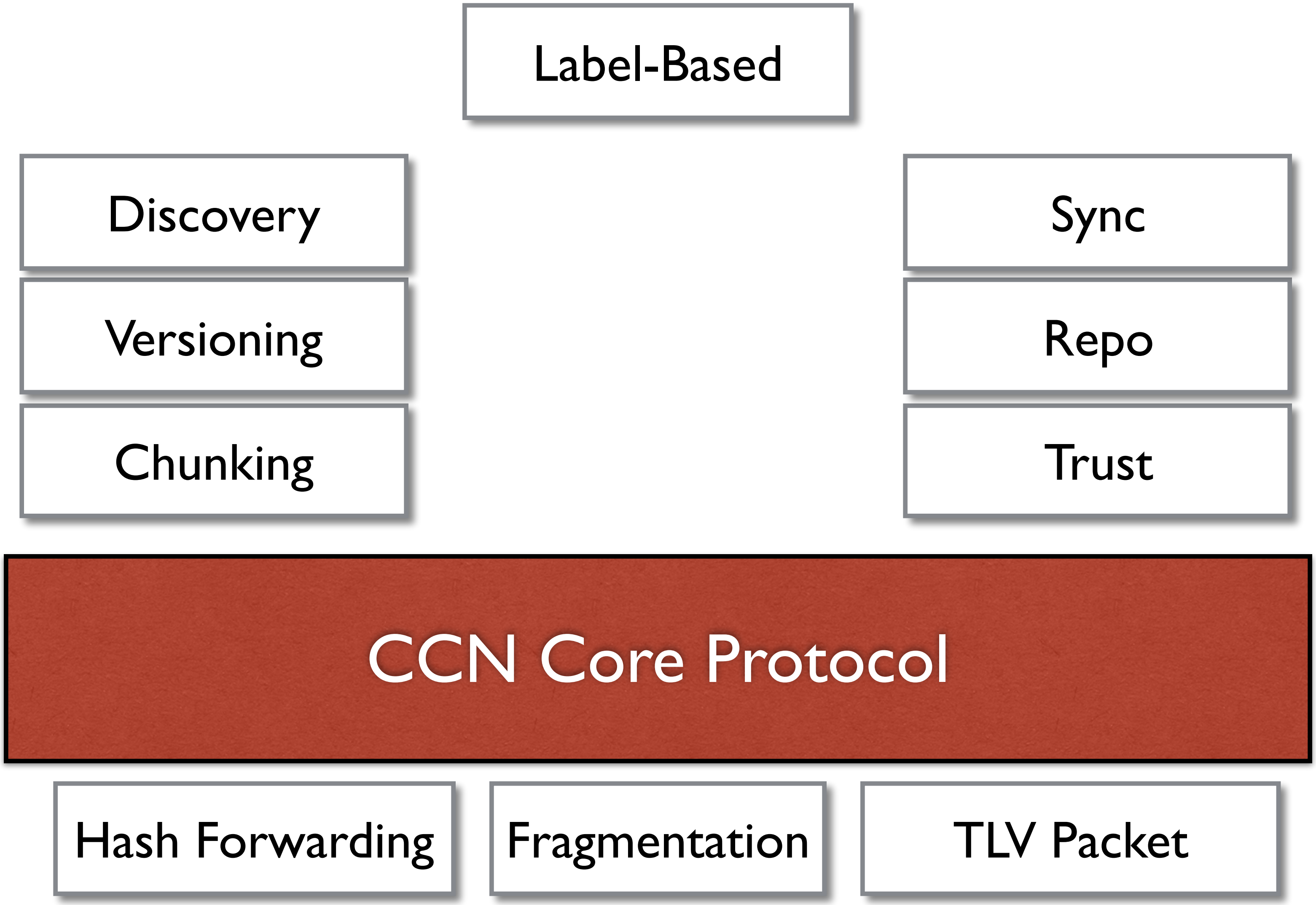
Changes to API

CCNx 0.x

- Individual packets “hand crafted” with CCNB fields.
- The application programmer must understand all details of CCN to write effective programs.

CCNx 1.0

- Interest and Content Object have native type representation with accessor functions.
- Socket API for Interests and Content Objects — this is the “low level” API.
- Key/Value store API
- Message Queue API
- Standard interface to make new APIs
- Well-defined algorithm and security libraries



CCNx 1.0 decouples high-functionality protocols from core protocol

1.0 Protocol Specifications

1. CCNx 1.0 Protocol Specification Roadmap
2. CCNx Semantics
3. TLV Packet Format
4. CCNx Messages in TLV Format
5. Labeled Segment URIs
6. Labeled Content Information URIs for CCNx
7. CCNx Content Object Caching
8. CCNx End-to-end Fragmentation
9. CCNx Content Object Segmentation
10. CCNx Publisher Clock Time Versioning
11. CCNx Publisher Serial Versioning
12. CCNx Selector Based Discovery
13. CCNx Hash Forwarding

CCNx 1.0 is well documented with separation of concerns

parc
A Xerox Company

Codebase Improvements

	LSLOC	2σ Cyclomatic Complexity	2σ Tokens Per Function	Unit Tests	Unit Test Code Coverage
0.x	36,279	23.92	634	0	0%
pre-alpha 1.0	21,047* * not full feature set	5.79	166	1254	69%

New codebase less complex, easier to understand, and tested

As measured with “hfcca”

parc
A Xerox Company

Principal Protocol Changes

CCNx 0.x

- CCNB (binary XML) wire format
- An Interests name matches variable number of name components in Content Object
- An Interest uses an "exclusions" list of terminal-name components to avoid.
- Time is expressed as a binary decimal in 1/4096th of a second.
- Signature algorithm uses an OID string

CCNx 1.0

- Fixed header + TLV format
- An Interest name exactly matches a Content Object name.
- Time is expressed in milli-seconds.
- Content Store must verify a key if user asks for content by KeyId (or not serve it from cache)
- Signature algorithm uses a Cipher Suite (2 bytes)

Content Object satisfaction of Interests

Answers the question

Does this Content Object satisfy this Interest?

