Chipmunk: Distributed Object Storage for NDN

ACM ICN 2020 DEMO

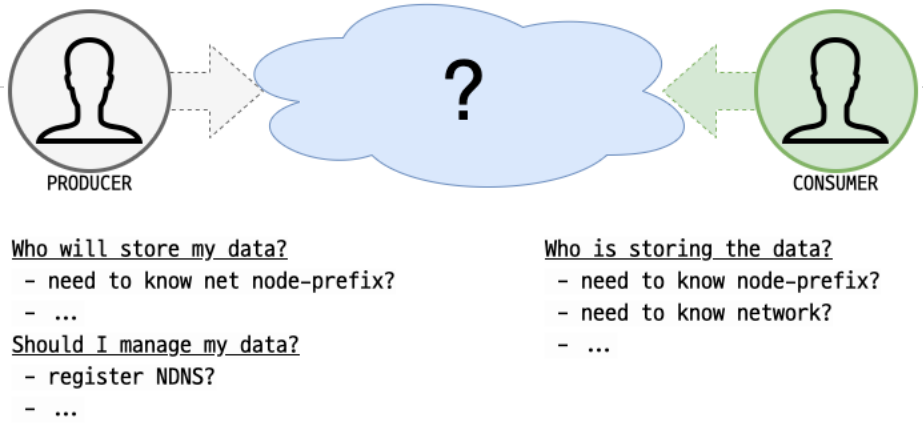
Yong Yoon Shin, Sae Hyung Park, Namseok Ko (ETRI), Arm Jeong (GurumNetworks)

2020.09.30. (UTC 6PM ~ 8PM)
- Poster and Demo Session, 5th presentation

GOAL

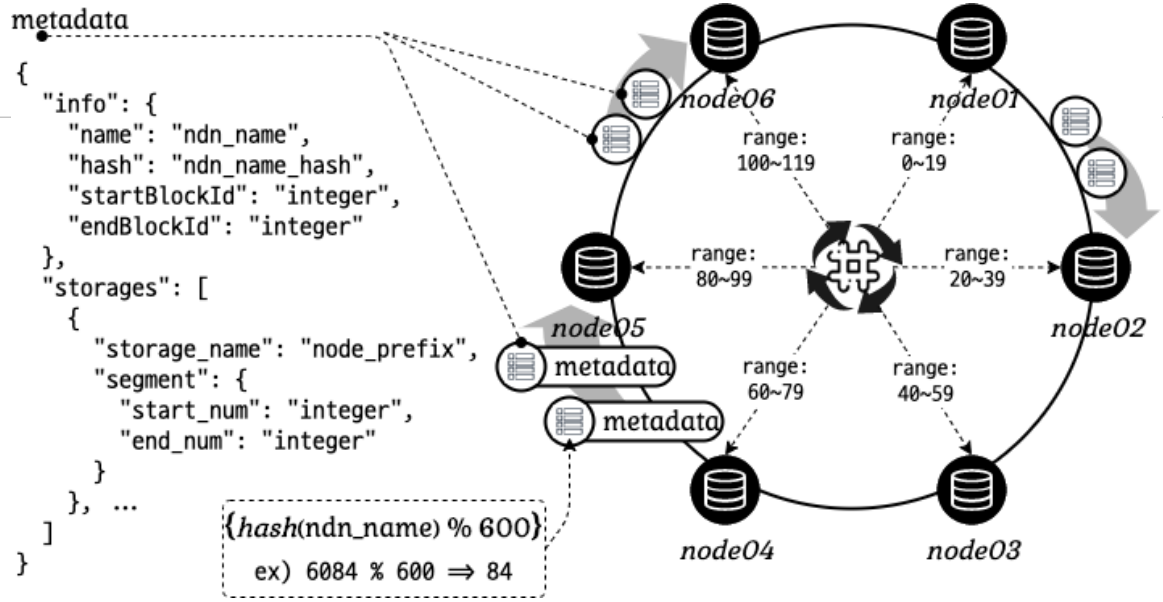
user-friendliness

- producers want persistent data store
  - they don’t know exact store prefix
  - store the data using simple prefix
- consumers want storage that is simple to use
  - they don’t know exactly who is storing data
- inherits the command-set of ***repo-ng***



