

ACM ICN 2020 Tutorial:

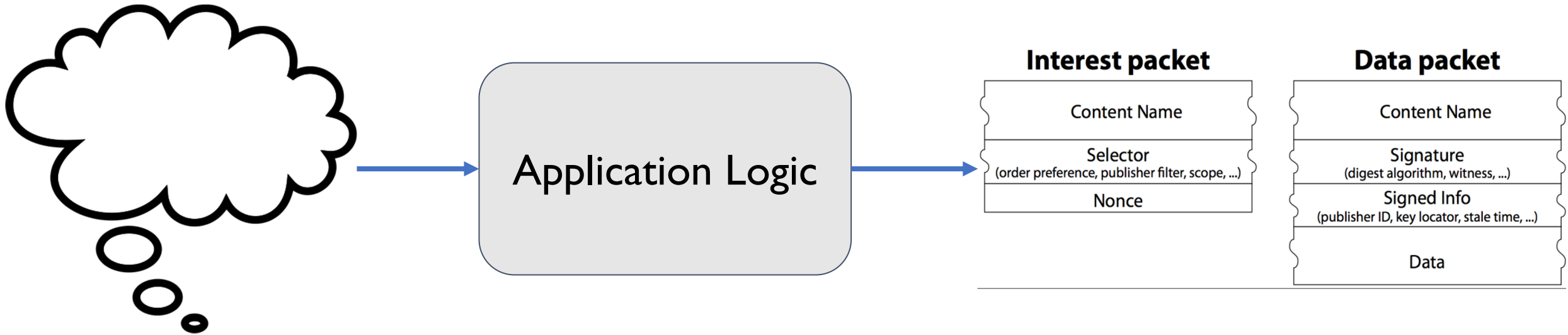
Practical NDN Application Development and Seamless Deployment

Name Tree Schema Code Example

September 29, Zoom, Earth, Solar System, Milky Way

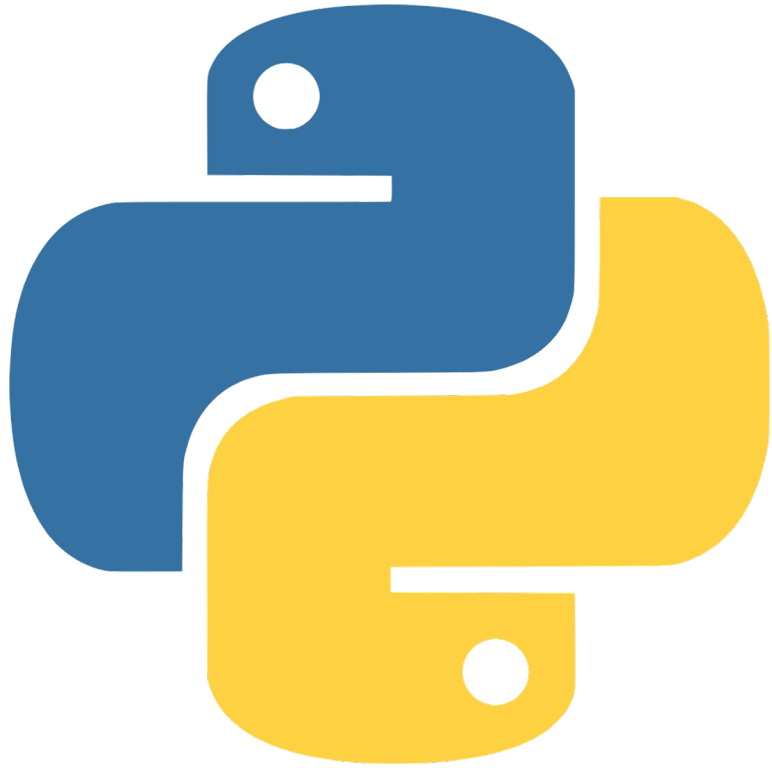
Xinyu Ma, UCLA

Why NDN App Development Is Hard?



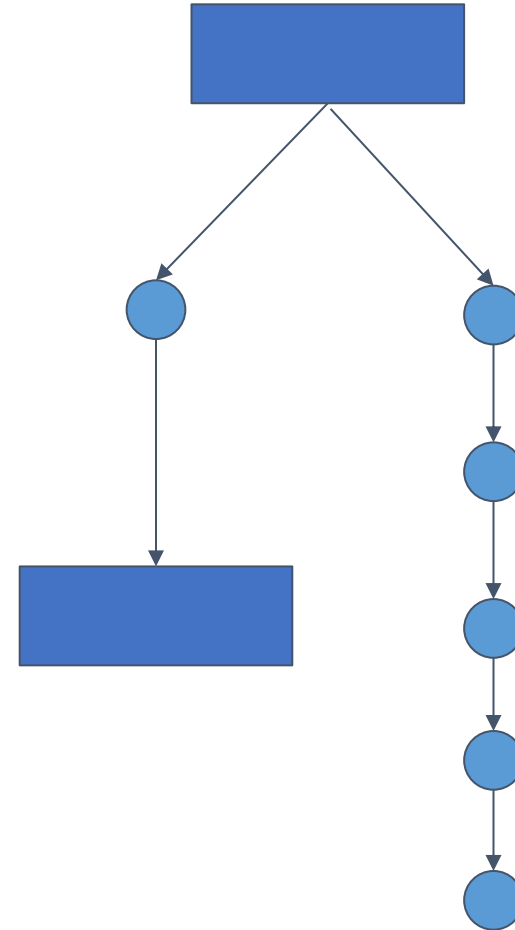
Need a systematic way!

