Consumer-Producer API for Named Data Networking

Ilya Moiseenko
How do we develop NDN apps?
Libraries: NDN-cxx, NDN-cpp

Figure 1. Interest / Data API 😊
Can we have more effective API?
Socket API offers:

- Point-to-point channel
- Reliable transmission
- Flow / congestion control
- Segmentation
- Ordering and reassembly
Closer look at NDN

Challenges

Integration of libraries requires near-expert knowledge of NDN
Is it possible?

To assist non-expert application developers fully exploiting NDN capabilities without a deep understanding of its network layer machinery.
Point-to-point model (IP)

Data transfer parameters are the property of the communication channel.
Socket abstraction

Keeps the state of communication channel

- **Protocol machinery** in use
- **state**: listen/connect, read/write
- send and receive **buffers**
- send and receive **timers**
- TCP maximum **segment size**
- ....
Distribution model (NDN)

What are the parameters (state) of information distribution?
What are the API calls?
Consumer-specific parameters

1. Selectors
2. Security
   - Data verification
   - Securing Interests
3. Fetching
   - Reliable transmission
   - Sequencing
4. Processing
   - Send / receive buffers
   - Reassembly
Producer-specific parameters

1. Security
   - Securing Data
   - Interest verification

2. Demultiplexing
   - Namespace (prefix) registration

3. Processing
   - Content segmentation
   - Send / receive buffers

4. Caching
New programming abstractions

1. Consumer context
2. Producer context

**Context** — keeps the state of data transfer under a specific name prefix
Consumer context

- Associates NDN name prefix with consumer-specific data transfer parameters
- Controls how Interests are expressed and how returning Data packets are processed

<table>
<thead>
<tr>
<th>Initialization</th>
<th>consumer (name prefix, type, sequencing) ➔ handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitives</td>
<td>consume (handle, name suffix)</td>
</tr>
<tr>
<td></td>
<td>stop (handle)</td>
</tr>
<tr>
<td></td>
<td>close (handle)</td>
</tr>
<tr>
<td></td>
<td>setcontextopt (handle, option name, value)</td>
</tr>
<tr>
<td></td>
<td>getcontextopt (handle, option name)</td>
</tr>
</tbody>
</table>
Producer context

- Associates NDN name prefix with producer-specific data transfer parameters
- Controls how data is produced and secured, and how Interests are demultiplexed.

<table>
<thead>
<tr>
<th>Initialization</th>
<th>producer (name prefix) ➞ handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitives</td>
<td>produce (handle, name suffix, content)</td>
</tr>
<tr>
<td></td>
<td>setup (handle)</td>
</tr>
<tr>
<td></td>
<td>close (handle)</td>
</tr>
<tr>
<td></td>
<td>setcontextopt (handle, option name, value)</td>
</tr>
<tr>
<td></td>
<td>getcontextopt (handle, option name)</td>
</tr>
</tbody>
</table>
Summary

Consumer and producer contexts

• Offer a richer set of functions than sockets
• are tailored for NDN distribution model
• include communication protocols for faster app development
• will include versioning protocols
Thank you!

See the poster to get more info!