Secure Fragmentation for Content Centric Networking

Christopher A. Wood
Palo Alto Reseach Center
cwood@parc.com

Marc Mosko
Palo Alto Reseach Center
mmosko@parc.com

IEEE CCN 2015
Dallas, TX, USA
10/19/2015

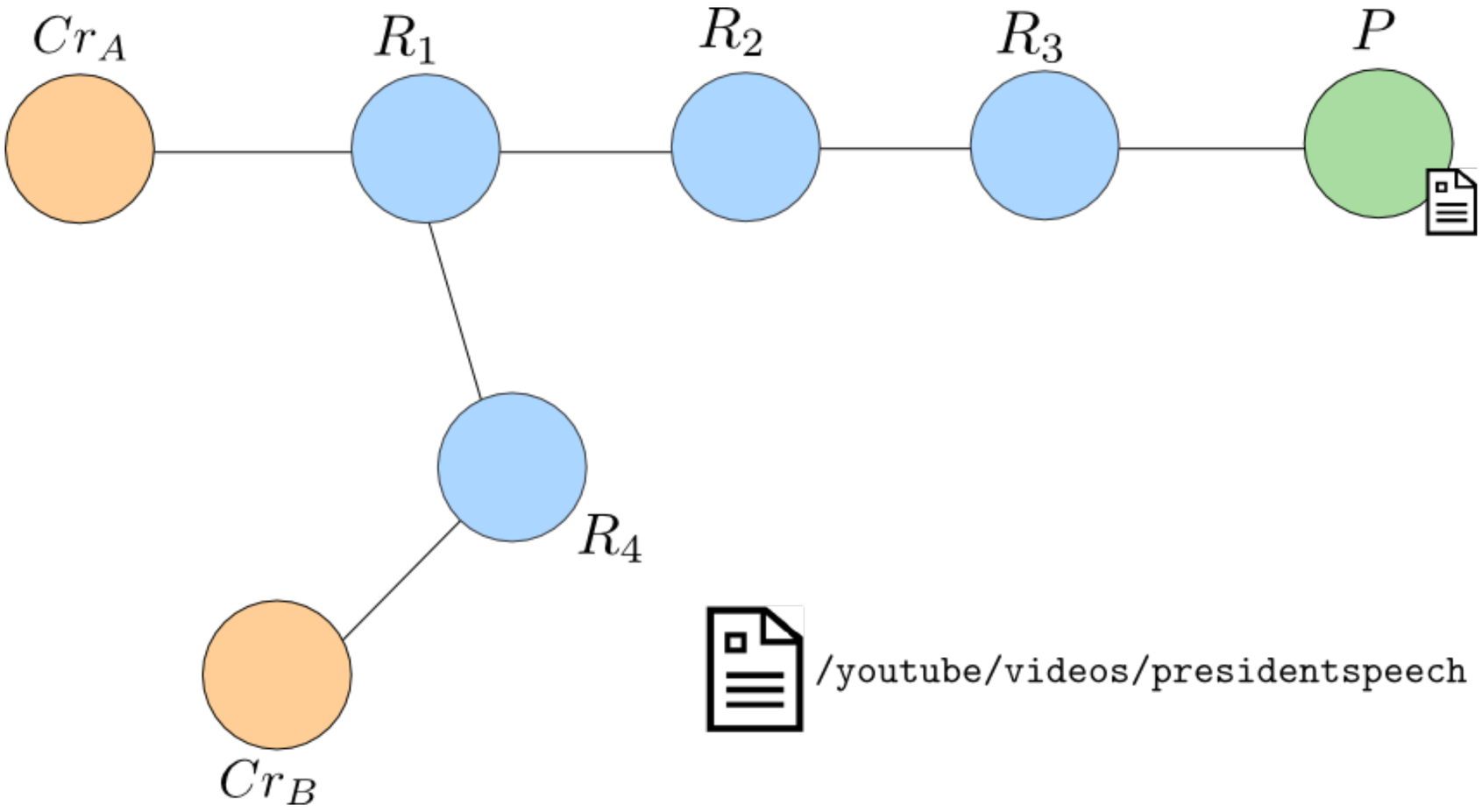
Agenda

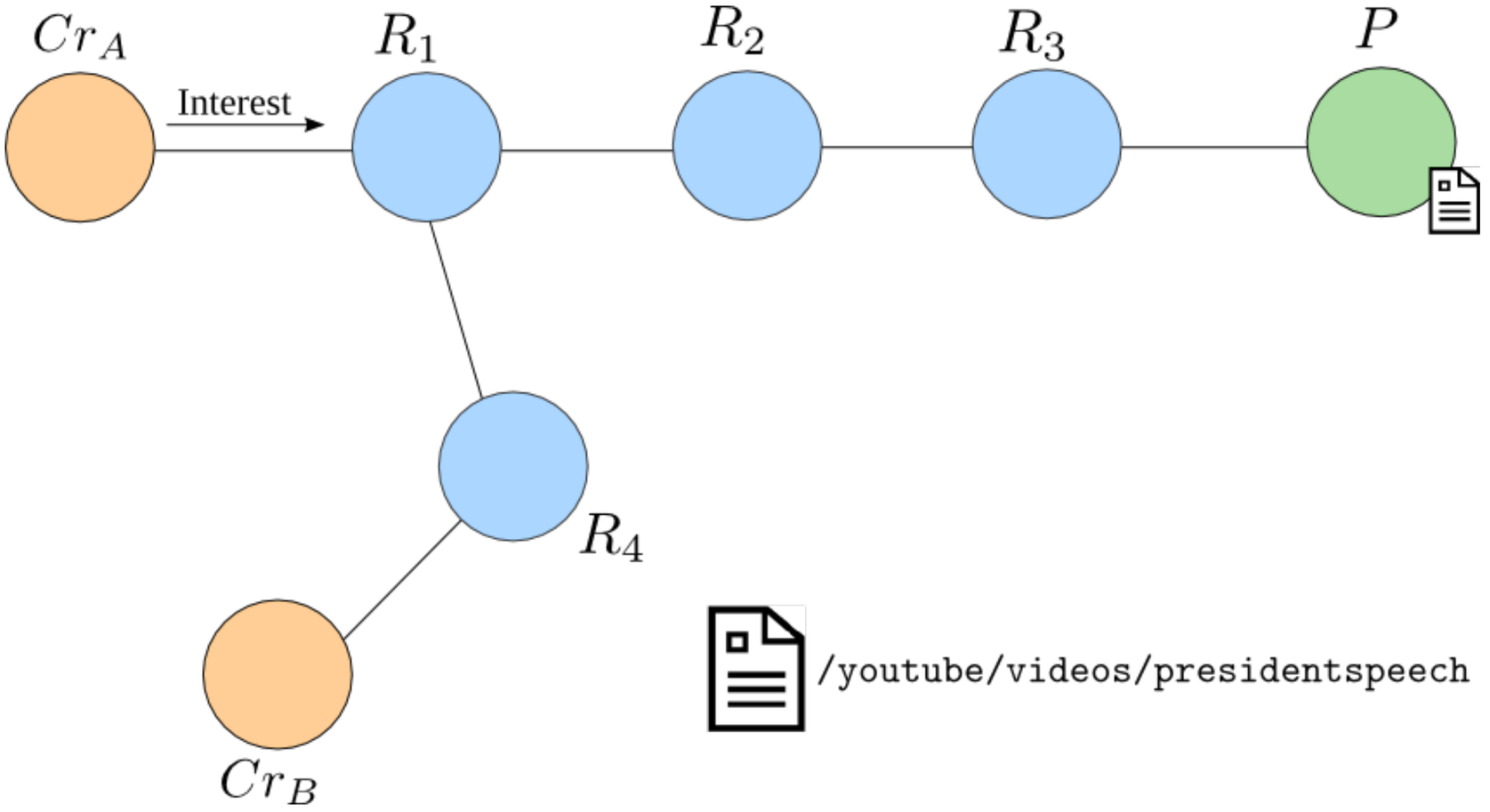
1. CCNx overview
2. Fragmentation and segmentation
3. CCNx fragmentation options
4. Named Network Fragments
5. Performance Results
6. Q&A

