

Socket API

bind() & listen() - establish packet demultiplexing
connect() & accept() - set up a reliable stream (TCP)
send() - segmentation, retransmission and flow/congestion control
read() - packet ordering and reassembly
 + UDP and RAW socket API calls

NDN API

Retrieval of single- and multi-segment content
Reliable and **unreliable** transmission
Setting up a demux point
Packaging of content in Data packets: segmentation, etc.
Plugging in user-defined security and verification actions
Monitoring events related to transmission of data

Consumer context

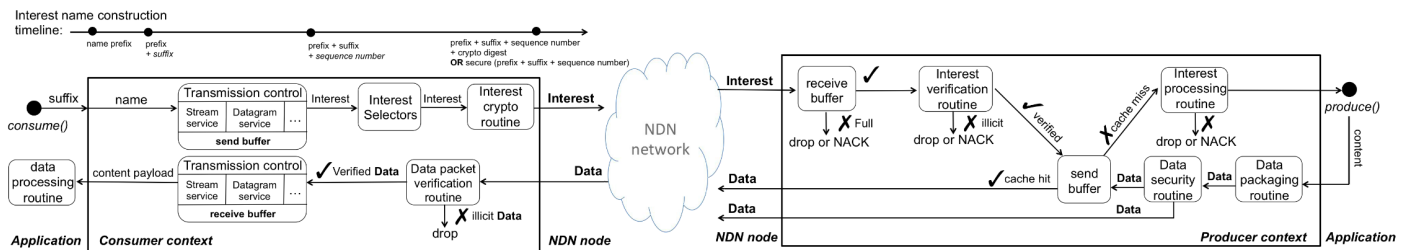
Consumer context associates a name prefix with consumer-specific data fetch parameters controlling Interest transmission and Data packet processing.

| | |
|----------------|--|
| Initialization | consumer (name prefix, type, sequencing) → handle |
| Primitives | consume (handle, name suffix) stop (handle) close (handle) setcontextopt (handle, option name, value) getcontextopt (handle, option name) |

Producer context

Producer context associates a name prefix with producer-specific data transfer parameters controlling Interests demultiplexing and Data packet production.

| | |
|----------------|--|
| Initialization | producer (name prefix) → handle |
| Primitives | produce (handle, name suffix, content) setup (handle) close (handle) setcontextopt (handle, option name, value) getcontextopt (handle, option name) |



File sharing

```

Pseudocode 1 Sharing a file
1: h ← producer("/broadcast/apps/filesync")
2: setcontextopt(h, packet_size, 16KB)
3: setcontextopt(h, interest_callback, ProcessInterest)
4: setup(h)
5: function PROCESSINTEREST(Interest i)
6:   Name suffix ← read i.name to understand what file is needed
7:   content ← read file from disk
8:   Name suffix ← append current time stamp
9:   produce(h, Name suffix, content)
10: end function

Pseudocode 2 Downloading a file
1: h ← consumer("/broadcast/apps/filesync", RELIABLE, SEQUENCE)
2: setcontextopt(h, receive_buffer_size, 20MB)
3: setcontextopt(h, content_callback, ProcessContent)
4: consume(h, "/class217/Reports/Report.pdf")
5: function PROCESSCONTENT(byte[] content)
6:   file ← read content
7:   Save file on disk
8: end function
  
```

Video streaming

```

Pseudocode 3 Producing video
1: h ← producer("/edu/ucla/stream")
2: setcontextopt(h, packet_size, 8KB)
3: setcontextopt(h, send_buffer_size, 100MB)
4: setup(h)
5: while True do
6:   Name suffix ← name and quality of the video
7:   content ← encode captured video
8:   produce(h, Name suffix, content)
9: end while

Pseudocode 4 Consuming video
1: h ← consumer("/edu/ucla/stream", UNRELIABLE, SEQUENCE)
2: setcontextopt(h, receive_buffer_size, 1MB)
3: setcontextopt(h, send_rate, 50ms)
4: setcontextopt(h, content_callback, ProcessContent)
5: consume(h, "%FD%04%F65ub/video0/h264-1024k")
6: function PROCESSCONTENT(byte[] content)
7:   video ← decode content
8:   Display video
9: end function
  
```

Lighting control

```

Pseudocode 5 NDN gateway for lighting panel
1: h ← producer("/ndn/ucla/boelter/3551/lights/fixture41/rgb-8bit-hex")
2: setcontextopt(h, verification_callback, VerifyInterest)
3: setcontextopt(h, interest_callback, ProcessInterest)
4: setcontextopt(h, security_callback, SecureData)
5: setup(h)

Pseudocode 6 NDN gateway for lighting panel (continued)
6: function VERIFYINTEREST(Interest i)
7:   Signature ← read signature bits from i.name
8:   if Signature is valid then
9:     return True
10:  else
11:    Name suffix ← read <color>, <state>, <signature> from i.name
12:    produce(h, Name postfix, VerificationNACK)
13:    return False
14:  end if
15: end function

Pseudocode 7 Light controller
1: h ← consumer("/ndn/ucla/boelter/3551/lights", RELIABLE, DATAGRAM)
2: setcontextopt(h, crypto_callback, SignInterest)
3: setcontextopt(h, verification_callback, VerifyData)
4: setcontextopt(h, content_callback, ProcessContent)
5: consume(h, "/fixture41/rgb-8bit-hex/FAF87F")
6: function SIGNINTEREST(Interest i)
7:   Signed Interest ← sign(i)
8:   return Signed Interest
9: end function
10: function VERIFYDATA(Data d)
11:   if verify(d.signature, d.keyLocator) is True then
12:     return True
13:   else
14:     return False
15:   end if
16: end function
17: function PROCESSCONTENT(byte[] content)
18:   Secured data ← encrypt(d)
19:   Secured data ← sign(Encrypted data)
20:   return Secured data
21: end function
  
```