scalability

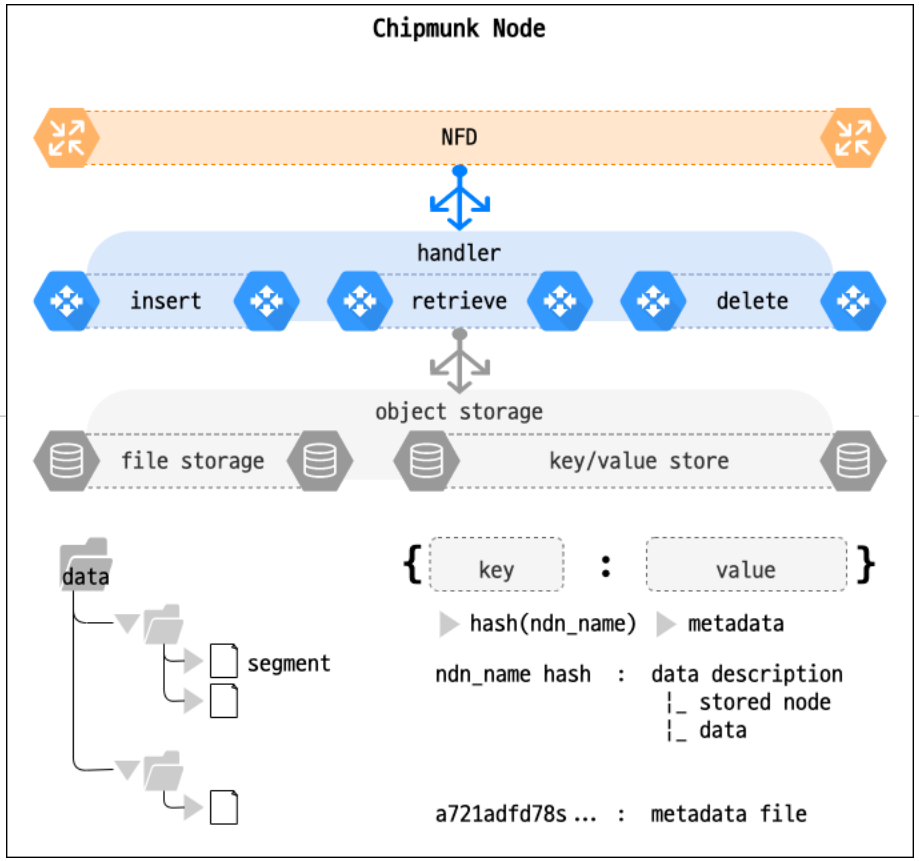
- DHT-based distributed file storage
- every participating node has **a common prefix as a service name** besides its own prefix
  - chipmunk service prefix <ndn:/chipmunk>
- files are stored after being **encapsulated** with node prefix
  - no need to announce the file
  - simple validation



DESIGN

node

- object storage based on ***repo-ng***
- node consists of **file storage** and **key/value store**
  - **file storage** stores segments of data
  - **key/value store** stores metadata