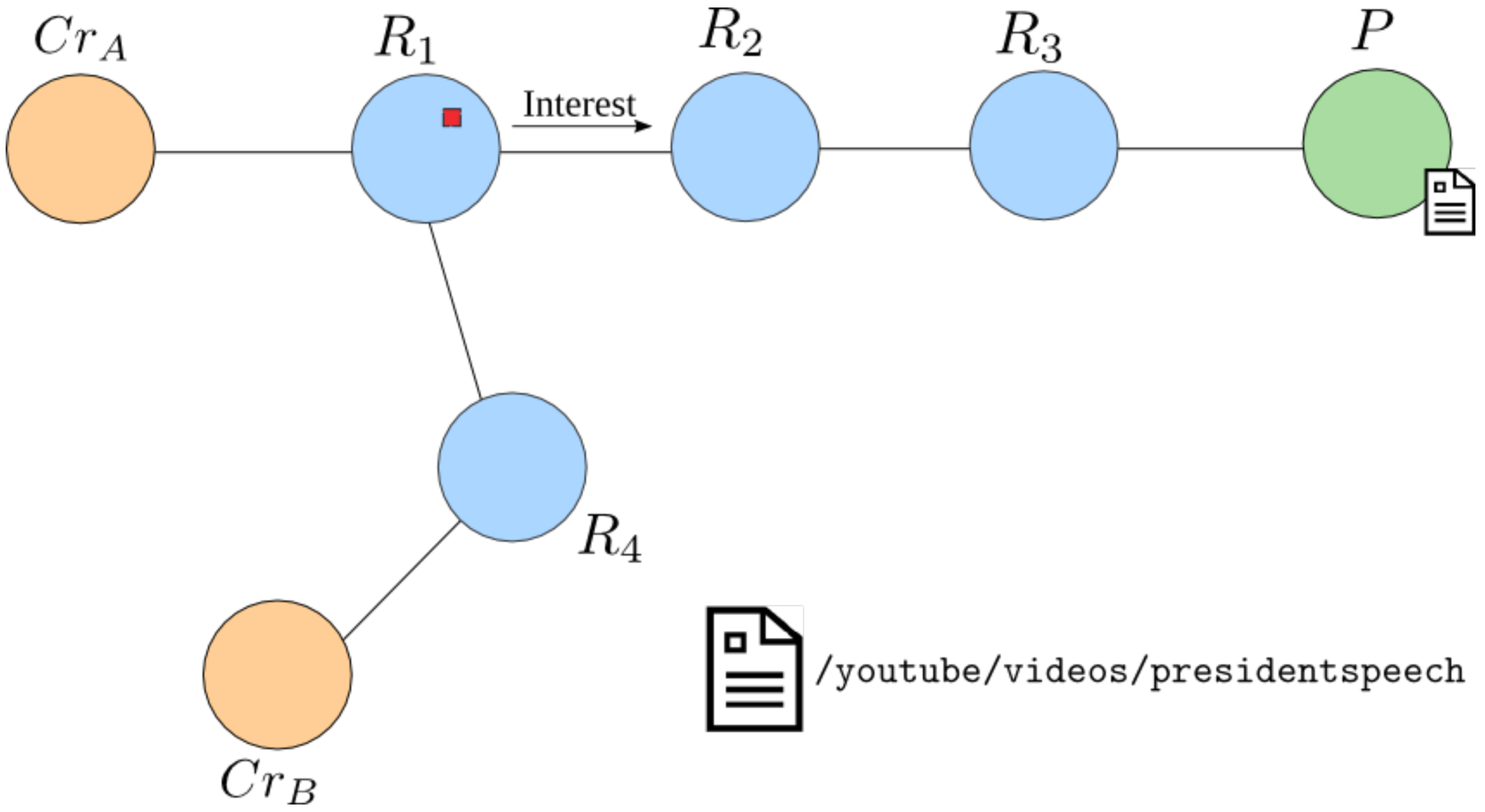
1. CCNx overview

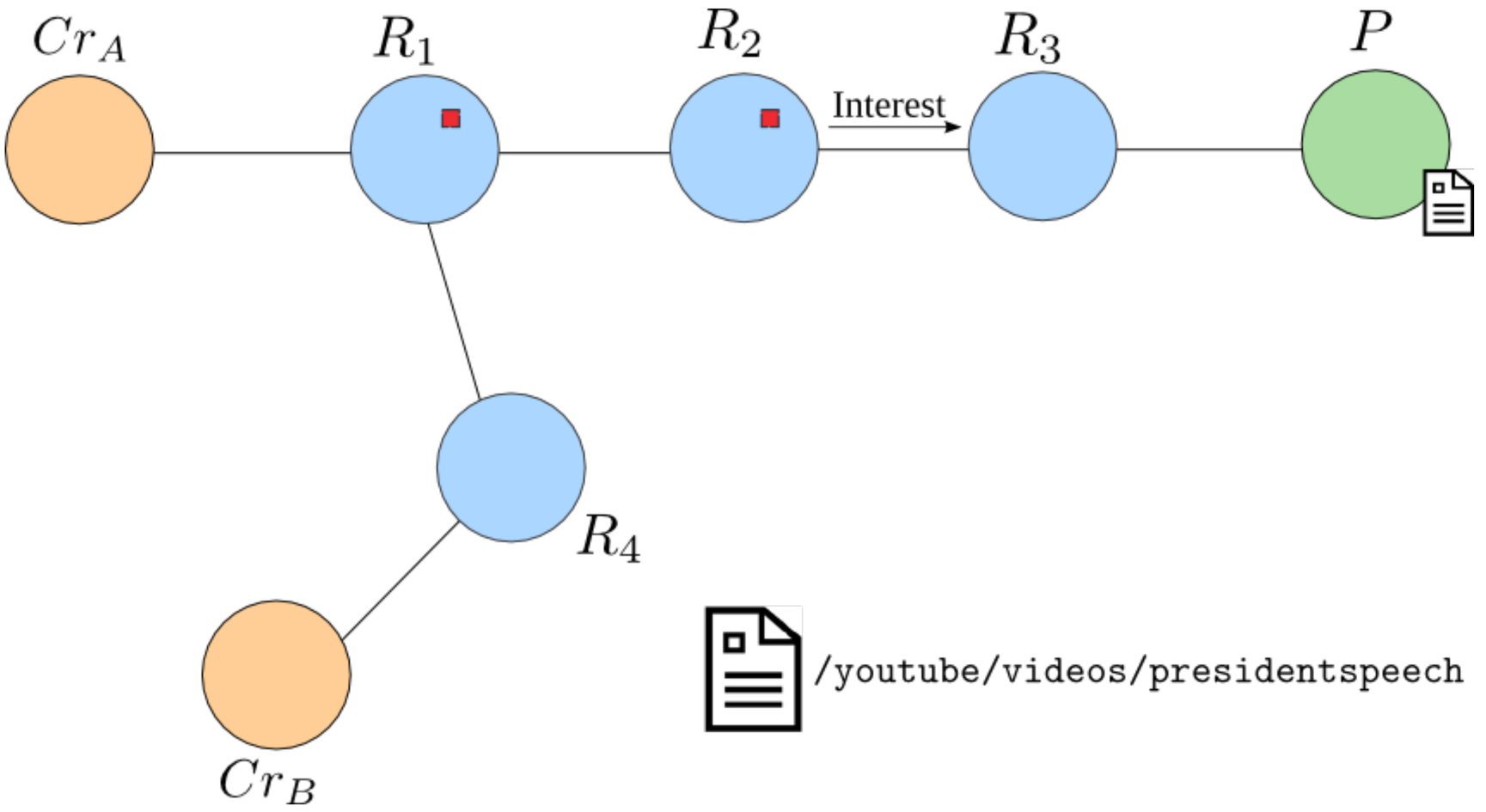
CCNx 101

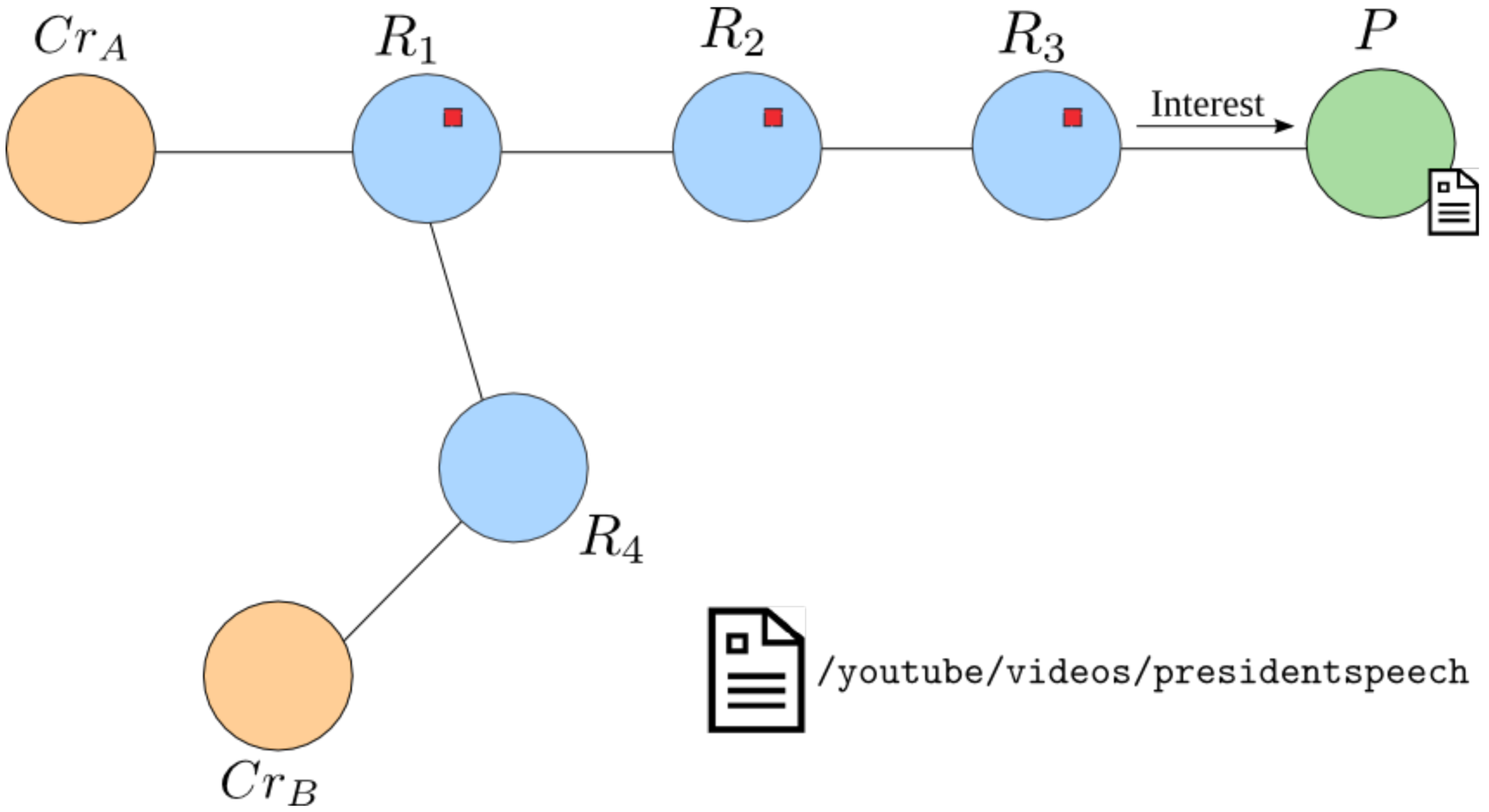
- Content is named and transferred through the network from producers to consumers upon request
- Consumers issue interest packets for content objects
- Forwarders (routers) move interests from consumers to producers
- Forwarder FIBs store forwarding information, Pending Interest Tables (PITs) store interest state, and Content Stores (caches) store previously requested content
- Producers satisfy interests and return the resulting content object

