



Creating a FABRIC Service

(KNIT 11 Tutorial)

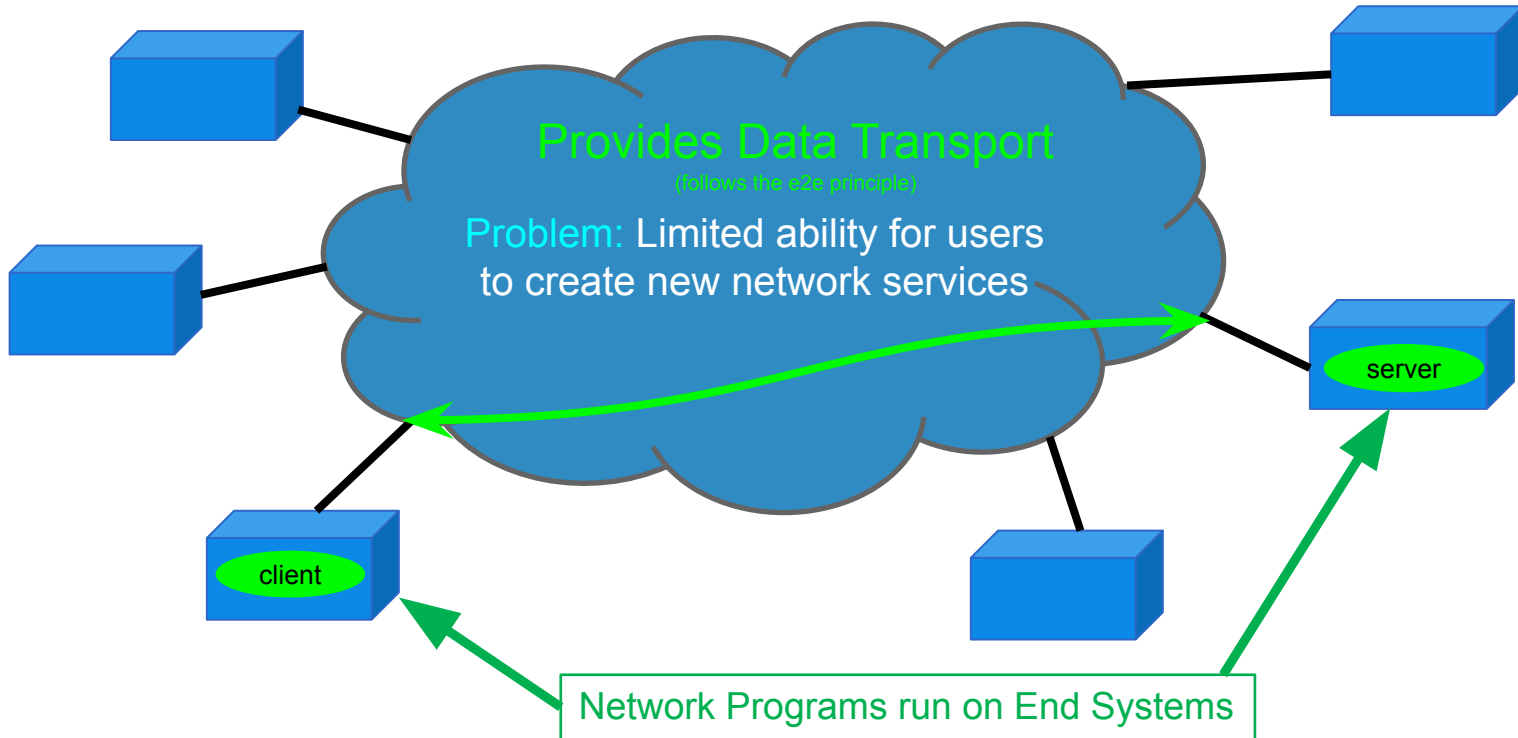
University of Kentucky FABRIC team

Presented by:
October 15, 2025

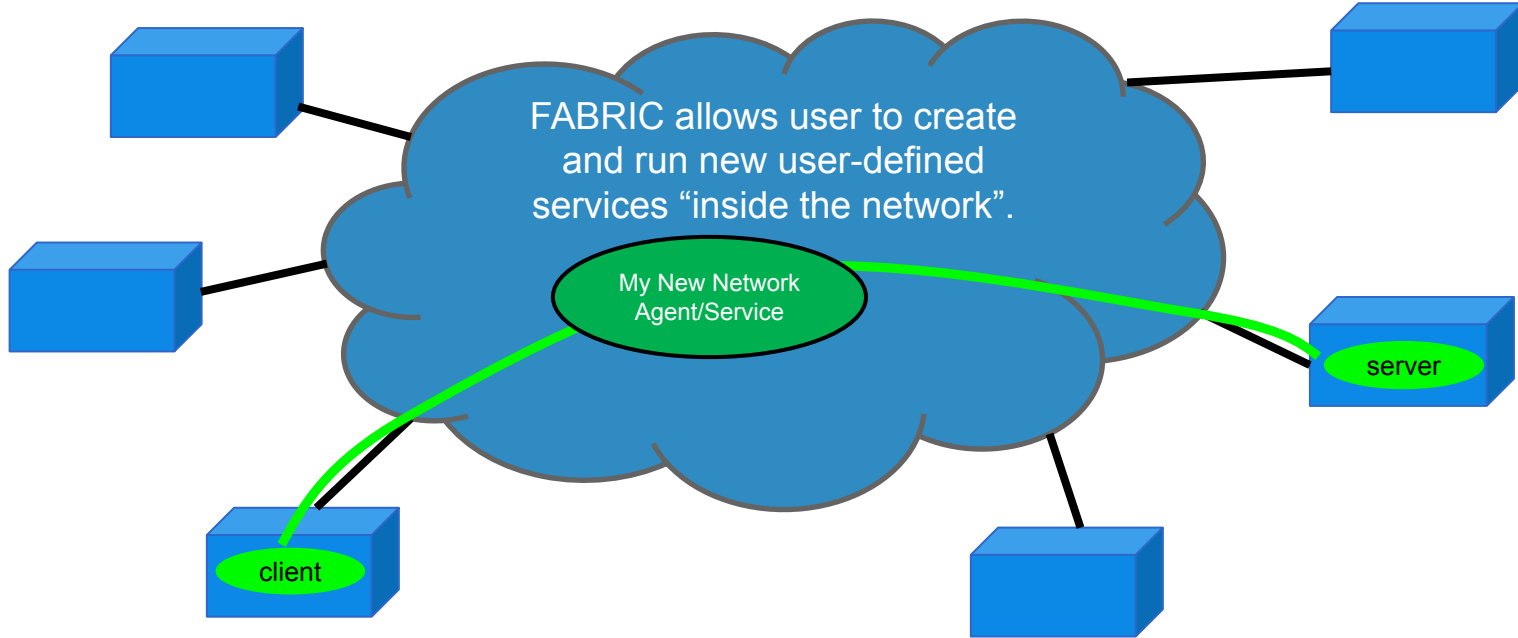
James Griffioen, Pinyi Shi, Charles Carpenter, and Hussamuddin Nasir