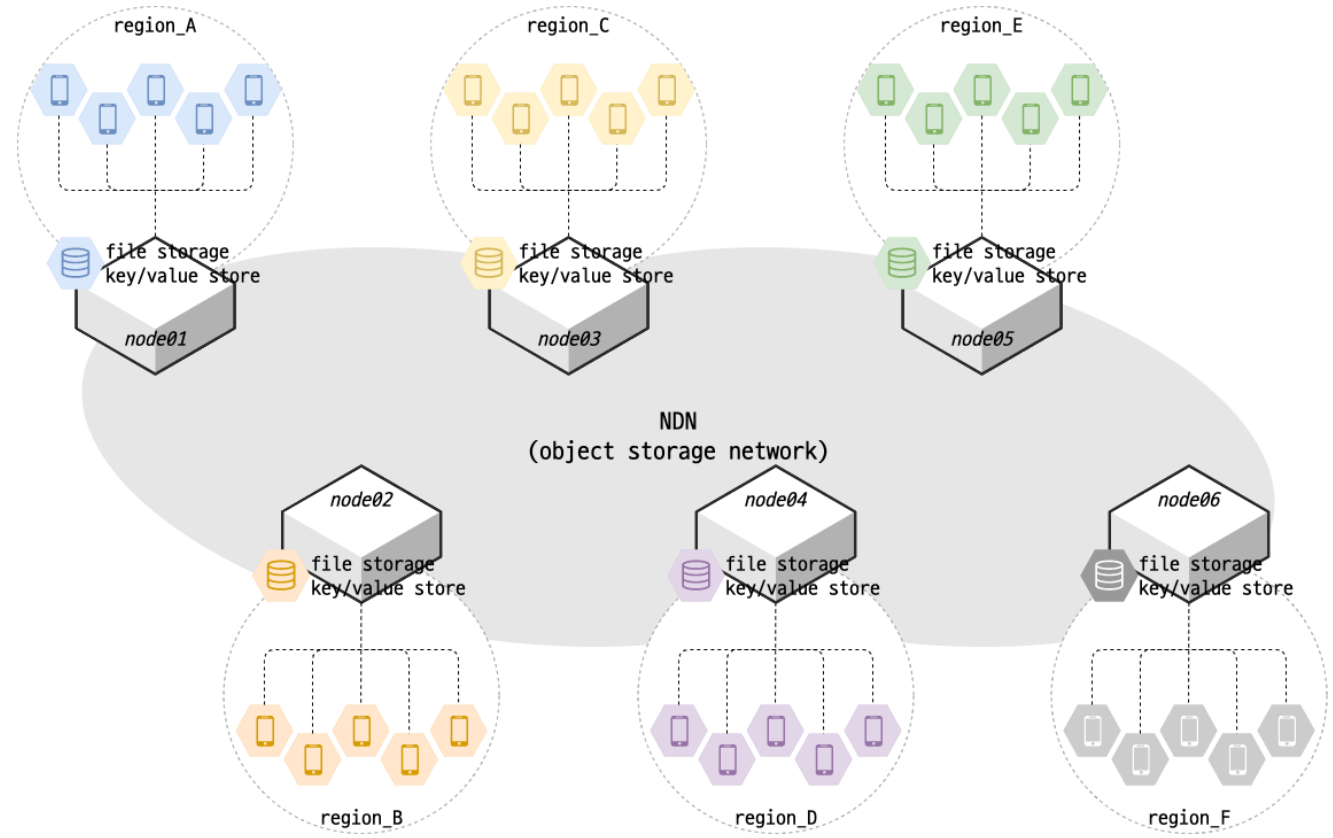


object storage

- store data in network
- name and data should be mapped
- easy to find data regardless of its network location