A Less Painful Start



python-ndn

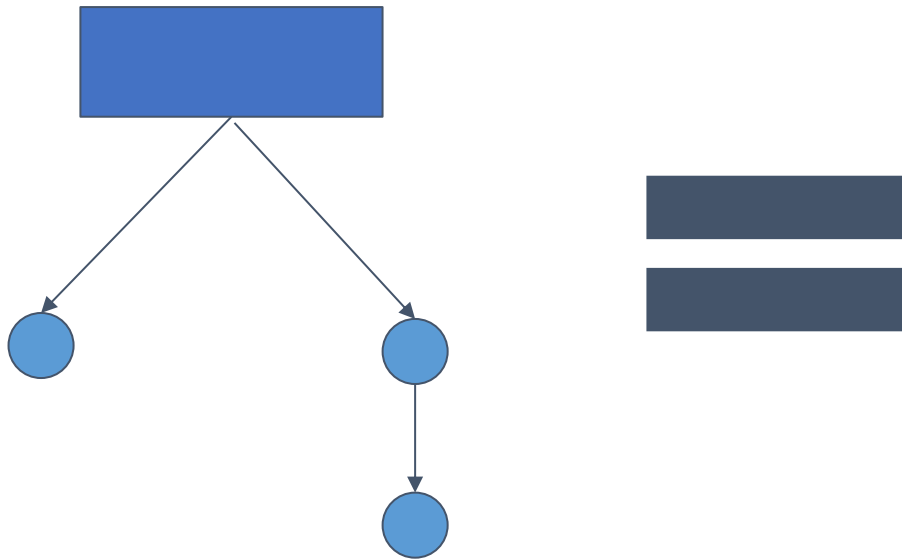
<https://github.com/named-data/python-ndn>