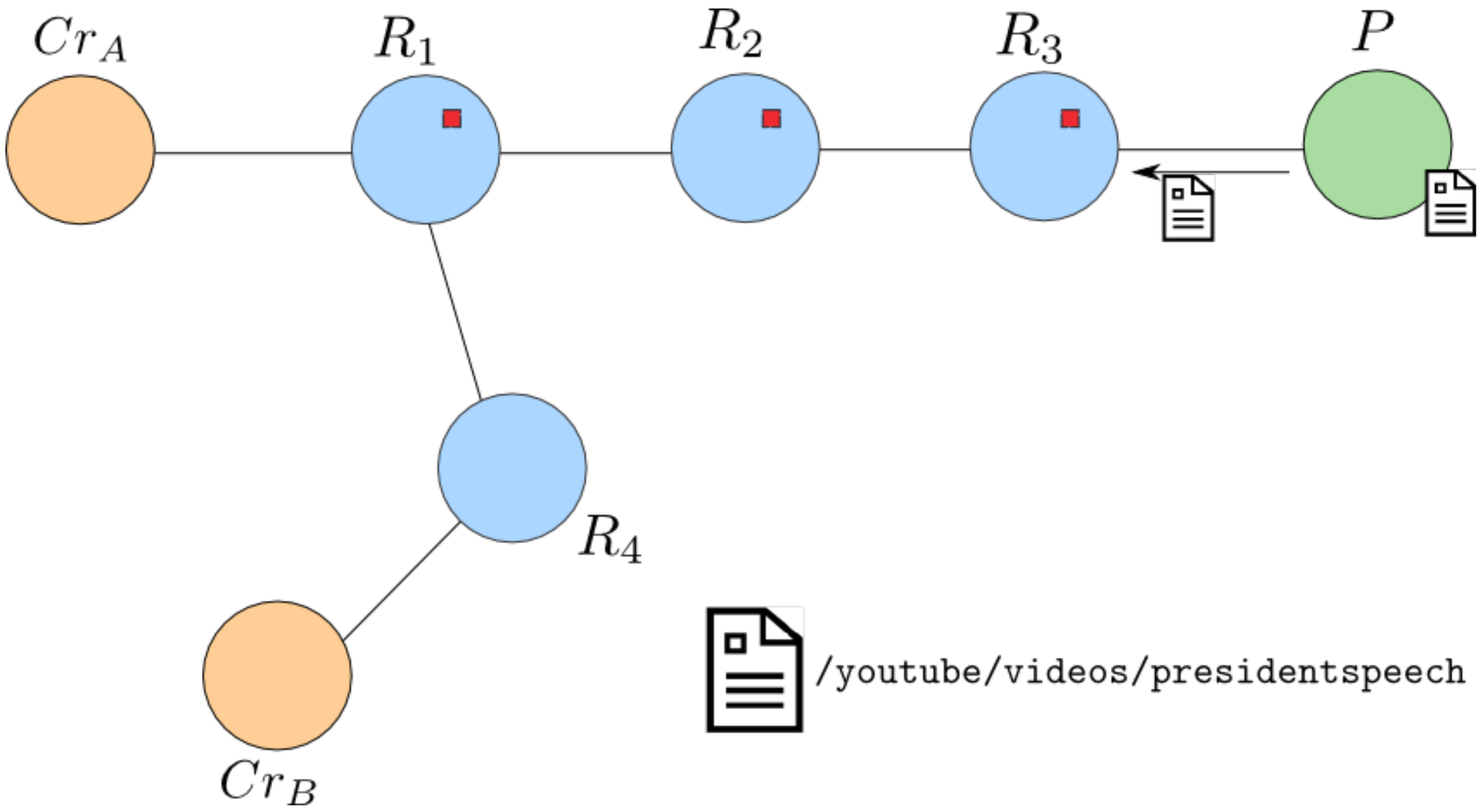


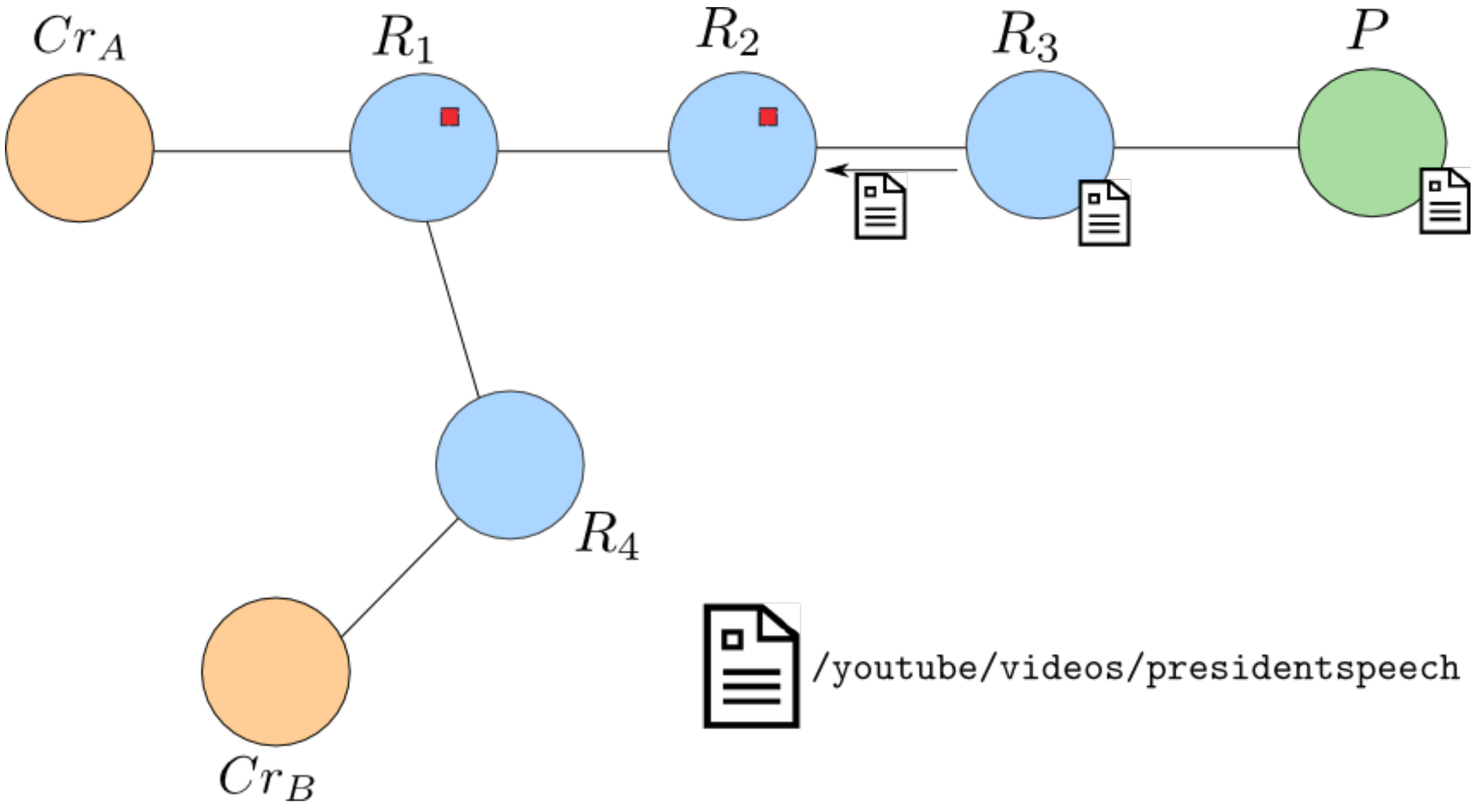


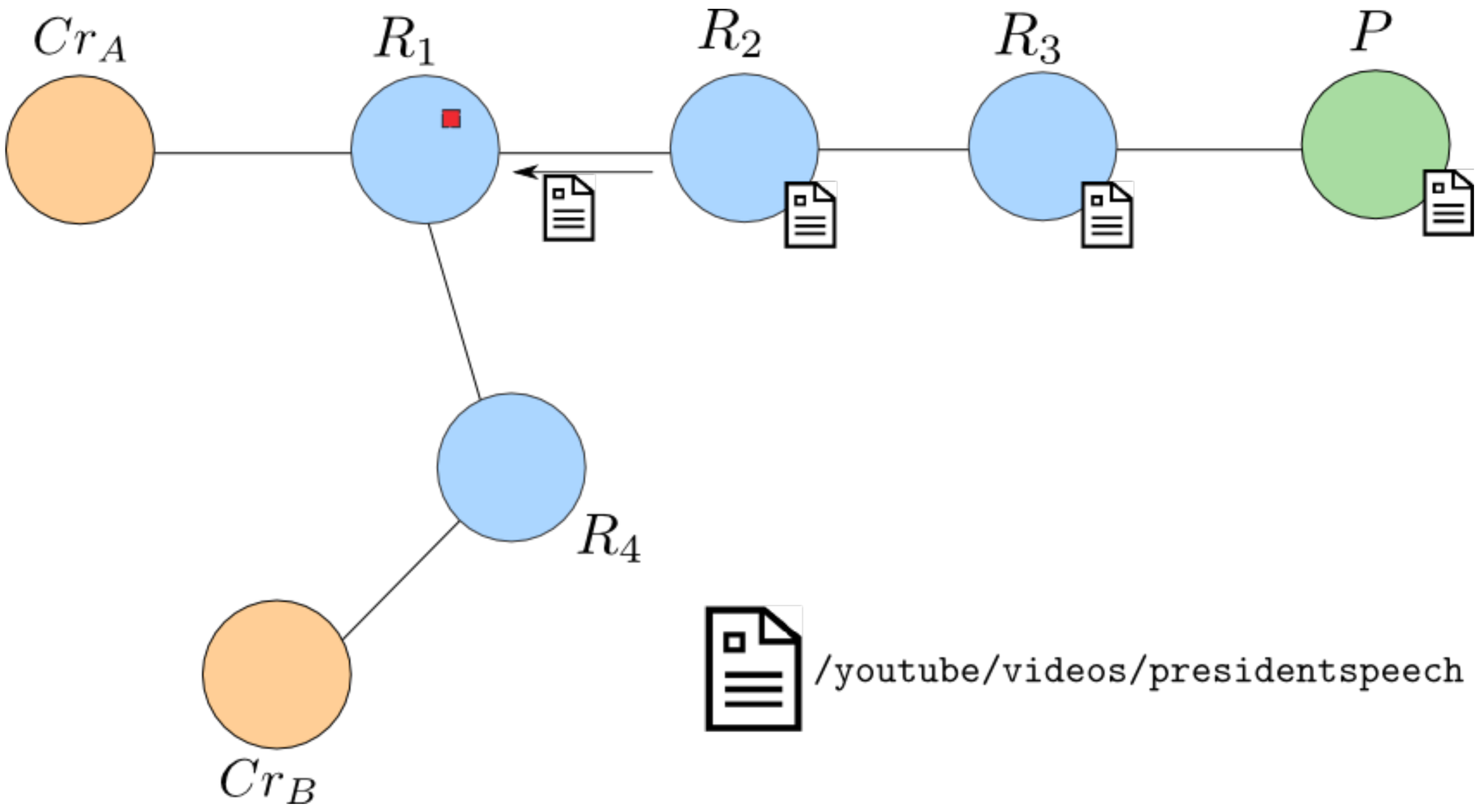


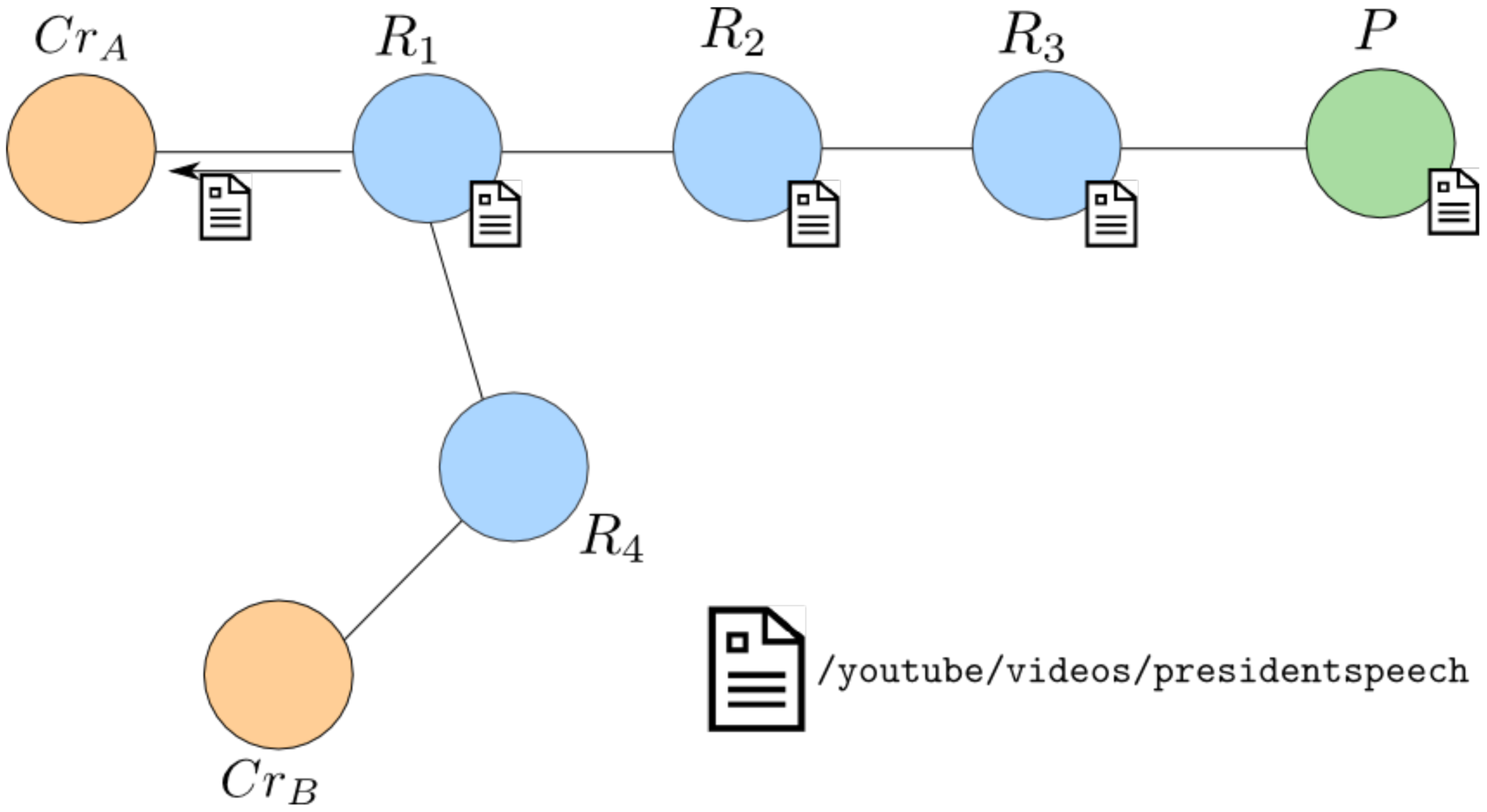


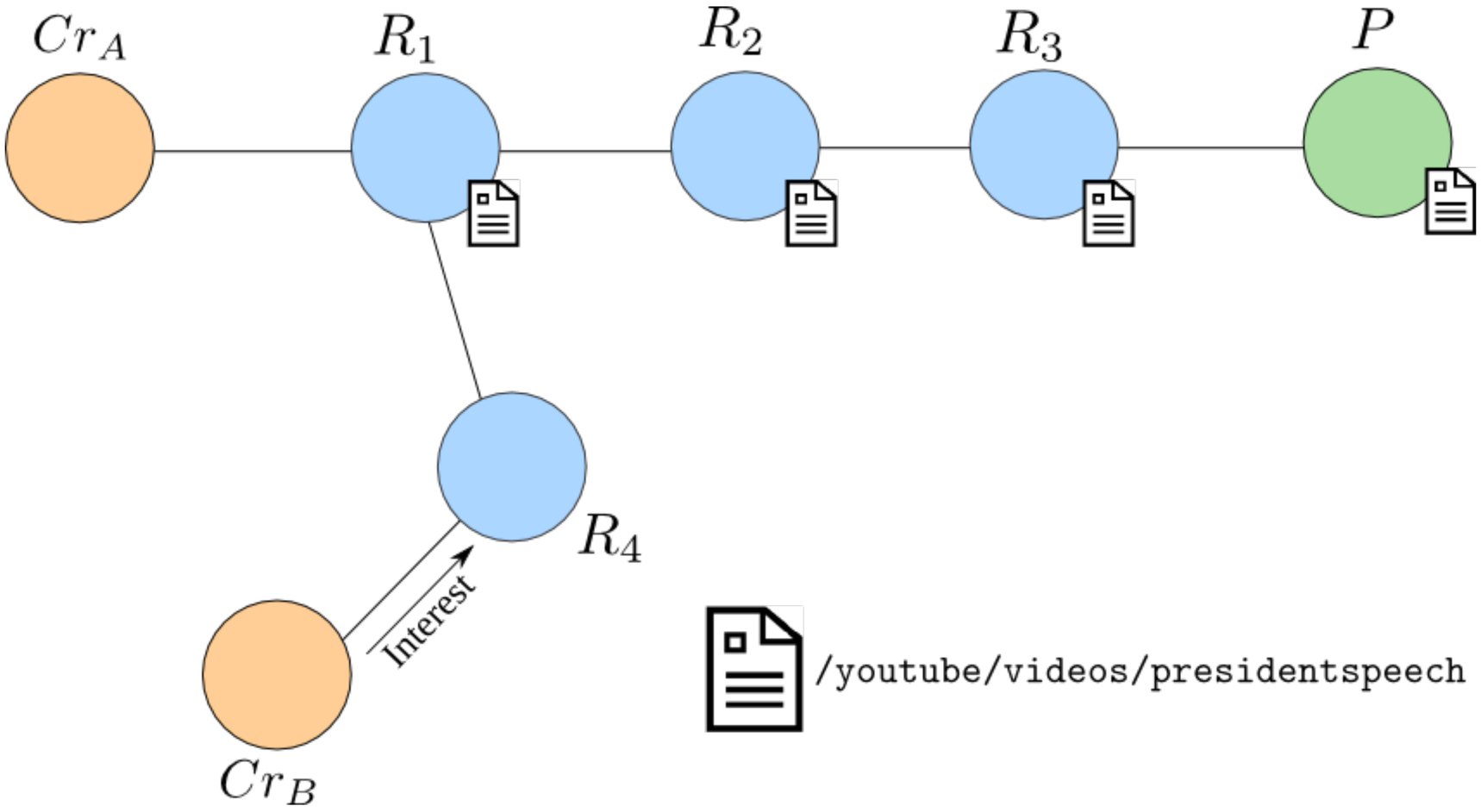


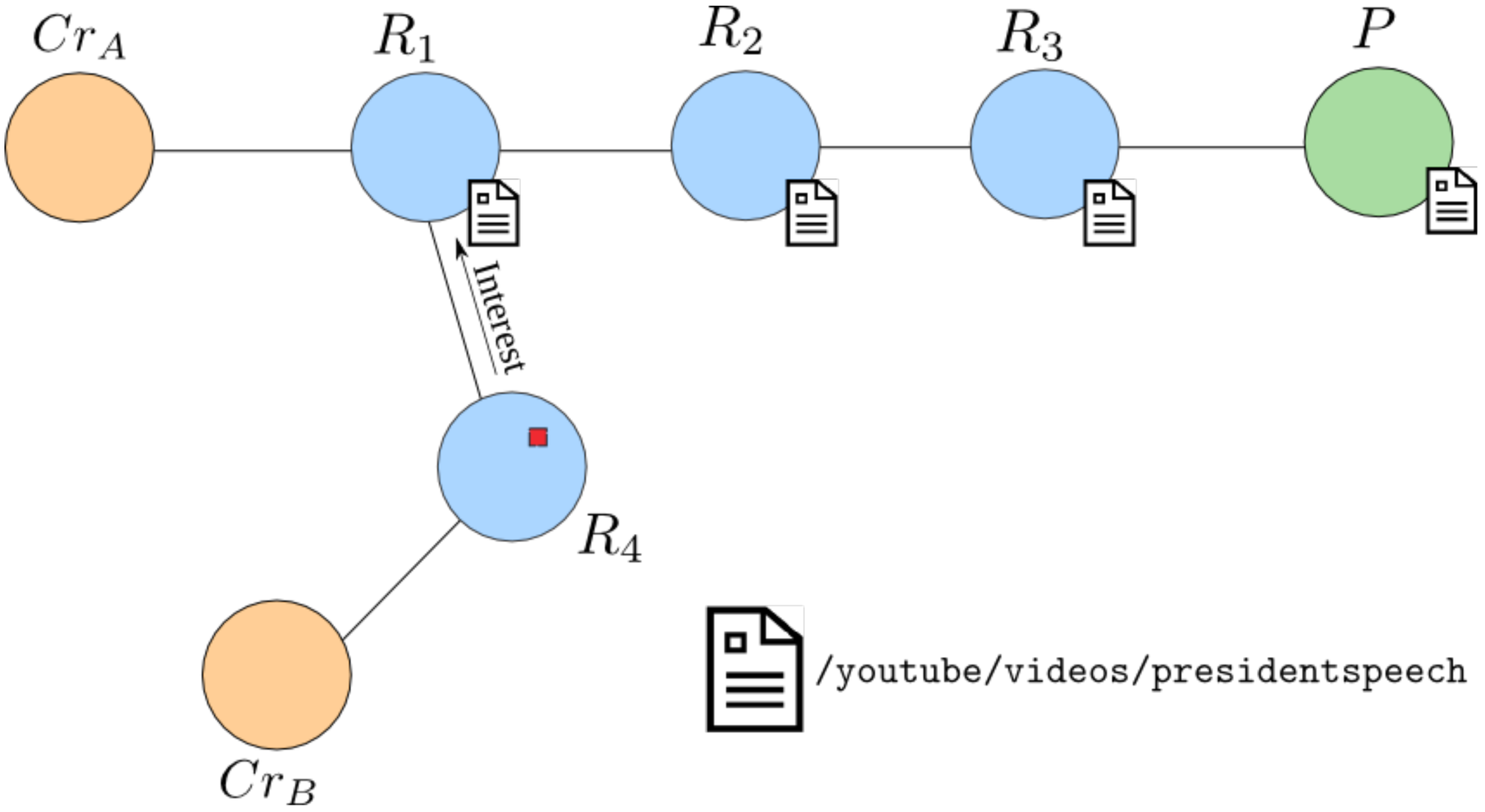












2. Fragmentation and segmentation

Network Links

The Internet connects heterogeneous devices over heterogeneous links with different:

- Physical layers (copper, fiber, radio)
- Link layers (Ethernet, WiFi)
- Maximum Transmission Unit (MTU) sizes

  (determined by link layer)

Fragmentation

Fragmentation: splitting a packet into fragments that fit into an outgoing link MTU

- Fragment header encodes information (e.g., ordering) of related fragments
- Re-fragmentation can occur if smaller MTU is encountered