Used by the **Forwarder** on the **Fast Path**
Matches against the Pending Interest Table (PIT)
Matches against the Content Store

Interest Similarity

Two Interests are similar

if and only if
they are satisfied by exactly the same set of Content Objects

Used by the **Forwarder** on the **Fast Path**
to aggregate Interests in Pending Interest Table (PIT)

parc
A Xerox Company

0.x Content Object Matching Rules

$$\text{Name}_a \preceq \text{Name}_b \Leftrightarrow \forall\, i \in (1,\ldots,|\text{Name}_a|)\ \exists\, \text{Name}_b[i] \in \text{Name}_b : \text{Name}_a[i] = \text{Name}_b[i]$$

$$(\text{Name}_{\mathcal{I}} \preceq (\text{Name}_{\mathcal{O}} + \text{SHA256}(\mathcal{O}))) \wedge$$
$$(\text{KeyId} \notin \mathcal{I} \vee \text{KeyId}_{\mathcal{I}} = \text{KeyId}_{\mathcal{O}}) \wedge$$
$$(\text{MinSuffix} \notin \mathcal{I} \vee |\text{Name}_{\mathcal{I}}| + \text{MinSuffix}_{\mathcal{I}} \leq |\text{Name}_{O}|) \wedge$$
$$(\text{MaxSuffix} \notin \mathcal{I} \vee |\text{Name}_{\mathcal{O}}| \leq |\text{Name}_{\mathcal{I}}| + \text{MaxSuffix}_{\mathcal{I}}) \wedge$$
$$(\text{Excludes} \notin \mathcal{I} \vee |\text{Name}_{\mathcal{O}} + \text{SHA256}(\mathcal{O})| = |\text{Name}_{I}| \vee$$
$$(\text{Name}_{\mathcal{O}} + \text{SHA256}(\mathcal{O}))[|\text{Name}_{\mathcal{I}}| + 1] \in \text{Excludes}(\mathcal{I}))$$

Complex rules

Forwarder must approximate "Similar To" so
PIT matching requires an entry for each variation
Forwarder must iterate over all Interests whose name
is a prefix of the Content Object and test predicate

1.0 Content Object Matching Rules

$$(\text{Name}_{\mathcal{I}} = \text{Name}_{\mathcal{O}}) \wedge$$
$$(\text{KeyId} \notin \mathcal{I} \vee \text{KeyId}_{\mathcal{I}} = \text{KeyId}_{\mathcal{O}}) \wedge$$
$$(\text{ContentHash} \notin \mathcal{I} \vee \text{ContentHash}_{\mathcal{I}} = \text{SHA256}(\mathcal{O}))$$

Simple rules

There is a simple Interest cover rule to determine Similarity exactly. The Forwarder must look <u>at most</u> 3 PIT entries to match a Content Object

parc A Xerox Company

0.x Timestamps

```
int
ccnb_append_timestamp_blob(struct ccn_charbuf *c,
                           enum ccn_marker marker,
                           intmax_t secs, int nsecs)
{
    int i;
    int n;
    uintmax_t ts, tsh;
    int tsl;
    unsigned char *p;
    if (secs <= 0 || nsecs < 0 || nsecs > 999999999)
        return(-1);
    /* arithmetic contortions are to avoid overflowing 31 bits */
    tsl = ((int)(secs & 0xf) << 12) + ((nsecs / 5 * 8 + 195312) / 390625);
    tsh = (secs >> 4) + (tsl >> 16);
    tsl &= 0xffff;
    n = 2;
    for (ts = tsh; n < 7 && ts != 0; ts >>= 8)
        n++;
    ccn_charbuf_append_tt(c, n + (marker >= 0), CCN_BLOB);
    if (marker >= 0)
        ccn_charbuf_append_value(c, marker, 1);
    p = ccn_charbuf_reserve(c, n);
    if (p == NULL)
        return(-1);
    for (i = 0; i < n - 2; i++)
        p[i] = tsh >> (8 * (n - 3 - i));
    for (i = n - 2; i < n; i++)
        p[i] = tsl >> (8 * (n - 1 - i));
    c->length += n;
    return(0);
}
```

1.0 Timestamps*

```
void
tlvEncoder(TLVEncoder *encoder, struct timeval timestamp)
{
   tlvEncoder_WriteUint16(encoder, T_TIMESTAMP);
   tlvEncoder_WriteUint16(encoder, 8);

   uint64_t millis = timestamp->tv_sec * 1000 + timestamp->tv_usec / 1000;
   tlvEncoder_WriteUint64(encoder, millis);
}
```

* Not actual code, which is based on flexible schema encoding

Conclusion

CCNx 1.0 Significantly

Simplifies Forwarder behavior

Improves the codebase

Clarifies and broadens the APIs