ENVIRONMENT

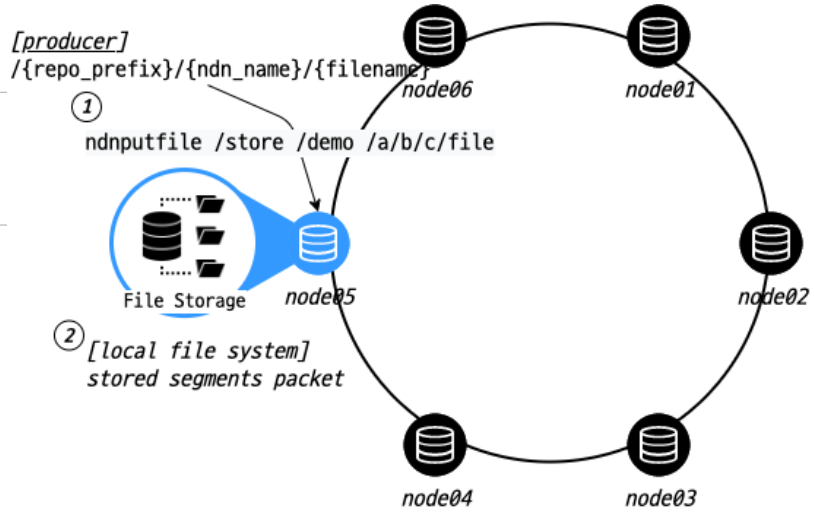
- each node connected by NDN
- using tools
  - `ndnputfile <repo_prefix> <ndn_name> <filename>`
  - `ndngetfile <ndn_name>`
  - `ndncypyfile <src> <dst> <ndn_name>`



DEMO1

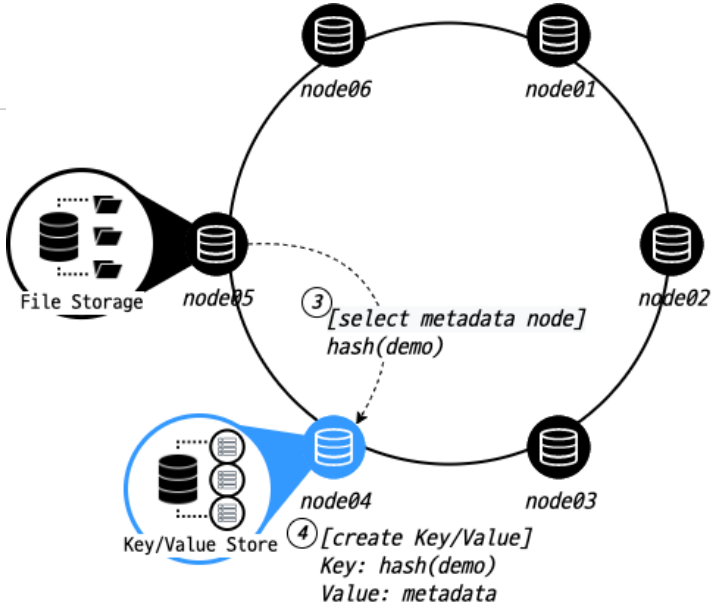
insert

- the producer requests to insert a data segment
  - `ndnputfile <repo_prefix> <ndn_name> <filename>`
- a Chipmunk node, which receives an insert request, pulls the data through the exchange of interest and data messages, and stores it in the file system



create metadata

- a metadata is created and the decision about selecting a node to store the metadata is made by a hash calculation on `ndn_name`
- the request to store the metadata is delivered to the selected metadata store node