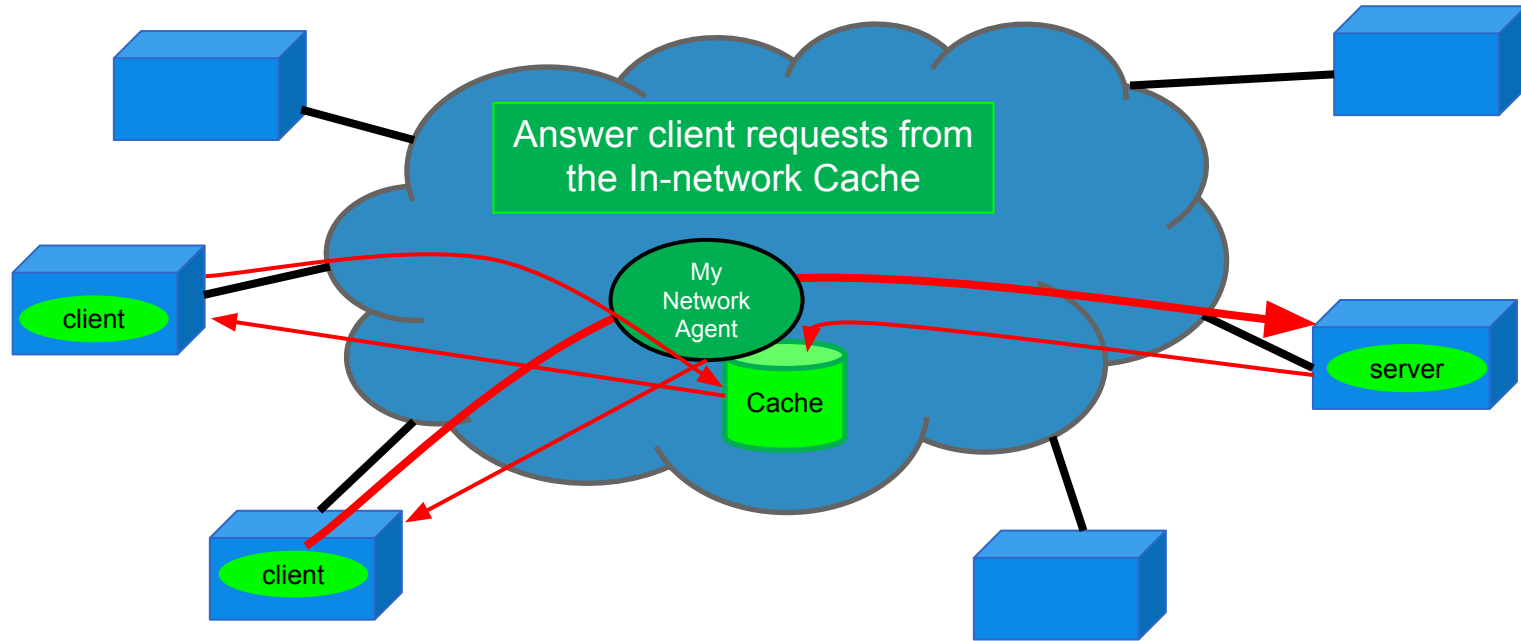
Today's Internet



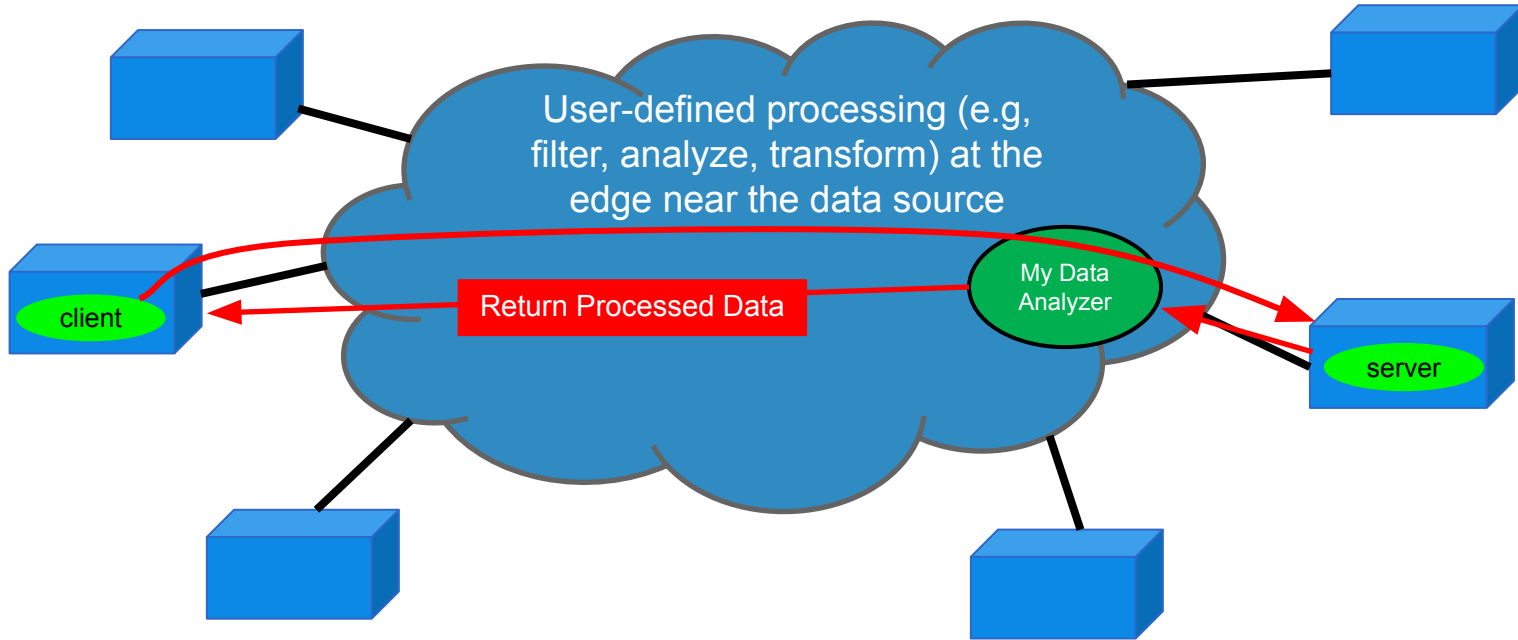