Name Tree Schema

CNL concept, different design

Our Goal

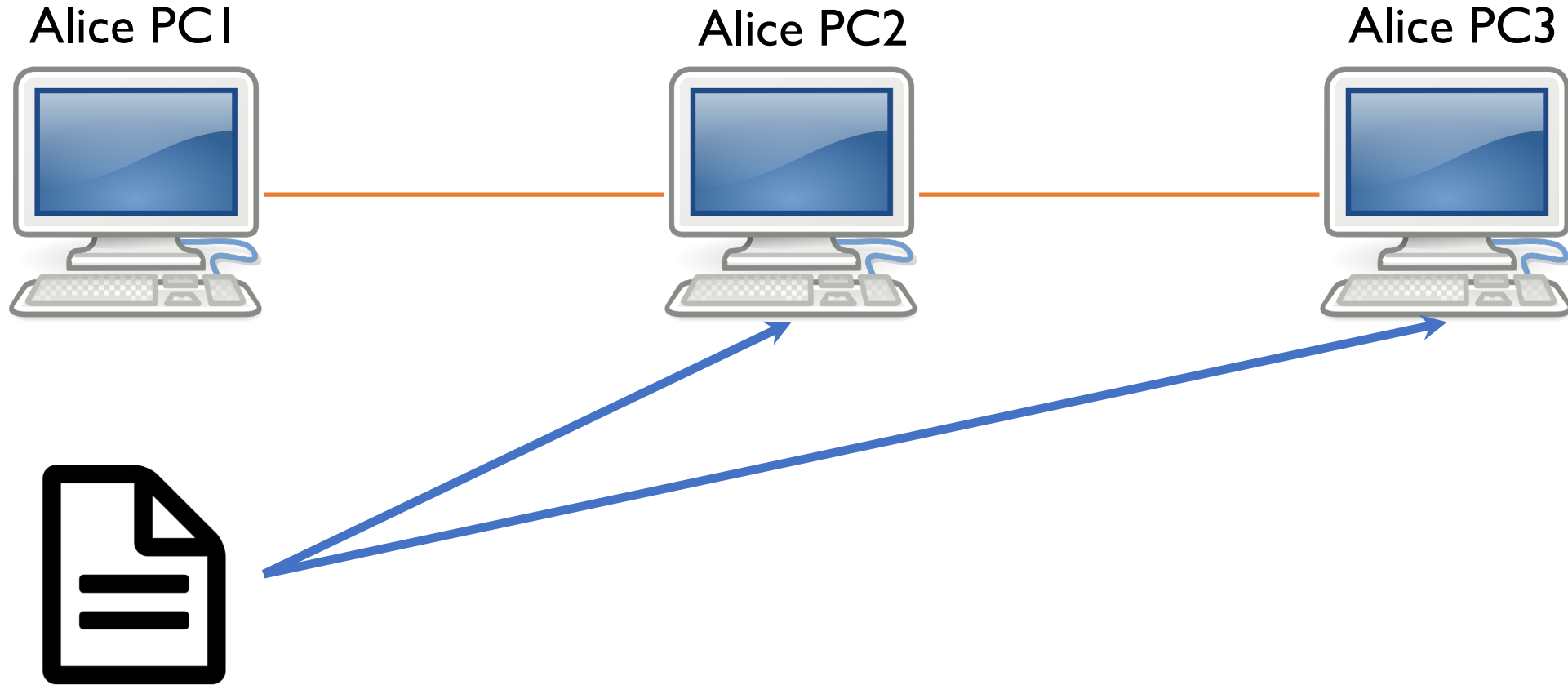


The structure of
Application Namespace

```
40
41 $(function){cards();});
42 $(window).on('resize', function){cards();});
43 function cards(){
44   var width = $(window).width();
45   if(width < 750){
46     cardssmallscreen();
47   }else{
48     cardsbigscreen();
49   }
50 }
51 function cardssmallscreen(){
52   var cards = $('<div class="card">').length;
53   var height = 0;
54   for(i=0; i<cards; i++){
55     card2 = $(cards[i]).height();
56     if(card2 > height){
57       height = card2;
58     }
59   }
60   cards.css('height', height);
61 }
```

The structure of
Program Code

Example Scenario: File Sharing

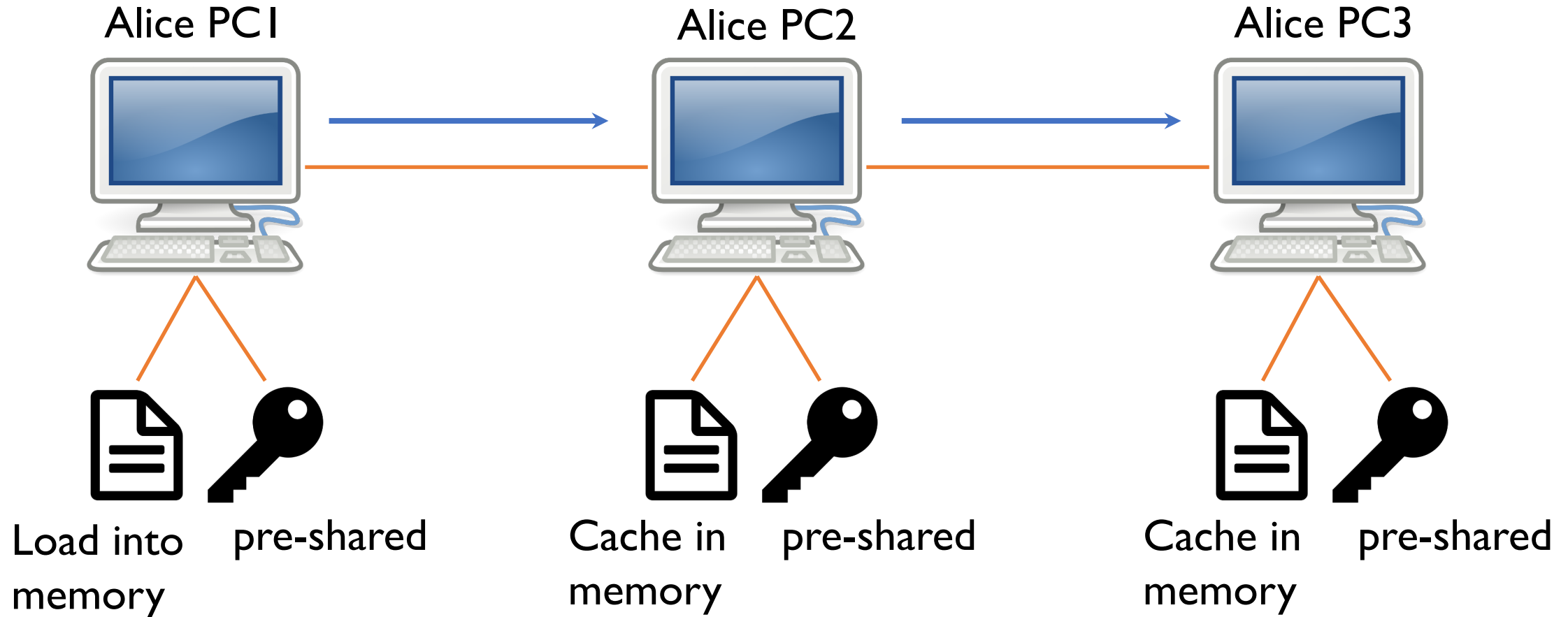


Scenario

- Alice wants to share a file among her computers.
- She can use her key as the trust anchor.
 - /Alice/KEY/%29i%D5%CE%C8A%CE%0F/self/%FD%00%00%0In%I0%8E%D1%F6
- The file may be large, so segmentation is needed.
 - RDR can be used here
 - <https://redmine.named-data.net/projects/ndn-tlv/wiki/RDR>
- RDR handles version discovery and segmentation.
 - A metadata packet contains the latest version number
 - The content is segmented
 - Let's use it as a black box, so no more details now.