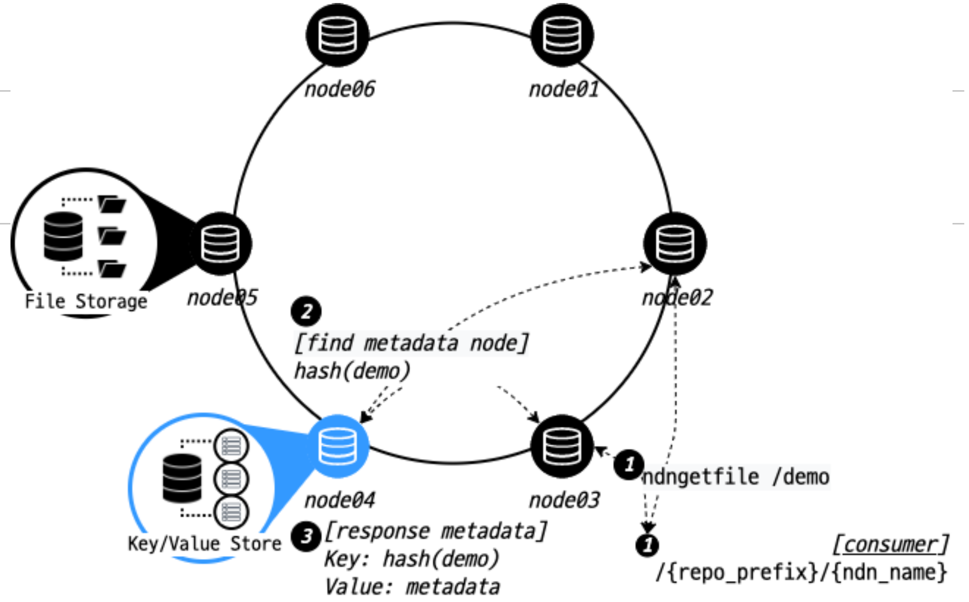
DEMO1

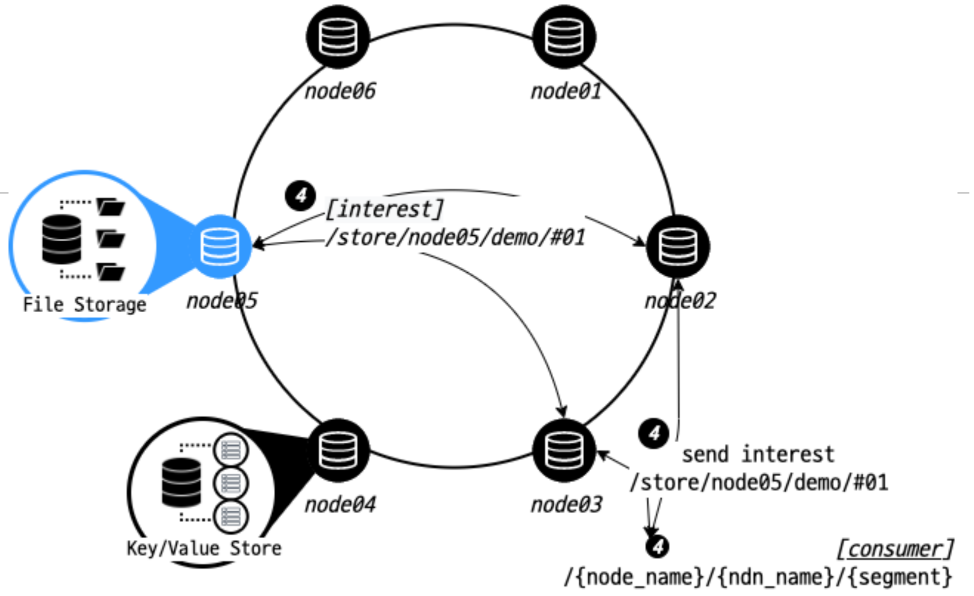
retrieve and metadata

- the consumer requests to get a data with `ndn_name`
  - `ndngetfile <ndn_name>`
- the node that gets the request from the consumer, which is usually the closest to the consumer, finds the metadata for the data
- the metadata is returned to the consumer, which has the information where the data is stored

send interest

- the consumer sends a request to a node that actually stores the data
  - `/<node_name>/<ndn_name>/<segment>`