FABRIC Enables New Network Services



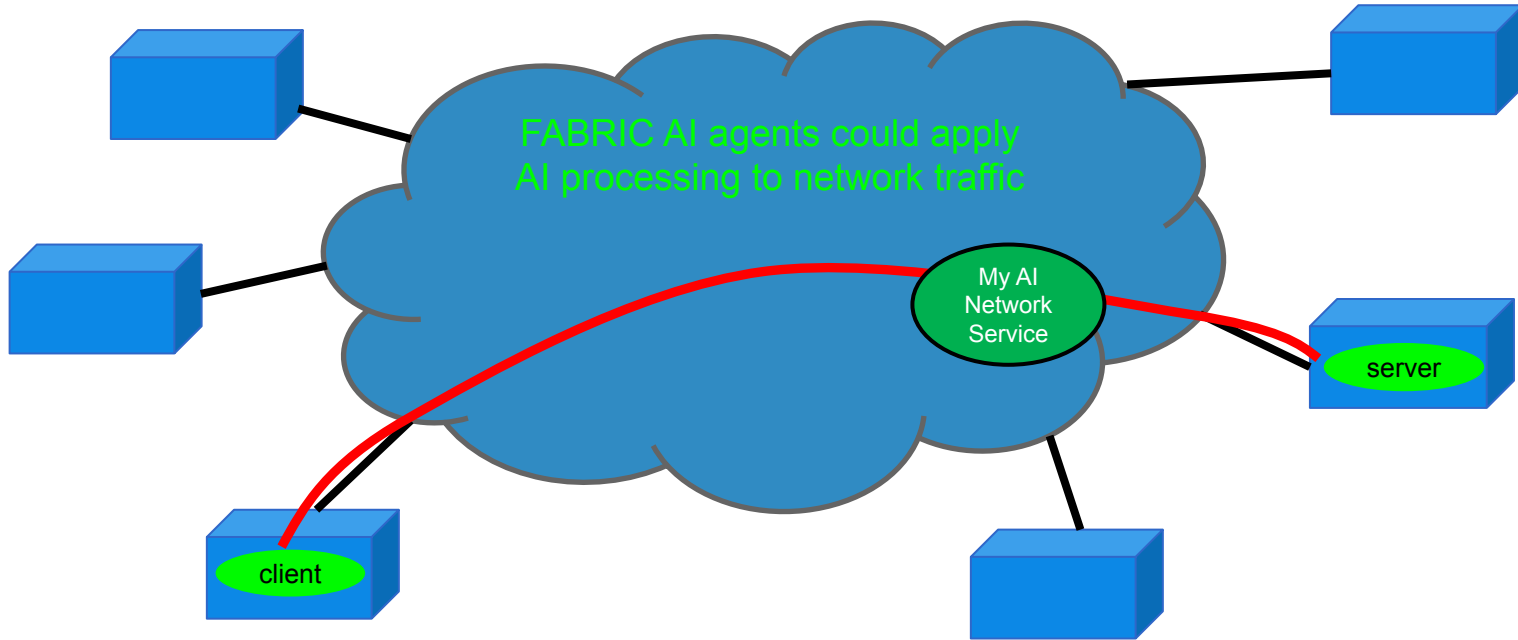
Example: In-network Data Caching Service