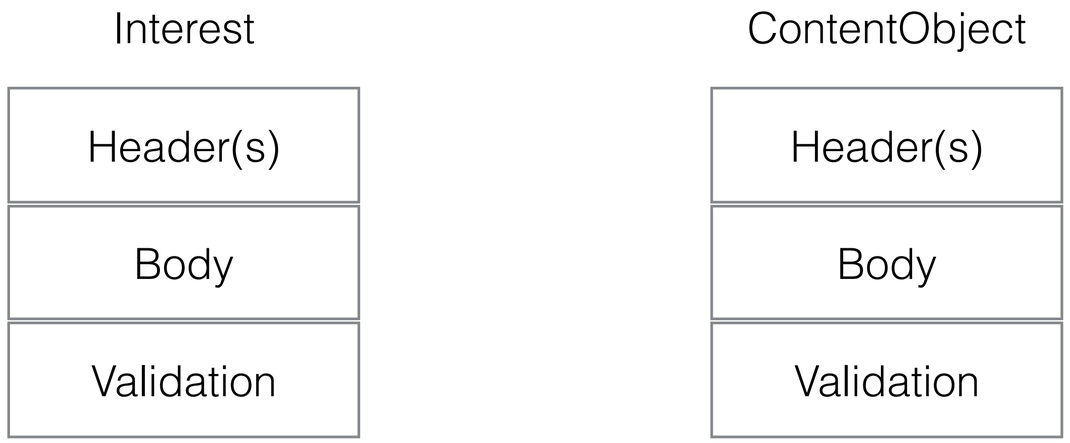
Segmentation

Segmentation: cutting up large pieces of data at the transport (or higher) layer

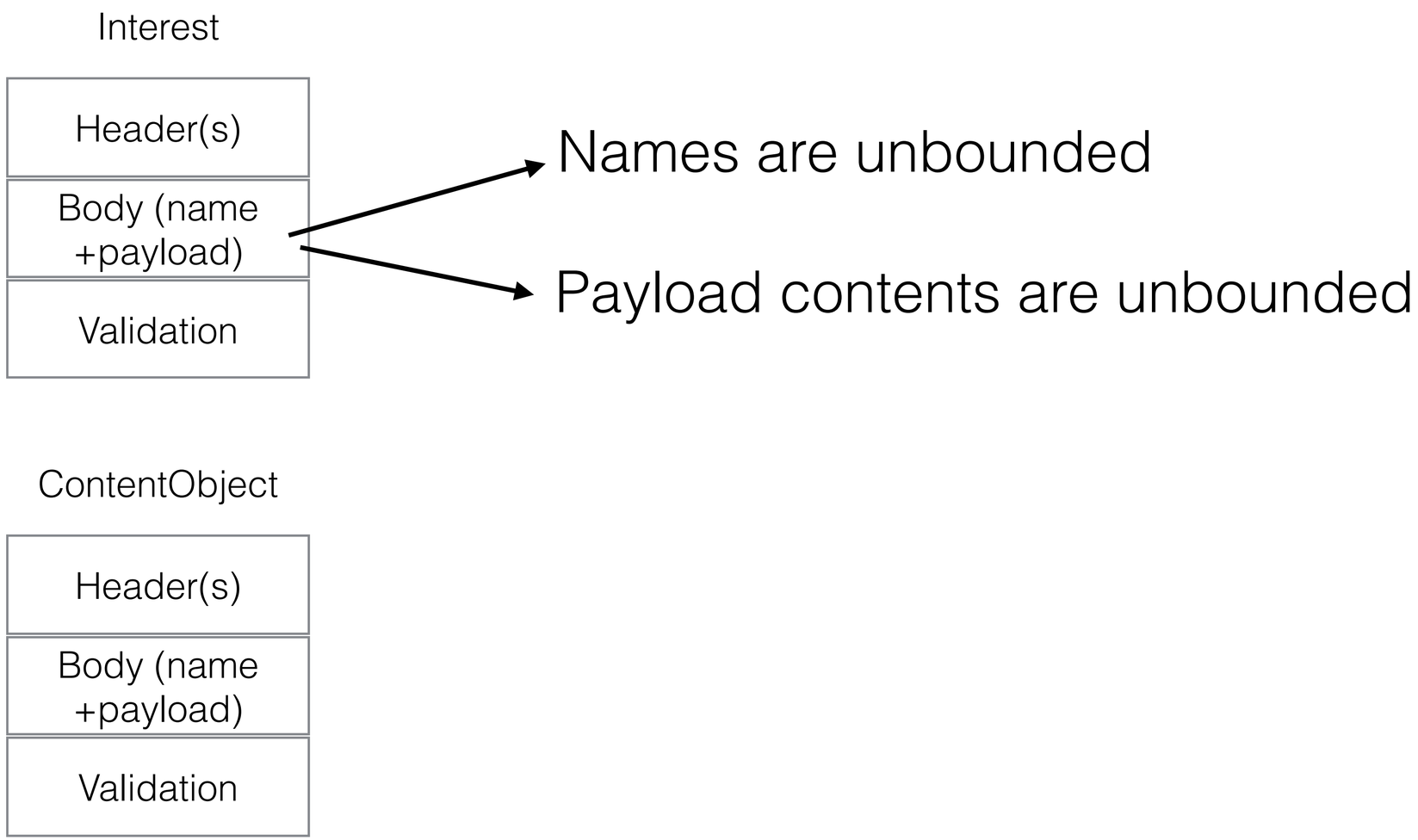
- Segmentation is not fragmentation
- Both may occur for a given consumer-to-producer path

How do fragmentation and segmentation apply to CCNx?

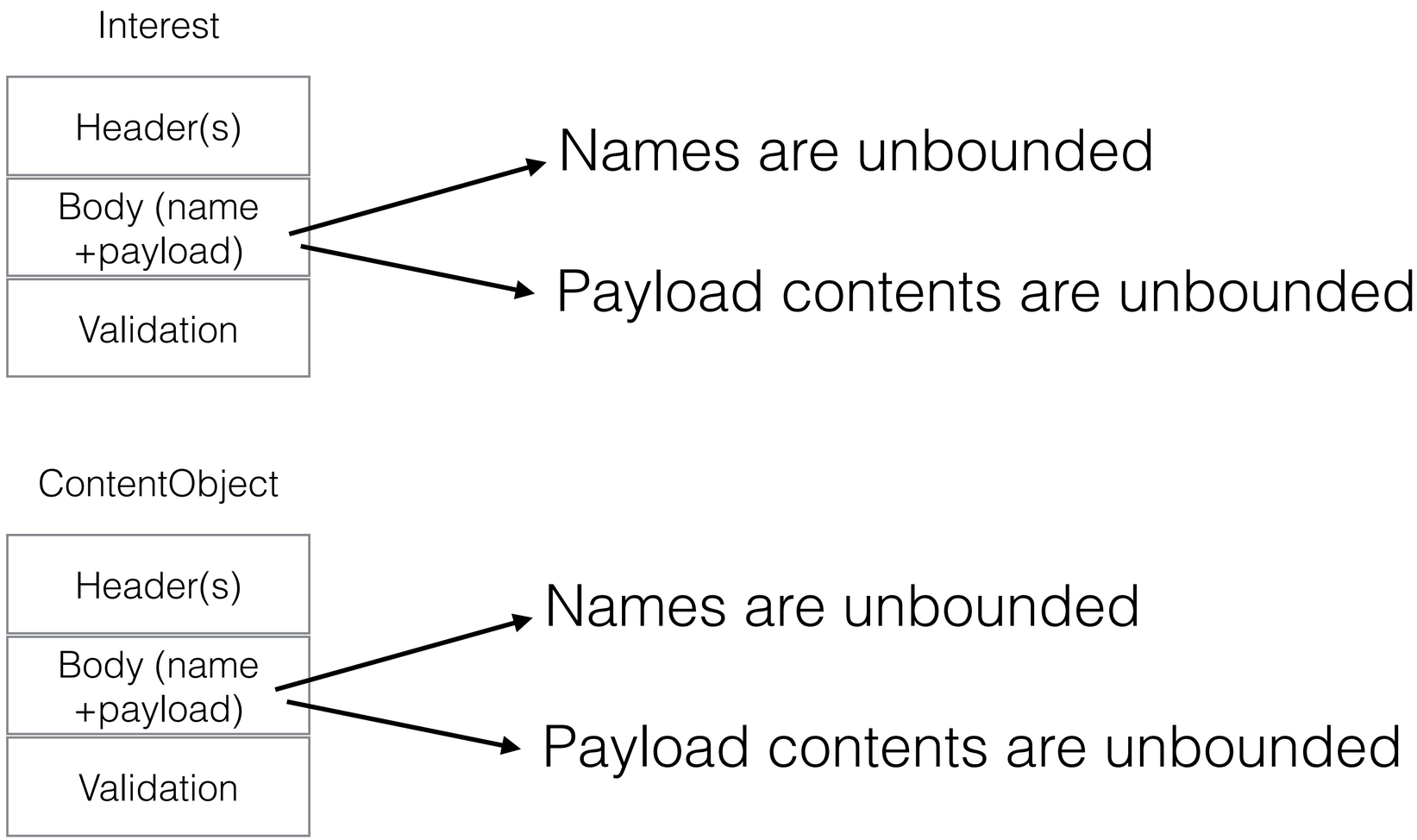
CCNx Messages



CCNx Messages



CCNx Messages



CCNx Segmentation

- CCNx packet fixed header imposes constraints on the message size
  - Names and payloads sizes are bounded
- If the payload for a Content Object (or Interest) exceeds this bound, it must be *segmented*
  - akin to TCP segmentation

CCNx Segmentation Problems

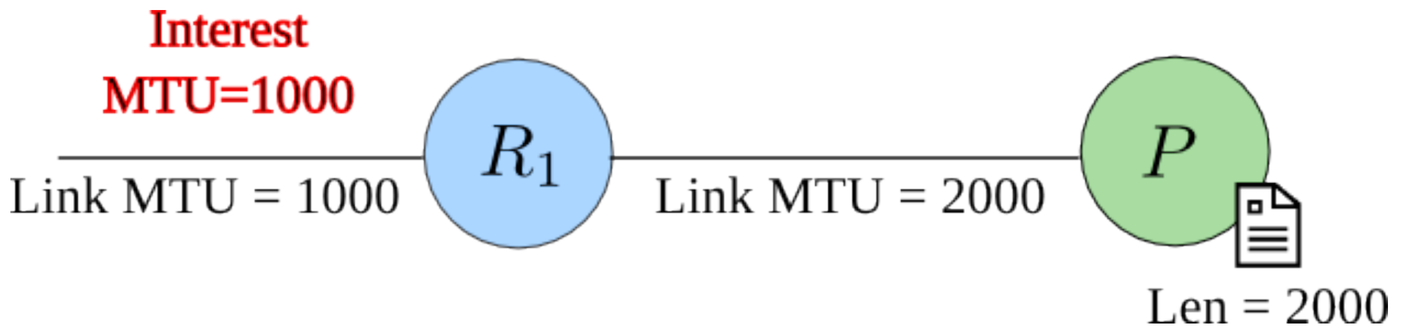
- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)

CCNx Segmentation Problems

- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!

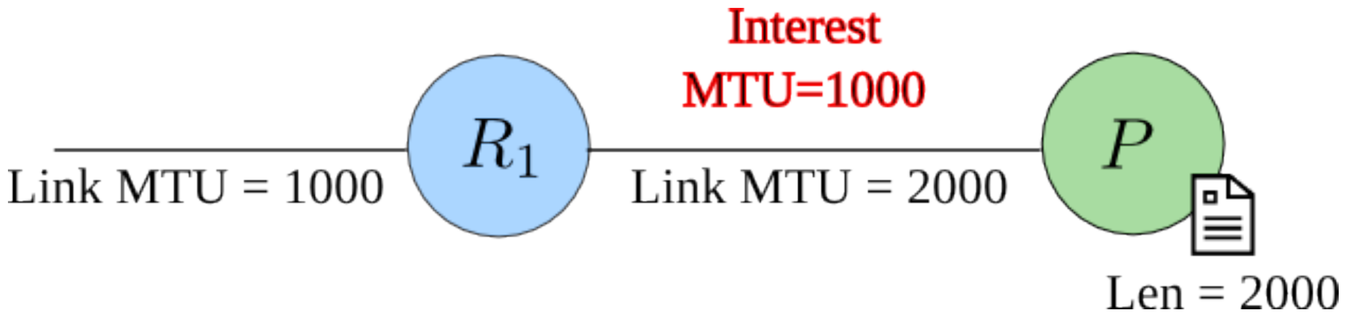
CCNx Segmentation Problems

- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!