Note: RDR is designed for versioned data. File sharing does not necessarily need it.

Overall Design

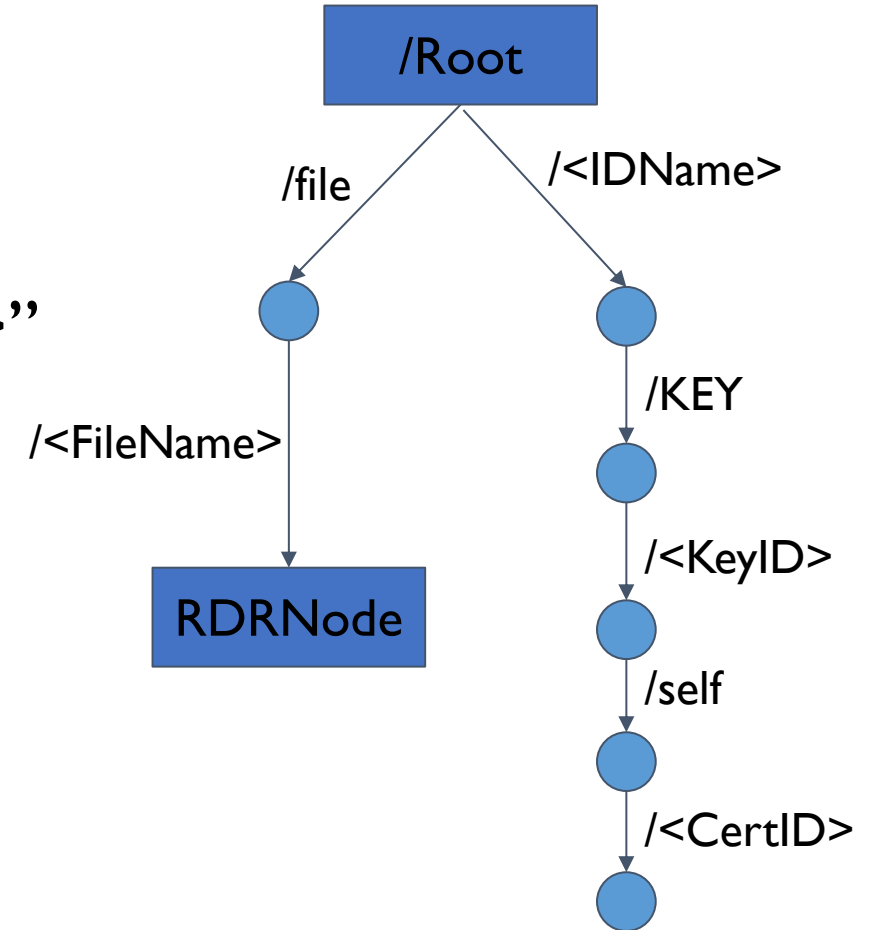


Namespace Schema Design

- A file object handled by RDR
 - Named “/file/<FileName>”
- Alice’s key and certificate
 - Named “/Alice/KEY/<KeyID>/self/<CertID>”

<...> are variables

Real names will be like “/file/foo.txt”