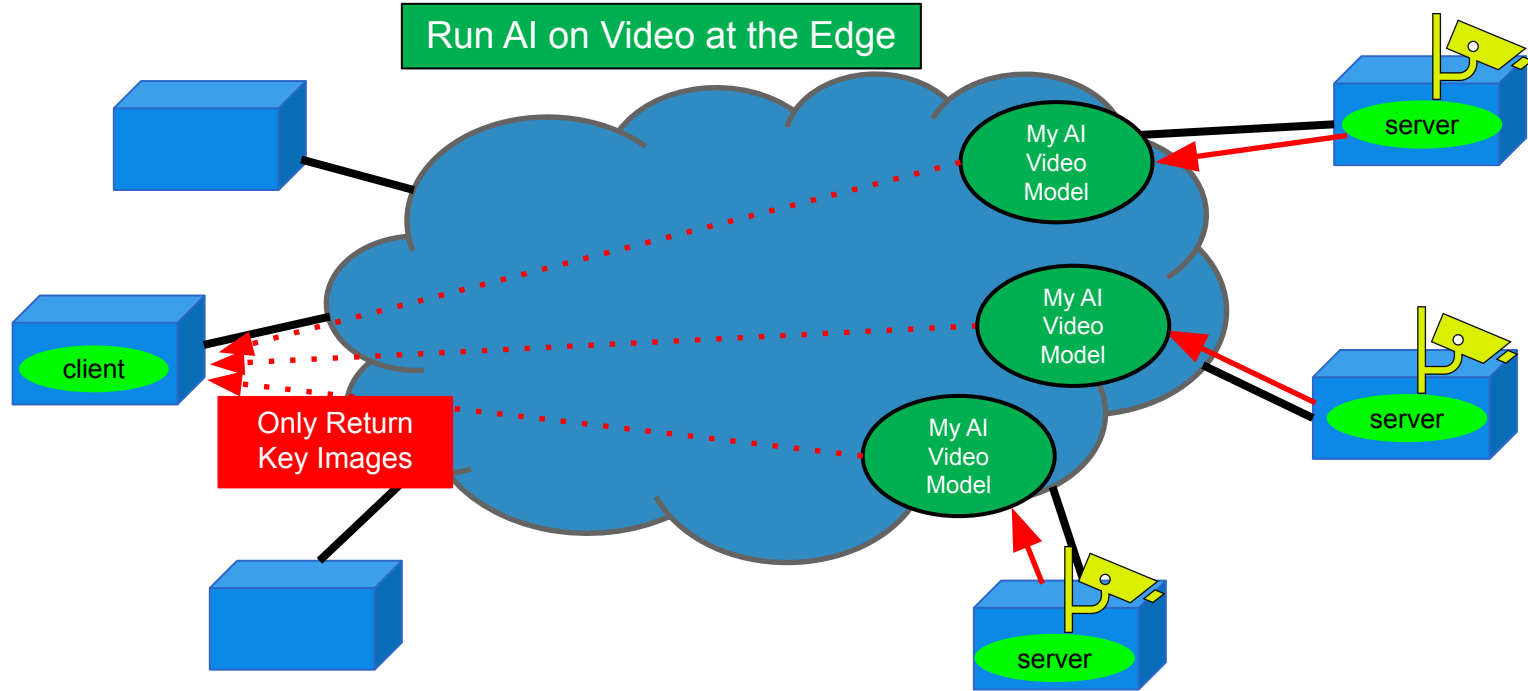
Example: Near Edge Services



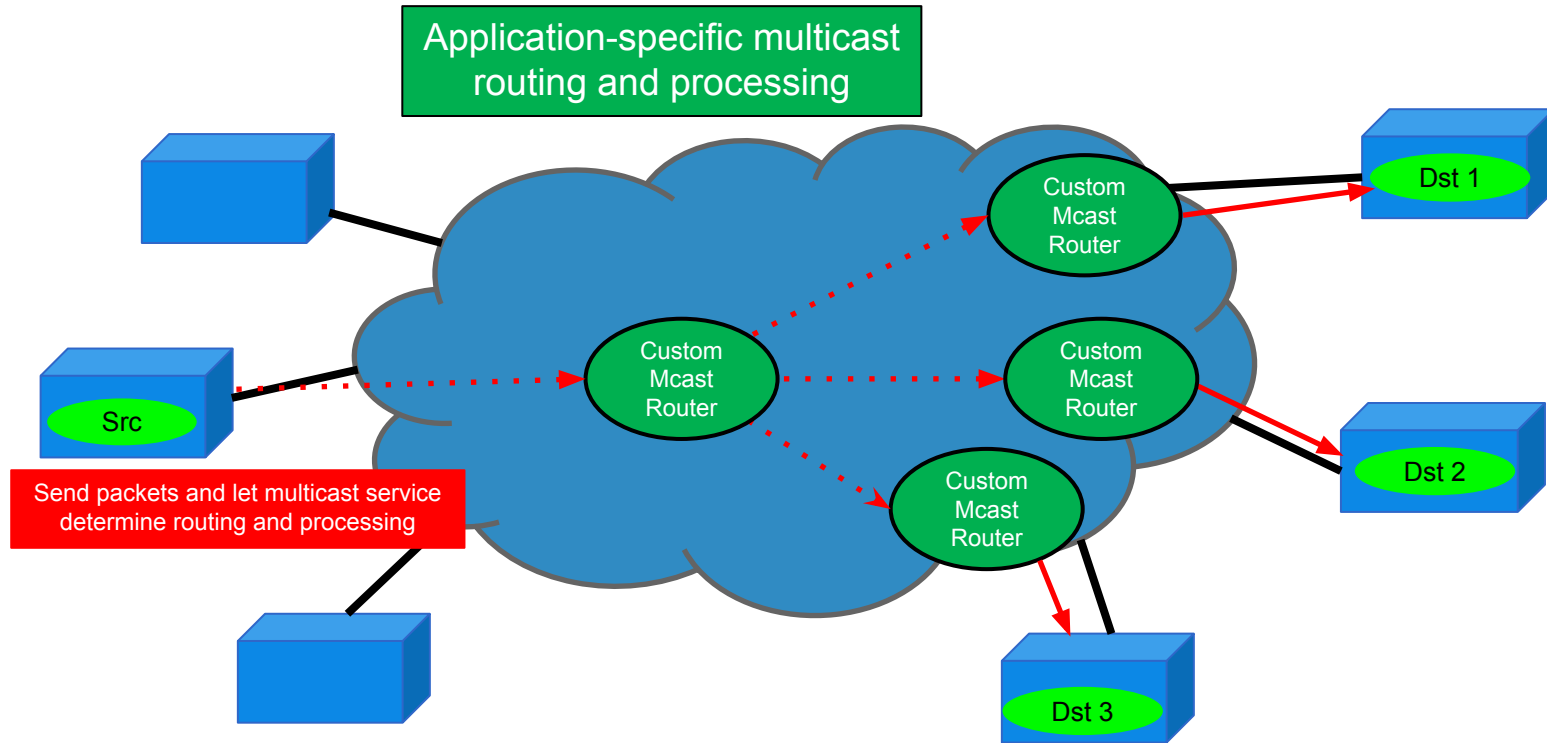
Example: Near Edge AI Services



For Example: Run AI on Public Video Servers