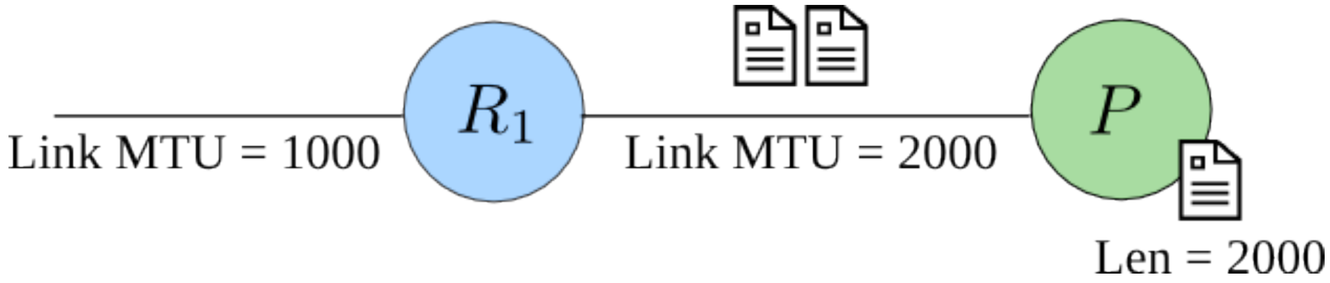
CCNx Segmentation Problems

- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!



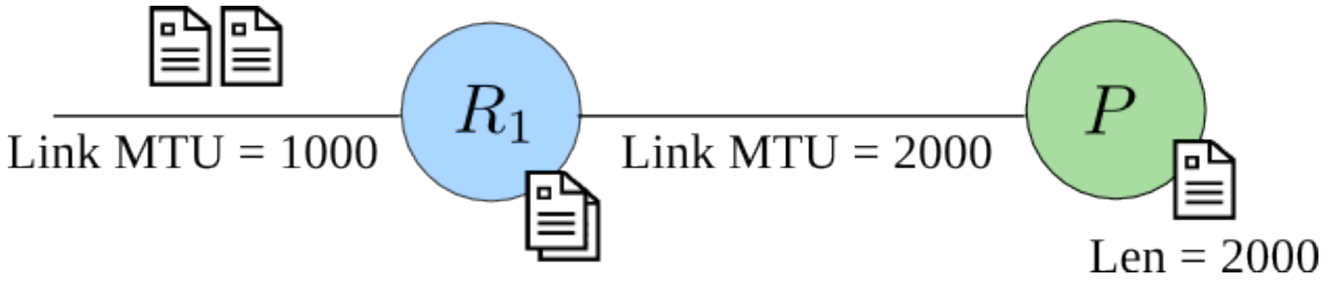
CCNx Segmentation Problems

- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!



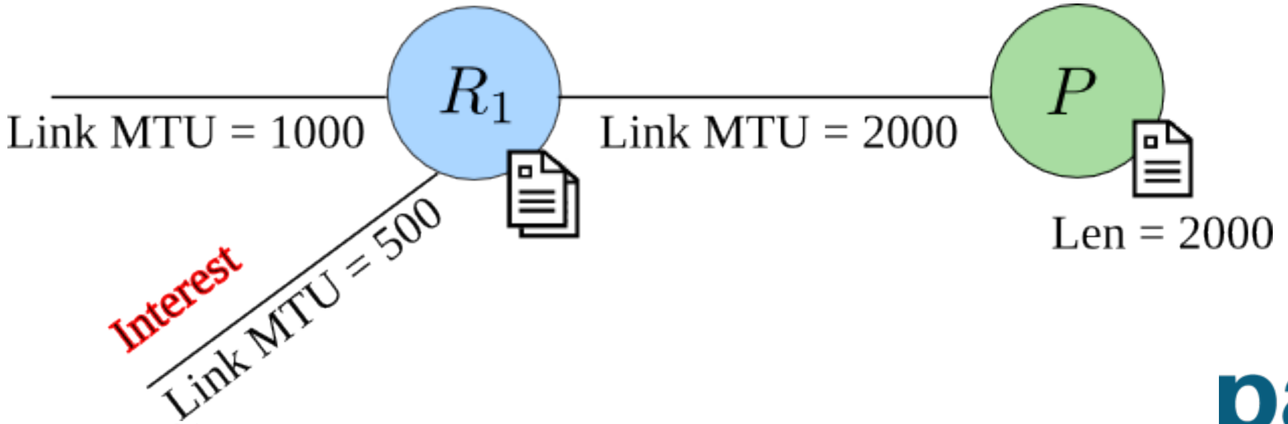
CCNx Segmentation Problems

- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!



CCNx Segmentation Problems

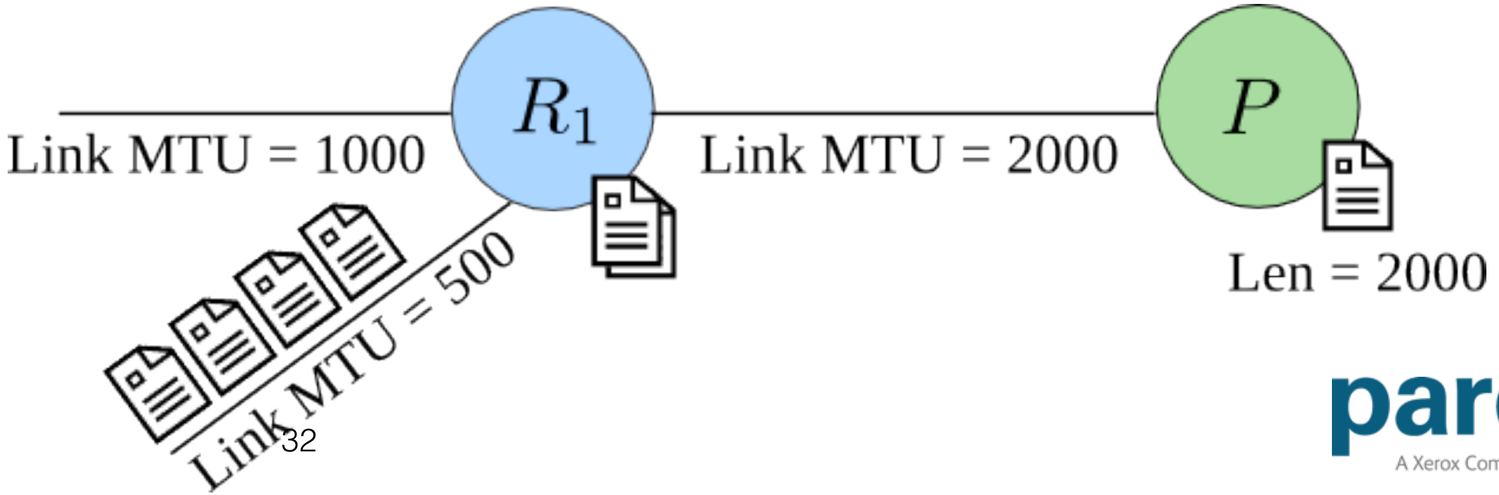
- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!



parc
A Xerox Company

CCNx Segmentation Problems

- Is (Content Object) segmentation a substitute for fragmentation?
  - Maybe, if the minimum path MTU is known
  - No, otherwise (i.e., for Interests)
- Does MTU discovery work?
  - Not all the time!



parc
A Xerox Company

CCNx Segmentation Problems

- Problem #1: routers do not have access to signing keys

CCNx Segmentation Problems

- Problem #1: routers do not have access to signing keys
- Problem #2: producer cannot segment for all MTUs

CCNx Segmentation Problems

- Problem #1: routers do not have access to signing keys
- Problem #2: producer cannot segment for all MTUs
- Problem #3: Interests cannot be segmented since the MTU is not known (among other reasons)

fragmentation is unavoidable

3. CCNx fragmentation options

CCNx Fragmentation Options

There are two flavors of CCNx fragmentation proposals:

- Hop-by-hop (Begin-End Fragmentation) [1]
- Cut-through (FIGOA) [2]

[1] - http://datatracker.ietf.org/doc/draft-mosko-icnrg-beginendfragment/
[2] - C. Ghali, A. Narayanan, D. Oran, G. Tsudik, C. A. Wood, NCA 2015, the 14th IEEE International Symposium on Network Computing and Applications, September 28 - 30, 2015, Cambridge, MA, USA.

Begin-End Fragmentation

- Run between a sender and “peer”
- B, E, and BE flags are used to signal the start, end, and entry of a fragment series
- Fragments are tagged with monotonically increasing sequence numbers
- Idle fragments can be used to advance the fragment number

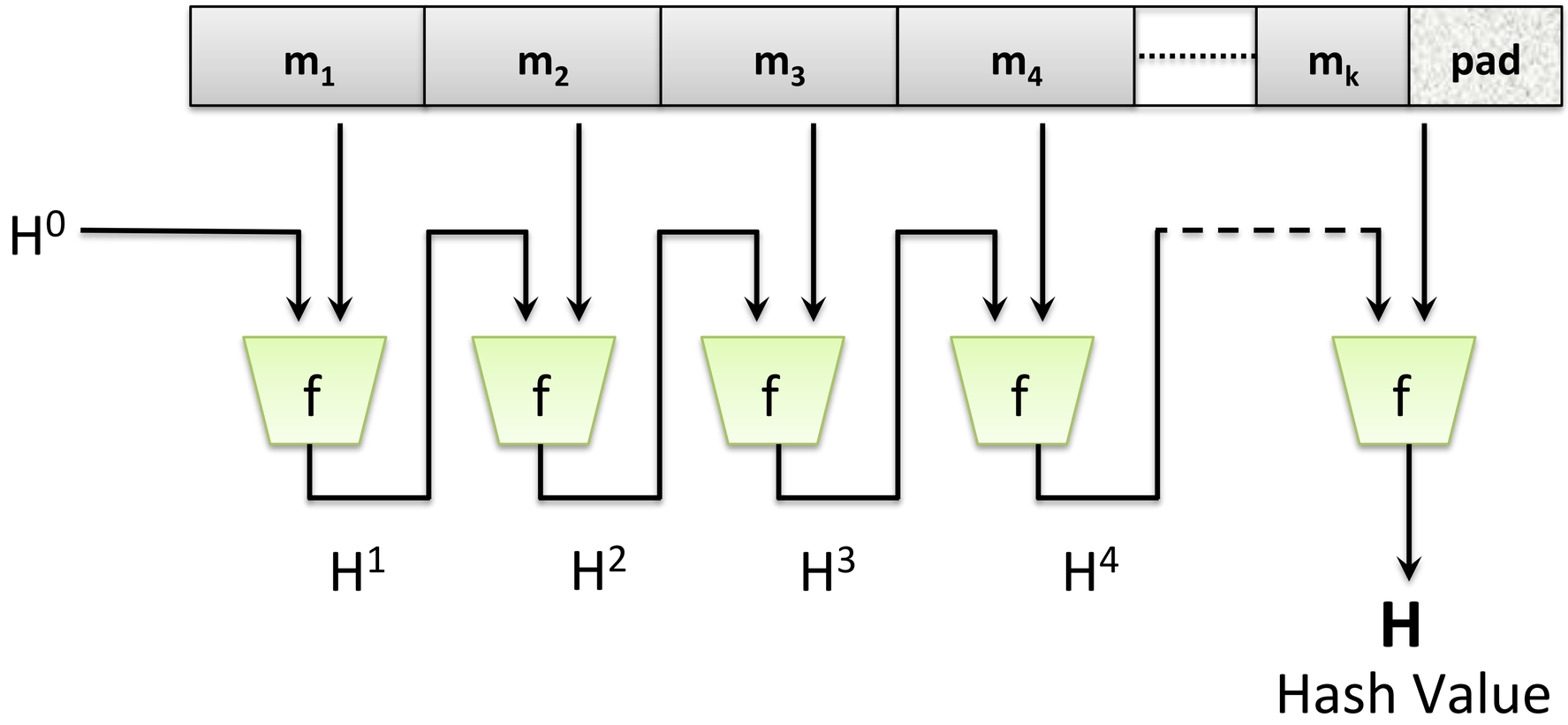
Begin-End Fragmentation Protocol

- Senders:
  - Break up a message into fragments with increasing numbers
  - Mark fragments with B and E bits as needed
- Receivers (peers):
  - Maintain one reassembly queue per sender
  - Gather while in-order fragments are received
  - Reassemble and pass up when end fragment is received
  - Discard the queue when an out-of-order fragment is received