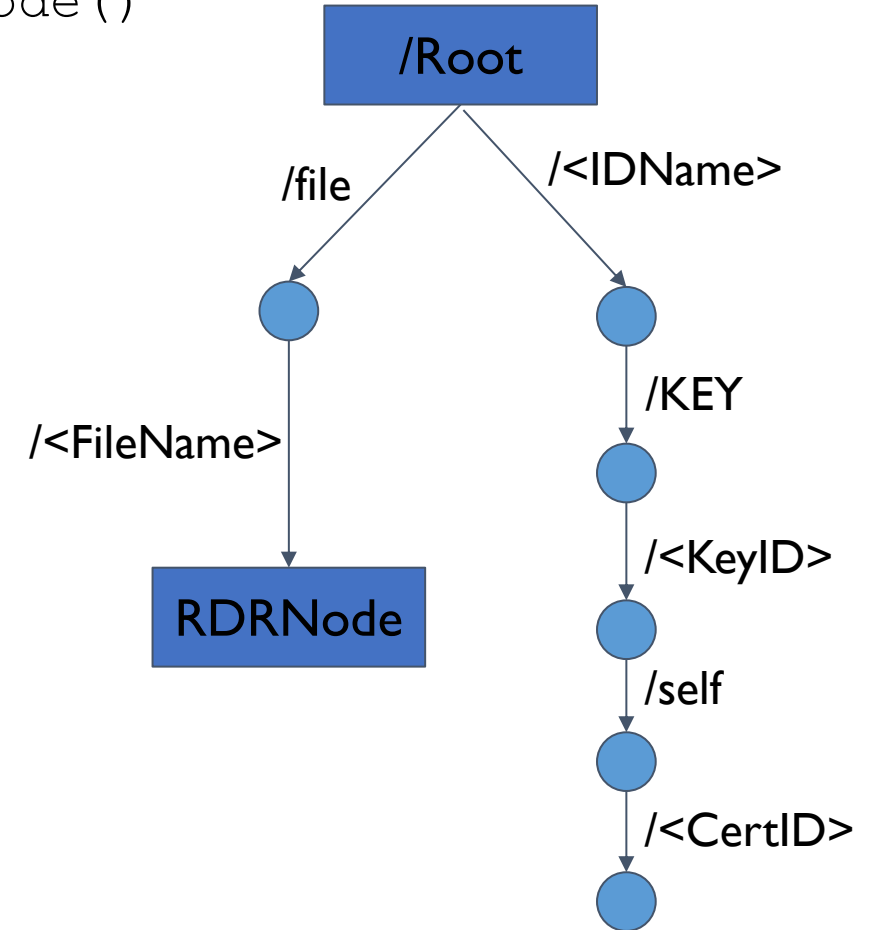


Namespace Schema Design

```
root = Node()
```

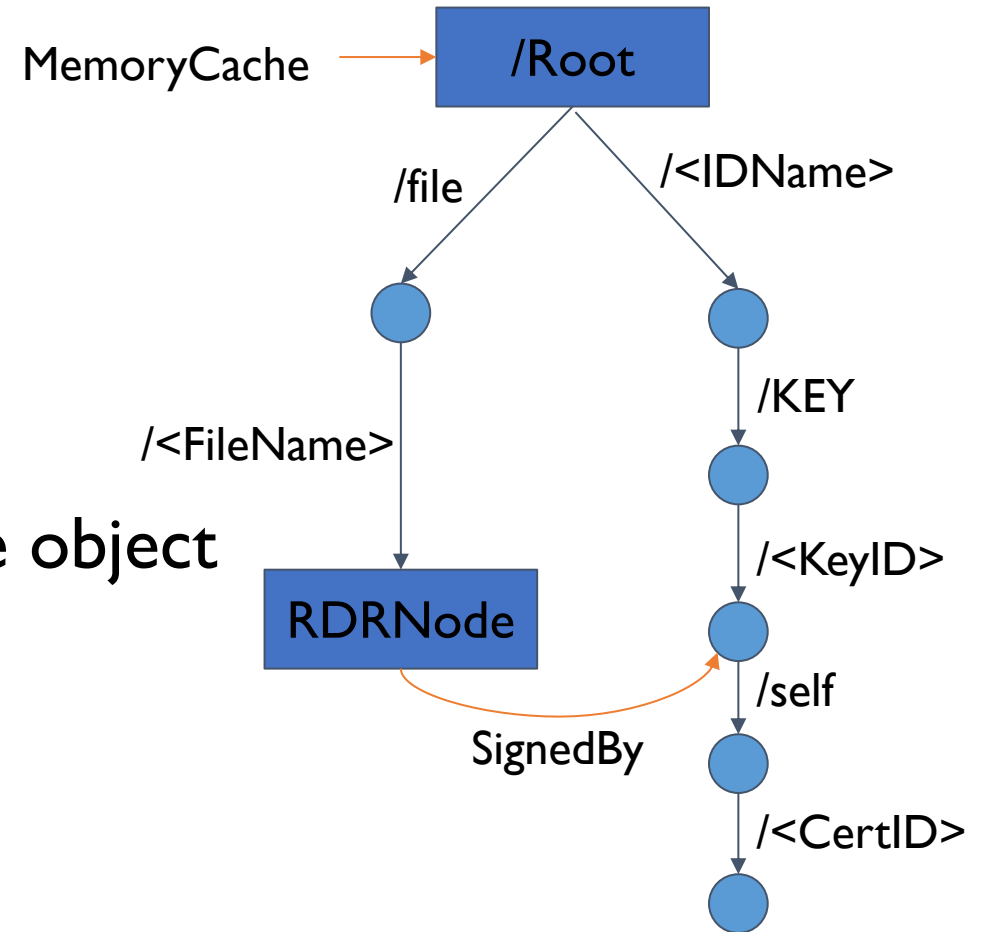
```
root[' /<IDName>/KEY/<KeyID>/self/<CertID>' ] = Node()
```

```
root[' /file/<FileName>' ] = RDRNode()
```



Policies

- A file object handled by RDR
 - All data stored in memory
 - Must be signed by Alice's key
- Alice's key and certificate
 - Preloaded and stored in memory
 - Can share the same storage with the file object