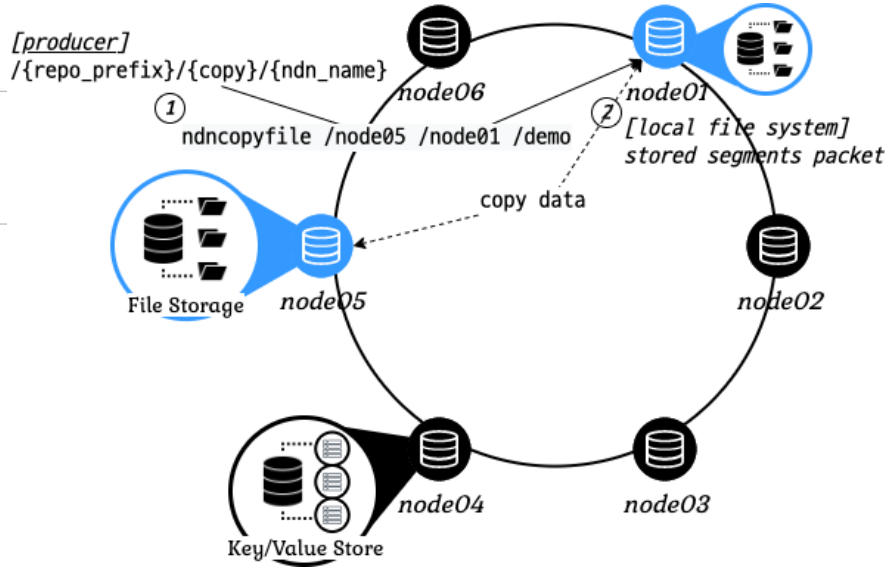




DEMO2

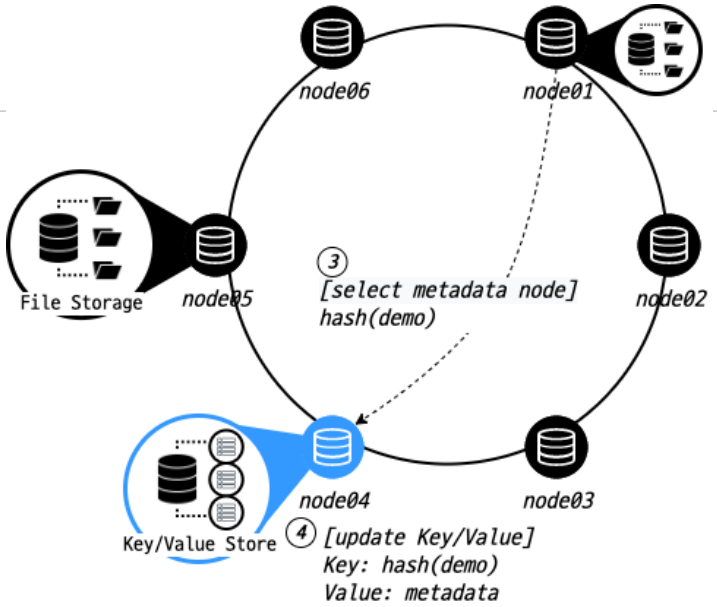
insert

- the producer requests to copy a data segment
  - `ndncopyfile <src> <dst> <ndn_name>`
- the same data can be requested to be stored in multiple Chipmunk nodes for some purposes such as increasing resiliency



create metadata

- a metadata is created and the decision about selecting a node to store the metadata is made by a hash calculation on `ndn_name`
- the request to store the metadata is delivered to the selected metadata store node