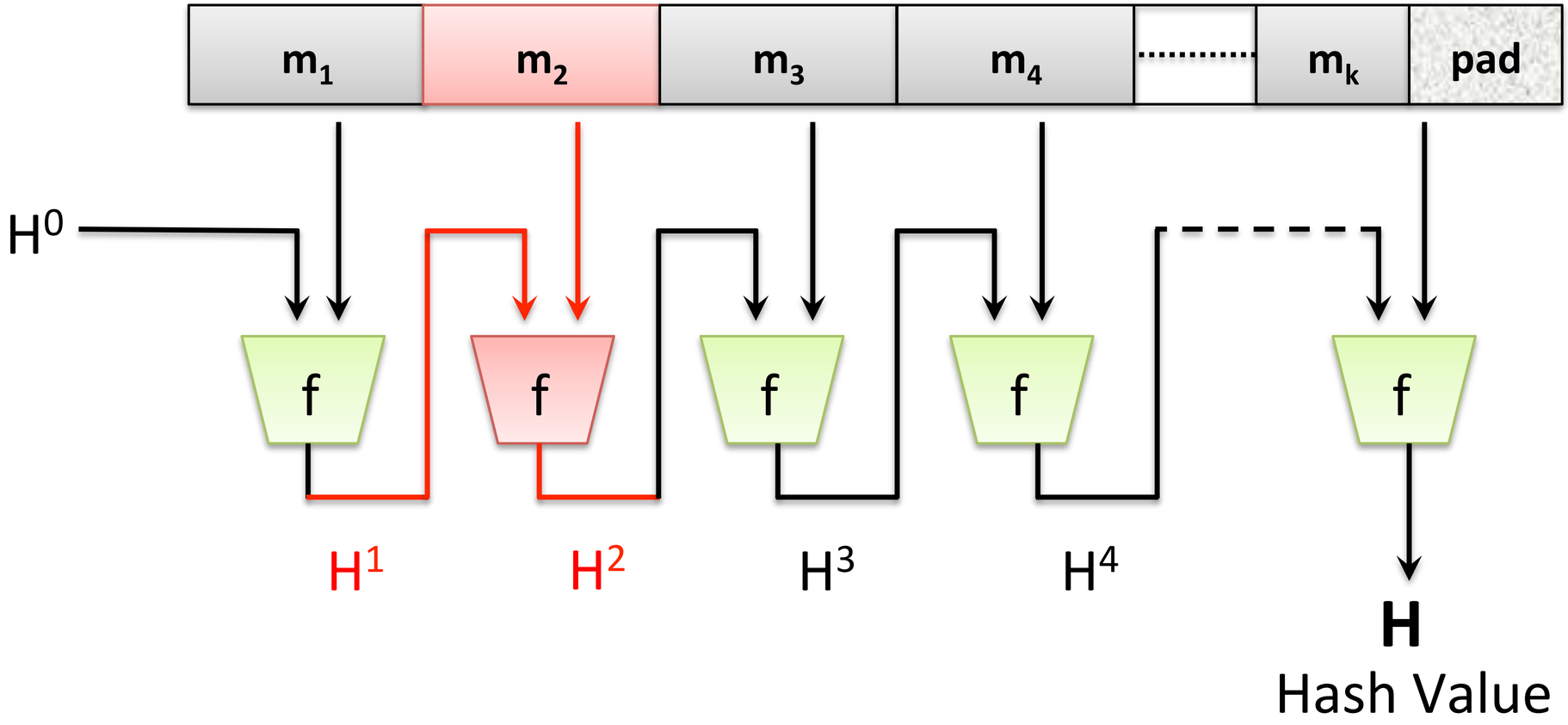
FIGOA Fragmentation

- Based on the concept of *delayed authentication*
- Fragment packets based on MTU size and tag with byte offsets (not indexes)
  - Fragment data size is a multiple of the hash function digest size
- Append the IV or intermediate state of the Merkle-Damgard hash function computation to each fragment (next slide)
- Allows fragments to be re-fragmented if needed

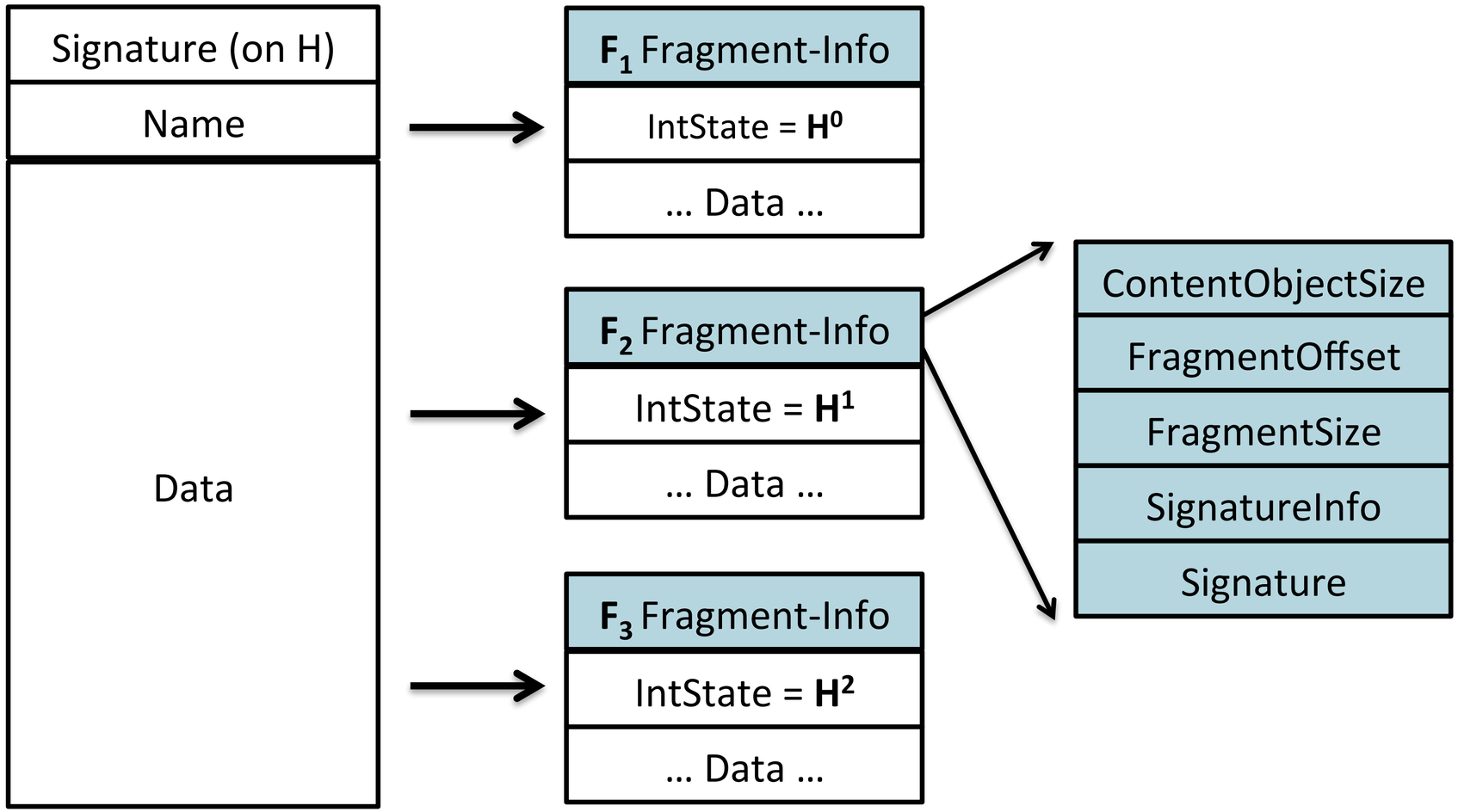
Merkle-Damgard Hash Functions



Merkle-Damgard Hash Functions



FIGOA Fragments



FIGOA Fragmentation Protocol (Sender)

- Fragment message into blocks that are multiples of the hash function digest
- Tag with the byte offset and include the hash function IV or intermediate state (IS)
- Send the fragments…

FIGOA Fragmentation Protocol (Receiver)

- Maintain one reassembly queue per message
- Upon receipt of a fragment without the previous fragment, compute hash and store in queue. If the successor is present, compare the output hash against the successor's IS
- Upon receipt of a fragment with the previous fragment, check against the computed IS, and do the step above.
- Forward all fragments except the last right away.
- Once the full message is received, verify the output digest (if given) and signature, and forward the fragment.

4. Named Network Fragments

FIGOA Shortcomings

- Not possible to match a fragment to a hash-based interest (format does not specify digest in the fragment response)
- Signature verification is deferred to the end “hostage” fragment

Named Network Fragments

NNF improvements over FIGOA:

- Unbounded content length

Named Network Fragments

NNF improvements over FIGOA:

- Unbounded content length
- Immediate signature verification

Named Network Fragments

NNF improvements over FIGOA:

- Immediate signature verification
- Unbounded content length
- Selective retransmission for dropped fragments

Named Network Fragments

NNF improvements over FIGOA:

- Immediate signature verification
- Unbounded content length
- Selective retransmission for dropped fragments
- Hash-based named fragment “chains”

Named Network Fragments

NNF improvements over FIGOA:

- Immediate signature verification
- Unbounded content length
- Selective retransmission for dropped fragments
- Hash-based named fragment “chains”
- Complete ContentObject replacement

NNF Packet Format

```
Fragment := FixedHeader *OptionalHeader NamedFragment
            Payload [ValidationAlg ValidationPayload]
FixedHeader := <as per CCNx 1.0 spec>
OptionalHeader := <as per CCNx 1.0 spec>
NamedFragment := <see right>
Payload := <blocks of original content>
ValidationAlg := <as per CCNx 1.0 spec>
ValidationPayload := <as per CCNx 1.0 spec>
```

```
NamedFragment := (FragmentStart | FragmentData |
                  SegmentStart | SegmentData | SegmentEnd)
                  ChainData
FragmentStart := Name [DigestAlg] OverallLen
                  OverallDigest
FragmentData := [Name] OverallDigest
SegmentStart := Name [DigestAlg] OverallLen SegmentID
SegmentData := [Name] SegmentID
SegmentEnd := [Name] SegmentID OverallDigest
ChainData := PayloadOffset InterState
Name := <as per CCNx 1.0 spec>
OverallLen := Integer
SegmentID := 1*OCTET
OverallDigest := 1*OCTET
DigestAlg := SHA256 / <others>
PayloadOffset := Integer
InterState := 1*OCTET
```