Policies

```
root = Node()
```

```
root['/<IDName>/KEY/<KeyID>/self/<CertID>'] = Node()
```

```
root['/file/<FileName>'] = RDRNode()
```

```
id_name = Name.Component.get_value(  
    app.keychain.default_identity().name[0])
```

```
cache = MemoryCache()
```

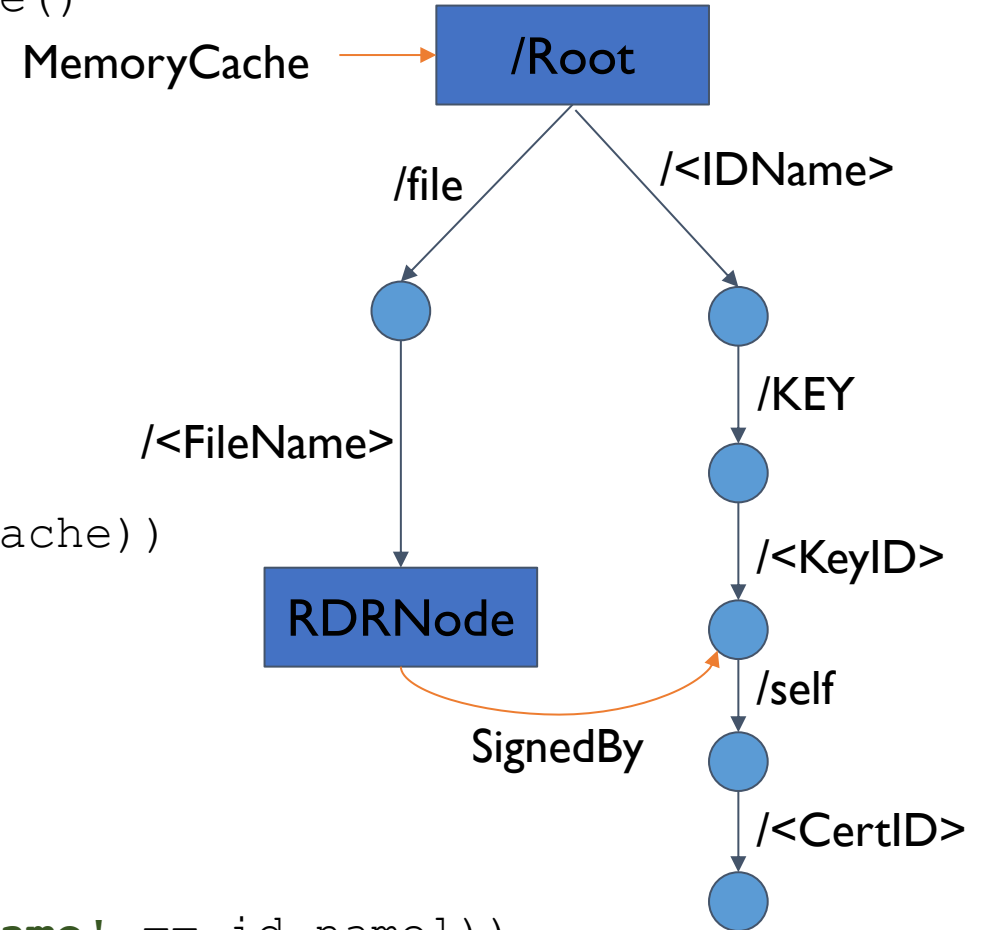
```
root.set_policy(policy.Cache, MemoryCachePolicy(cache))
```

```
root['/file/<FileName>'].set_policy(  
    policy.DataValidator,
```

```
    SignedBy(  
        root['/<IDName>/KEY/<KeyID>'],
```

```
        subject_to=lambda _, vars: vars['IDName' == id_name]))
```

```
subject_to=lambda _, vars: vars['IDName' == id_name]))
```



We Are Almost Done!

```
# Store the certificate
```

```
cert = app.keychain.default_identity().default_key().default_cert()
```

```
await cache.save(Name.normalize(cert.name), cert.data)
```

```
# Attach the tree to the face
```