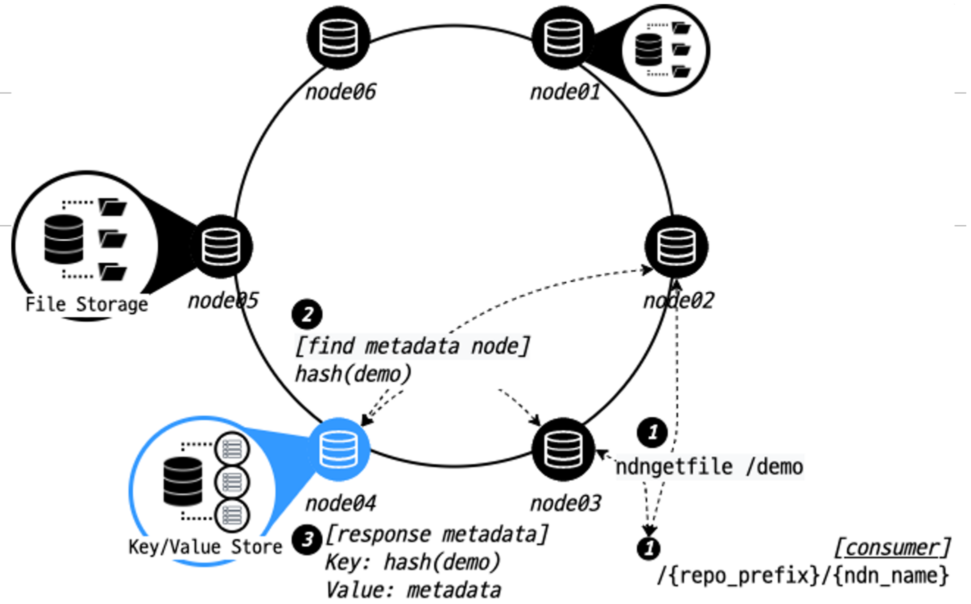
DEMO2

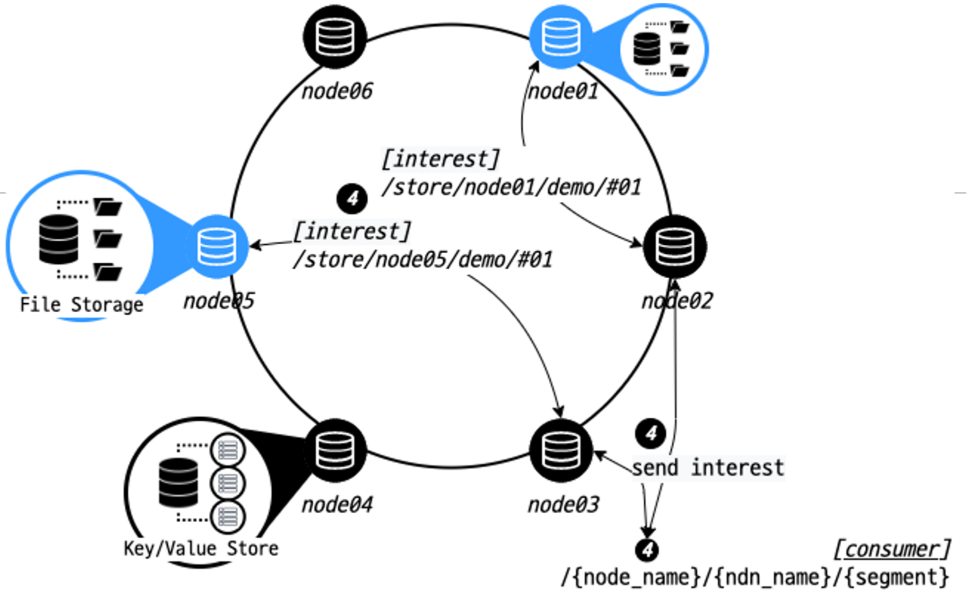
retrieve and metadata

- the consumer requests to get a data with `ndn_name`
  - `ndngetfile <ndn_name>`
- the node that gets the request from the consumer, which is usually the closest to the consumer, finds the metadata for the data
- the metadata is returned to the consumer, which has the information where the data is stored

send interest

- there are multiple nodes storing the data, so consumers can choose one of them depending on the policies
  - `/<node_name>/<ndn_name>/<segment>`





NOW CHIPMUNK

phase1

- basic model (this demo)

▶ phase2 (on going)

- more like NDN
  - forwarding hint
- new signature model
  - hash chain
- use container & NFN
  - storing network function data
  - sharing host volume
- self config (consistent hashing)
  - auto configuration
  - dynamic storage node add/delete
- performance
  - increases performance in indicators such as speed and capacity
- opensource contribute

Thanks

email

- Yong Yoon Shin (uni2u@etri.re.kr)
- Sae Hyung Park (labry@etri.re.kr)