NNF Packet Format

```
Fragment := FixedHeader *OptionalHeader NamedFragment
            Payload [ValidationAlg ValidationPayload]
FixedHeader := <as per CCNx 1.0 spec>
OptionalHeader := <as per CCNx 1.0 spec>
NamedFragment := <see below>
Payload := <blocks of original content>
ValidationAlg := <as per CCNx 1.0 spec>
ValidationPayload := <as per CCNx 1.0 spec>
```

```
NamedFragment := (FragmentStart | FragmentData |
                 SegmentStart | SegmentData | SegmentEnd)
                 ChainData
FragmentStart := Name [DigestAlg] OverallLen
                 OverallDigest
FragmentData := [Name] OverallDigest
SegmentStart := Name [DigestAlg] OverallLen SegmentID
SegmentData := [Name] SegmentID
SegmentEnd := [Name] SegmentID OverallDigest
ChainData := PayloadOffset InterState
Name := <as per CCNx 1.0 spec>
OverallLen := Integer
SegmentID := 1*OCTET
OverallDigest := 1*OCTET
DigestAlg := SHA256 / <others>
PayloadOffset := Integer
InterState := 1*OCTET
```

Specification in progress…

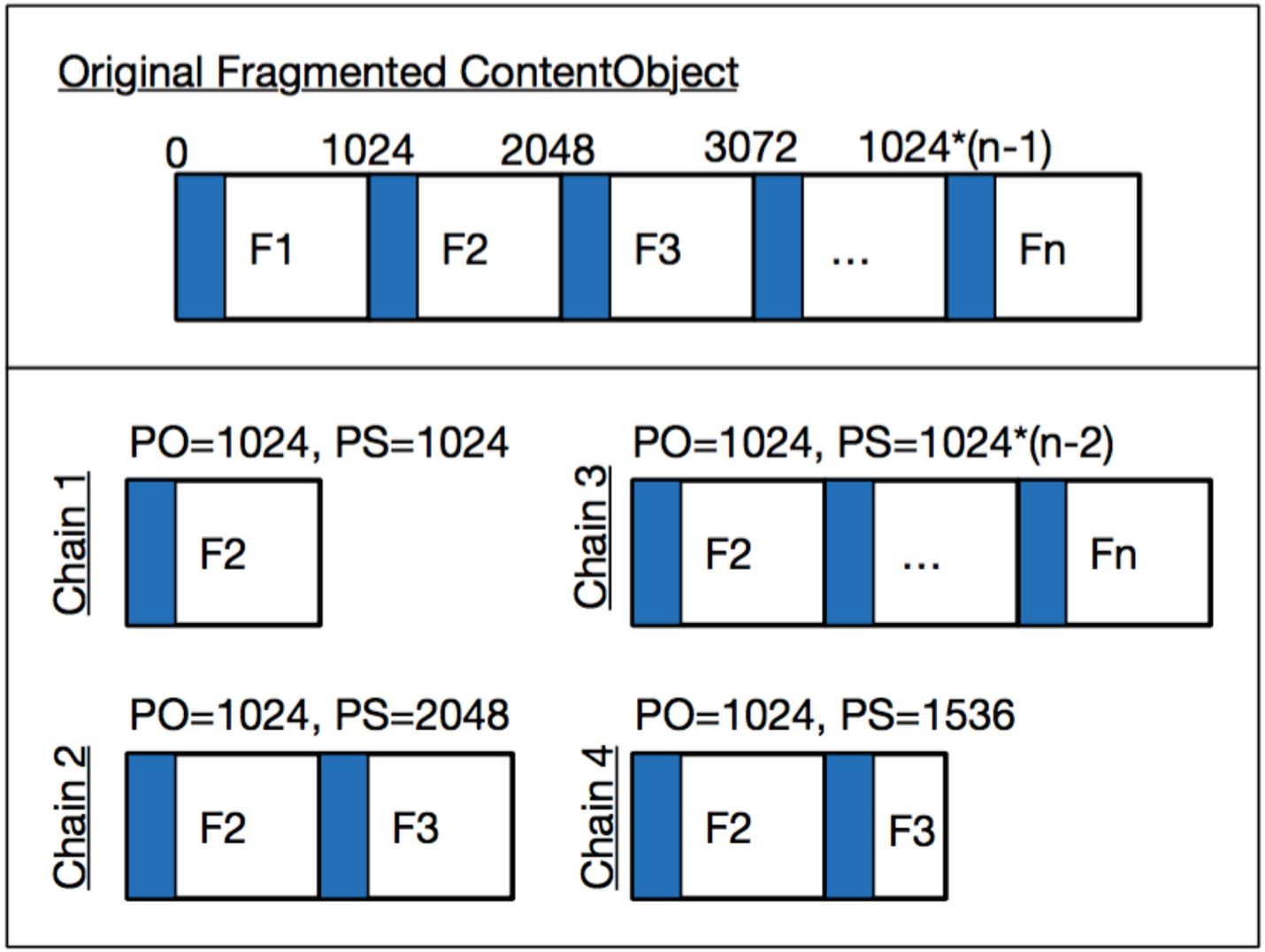
NNF Selective Retransmission

- Link recovery protocols can be used to retransmit dropped or corrupted packets (fragments)
- Selective retransmit is used if (groups of) fragments need to be requested over more than a single hop (link)
- Fragments are uniquely defined by

```
{Name, OverallDigest, PayloadOffset, IntermediateState}
```

- Nodes (routers or consumers) can retransmit

NNF Selective Retransmission



NNF Fragmentation Logic

Similar to FIGOA fragmentation logic, except:

- Only the OverallDigest must be verified upon the arrival of the last fragment
- If valid, the PIT entry is cleared

NNF PIT Logic

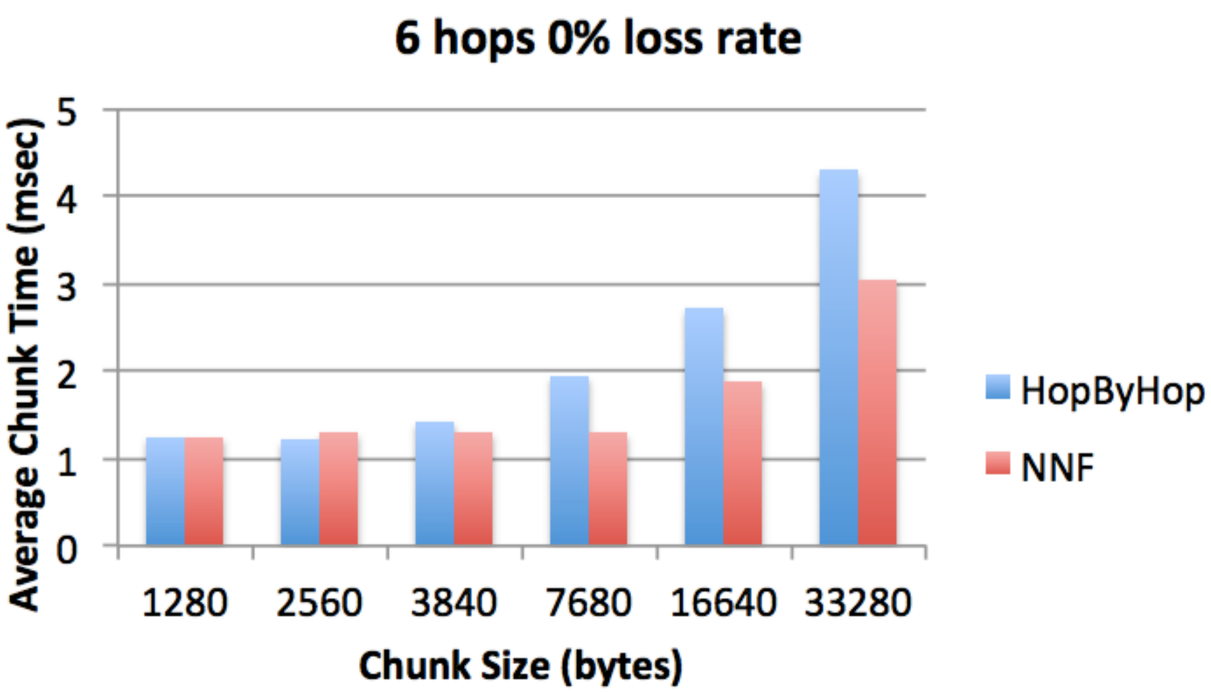
- PIT entries are partially satisfied as fragments arrive
  - PIT lookup needs to take this into account
- See the paper *or upcoming spec* for specific details

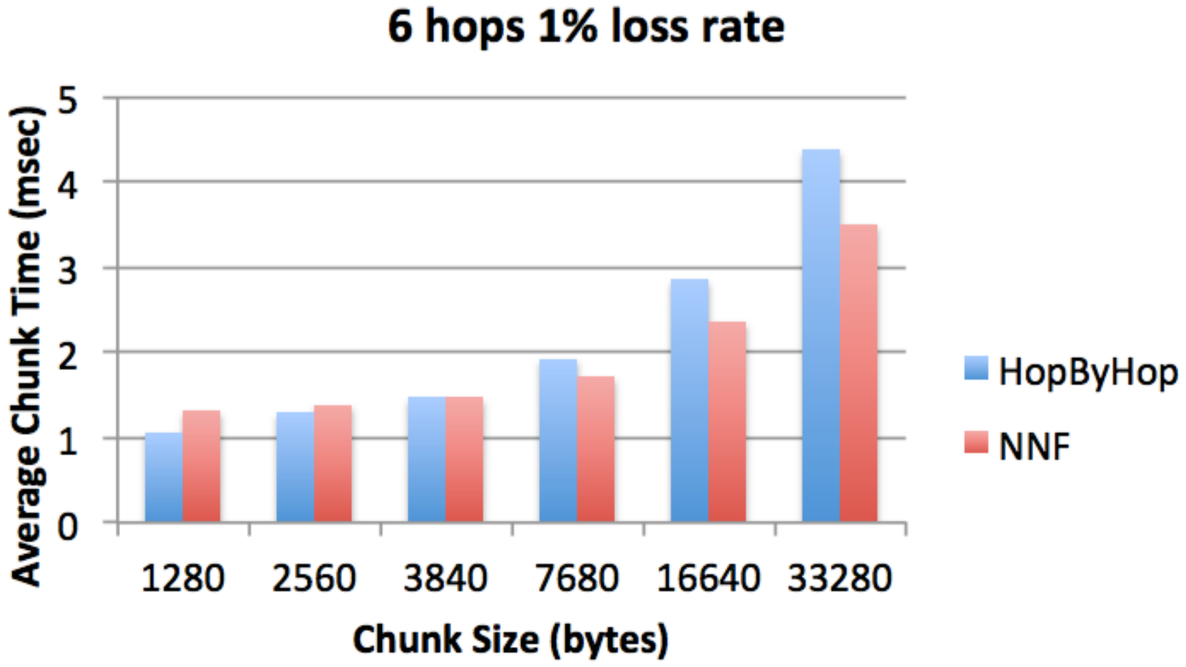
5. Performance Analysis

Experimental Analysis

- Two experiments with a 6-hop topology:
  - 0% loss and 1% loss
- Transfers of a 10MB file chunked (segmented) into 1280, 2560, 3840, 7680, 16640, and 33280 bytes
  - Fragment size must be a multiple of 64 and the implementation chunk size
- Clients transfer chunks serially and measure the latency

Experimental Analysis





Conclusion

- Discussed existing CCNx fragmentation proposals
- Described the new NNF fragmentation protocol
- Displayed some preliminary experimentation results

What's Next?

Write the specification for increased clarity

Questions?…