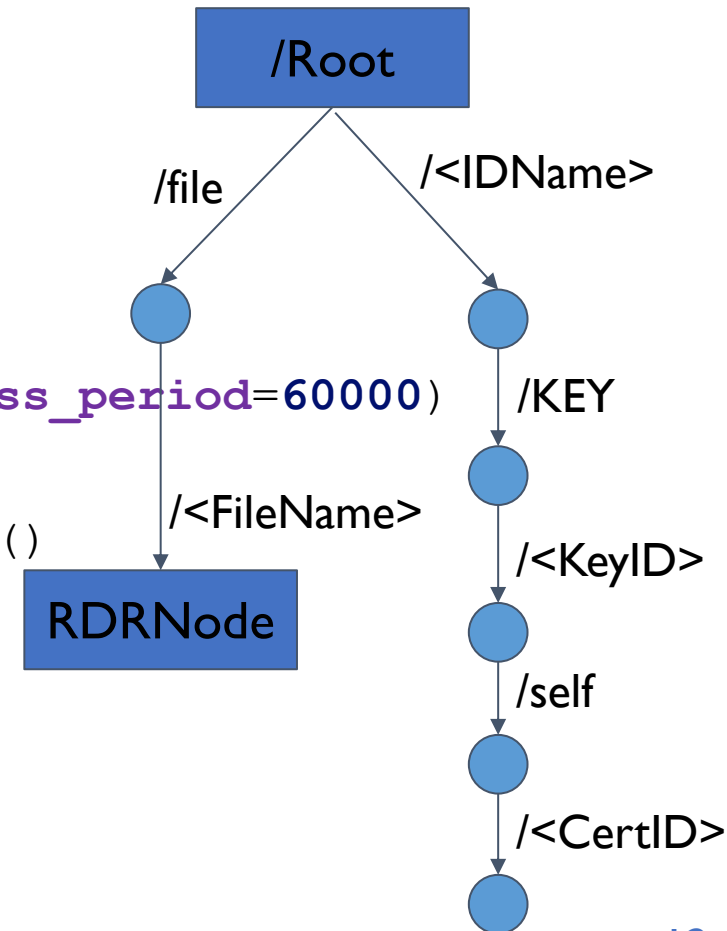
```
await root.attach(app, '/')
```

```
# If it runs as a producer, provide file
```

```
await root.match('/file/' + filename).provide(data, freshness_period=60000)
```

```
# If it runs as a consumer, require the file
```

```
data, metadata = await root.match('/file/' + filename).need()
```



Demo

```
pyndn3 — cs217a@s-164-67-229-234 — -zsh — 80x24
~ -zsh ... ..objects/pyndn3 ..objects/pyndn3 ..objects/pyndn3
(venv) await aio.sleep(0.1)
~/PycharmProjects/pyndn3 on schema! 1:54:46
$ [INSERT]
# If it's the producer
print(f'Try to fetch {filename}...')

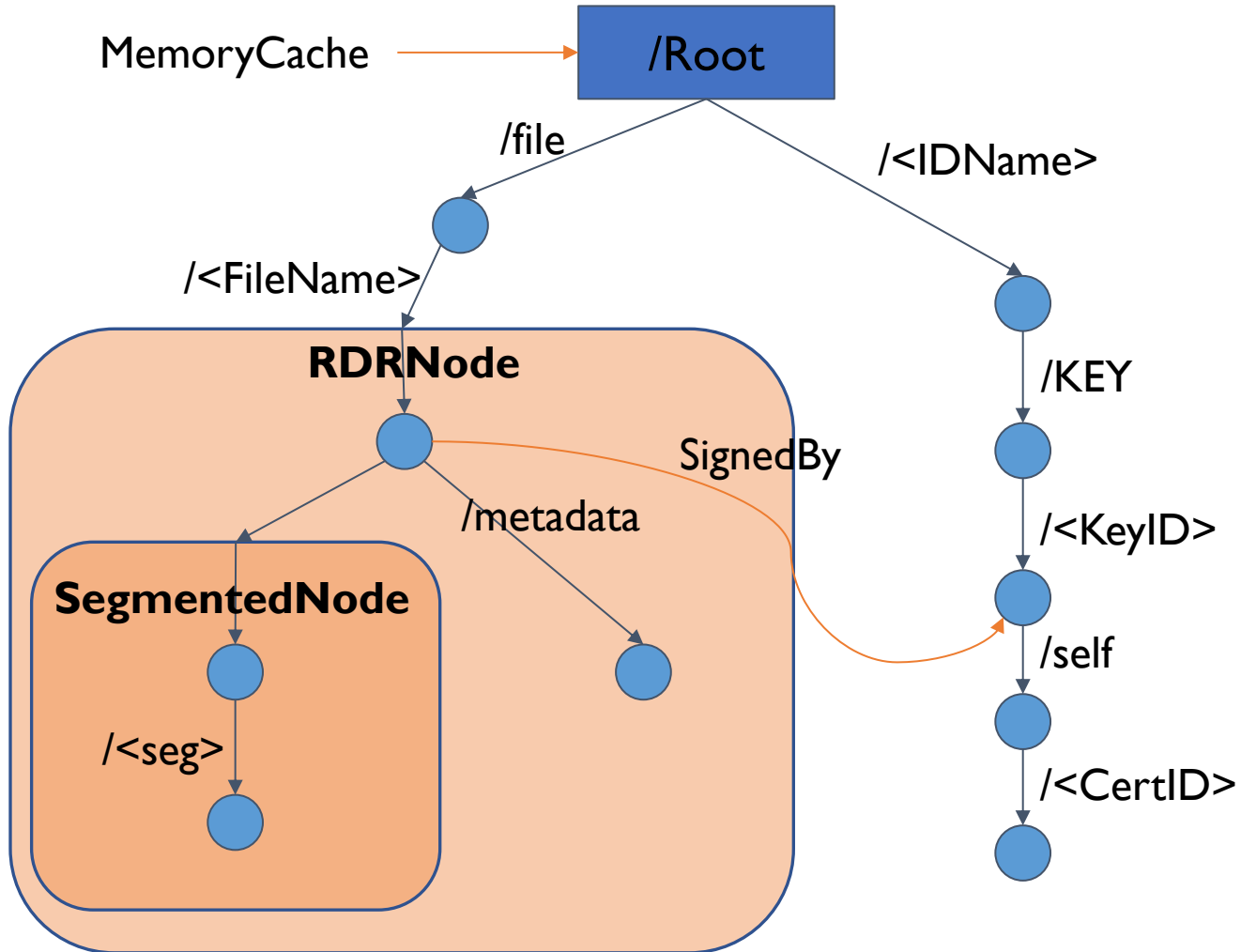
# The file is ready!
data, metadata = await root.match('/file/' + filename).need()
print(f'Content size: {len(data)}')
print(f'Content: {data[:70]} ...')
print(f'Number of segments: {metadata["block_count"]}')
print(f'Serving {filename}')

if __name__ == '__main__':
    app.run_forever(after_start=main())

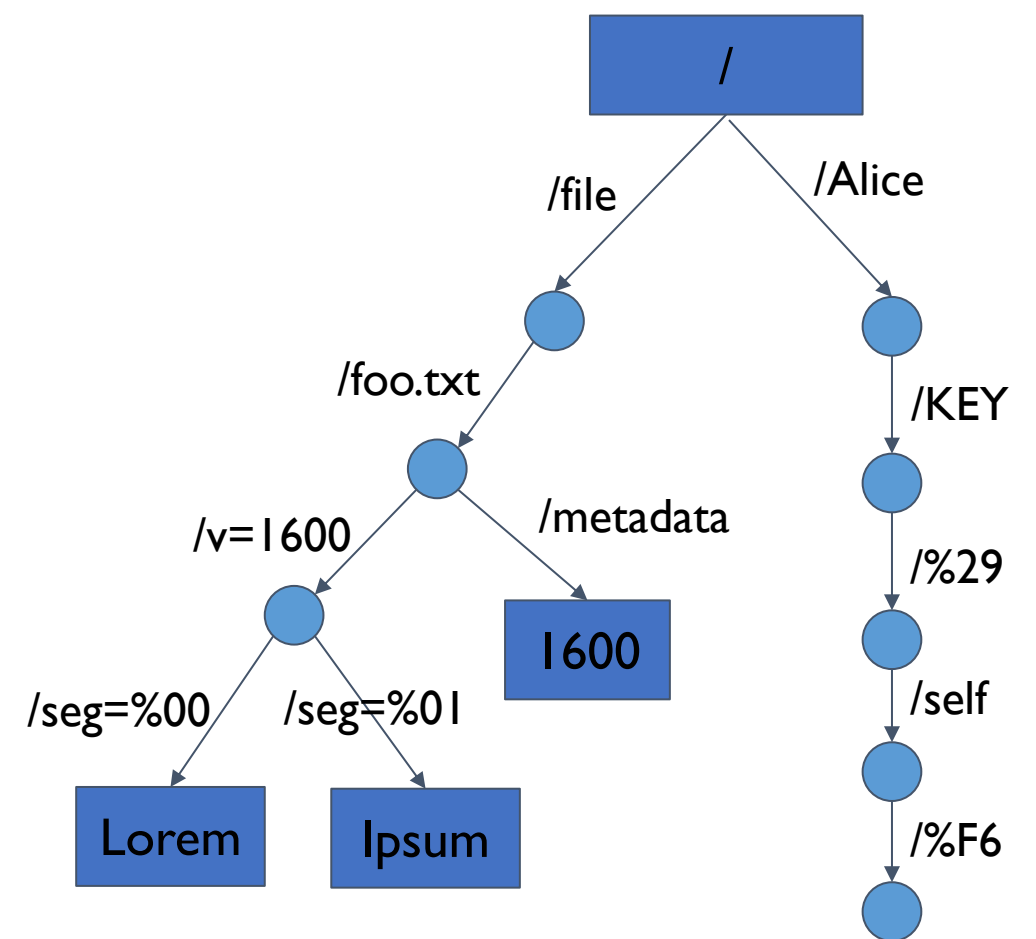
async main() Start the first node (producer)
```

But How It Works?

Namespace Schema



Name Tree in CNL



Summary

- Application namespace = Code structure
- A framework that all different libraries can be integrated into
 - Custom nodes are subtrees / substructures
 - Policies as filters / annotations
- Tutorial (with code) is online
 - <https://python-ndn.readthedocs.io/en/latest/src/schema/ex1.html>
- Future Work
 - Sync (CNL's functionality)
 - Repo (in-network storage)
 - Serialization of the schema tree