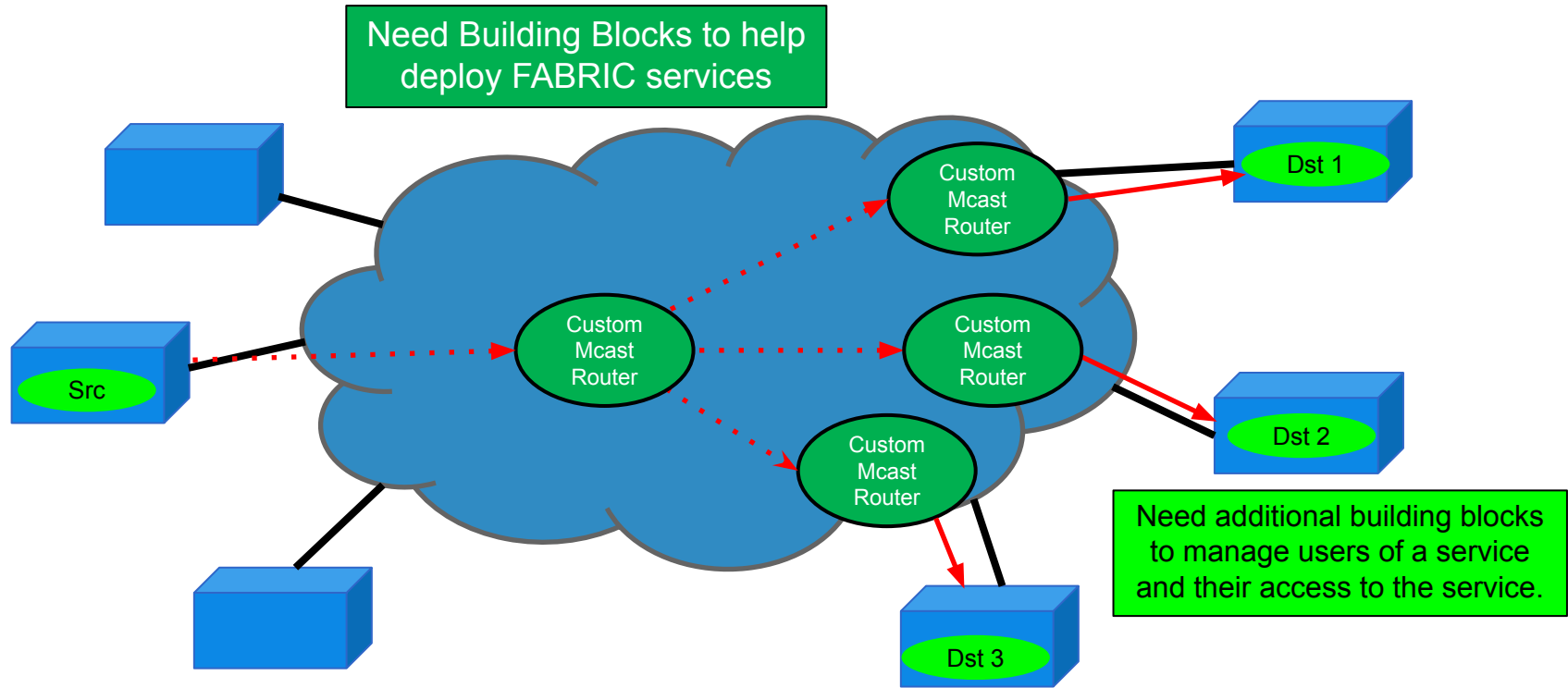


Example: Application-specific Multicast Service



Other in-network examples include Concast, Edge computing/services, Federated Learning, Data filtering, Resilient IoT, Application/Service-specific AI networks, etc

Deploying Services and Managing Access/Users



The Good News

FABRIC enables users to design user-defined network services “from the ground up”, allowing full control

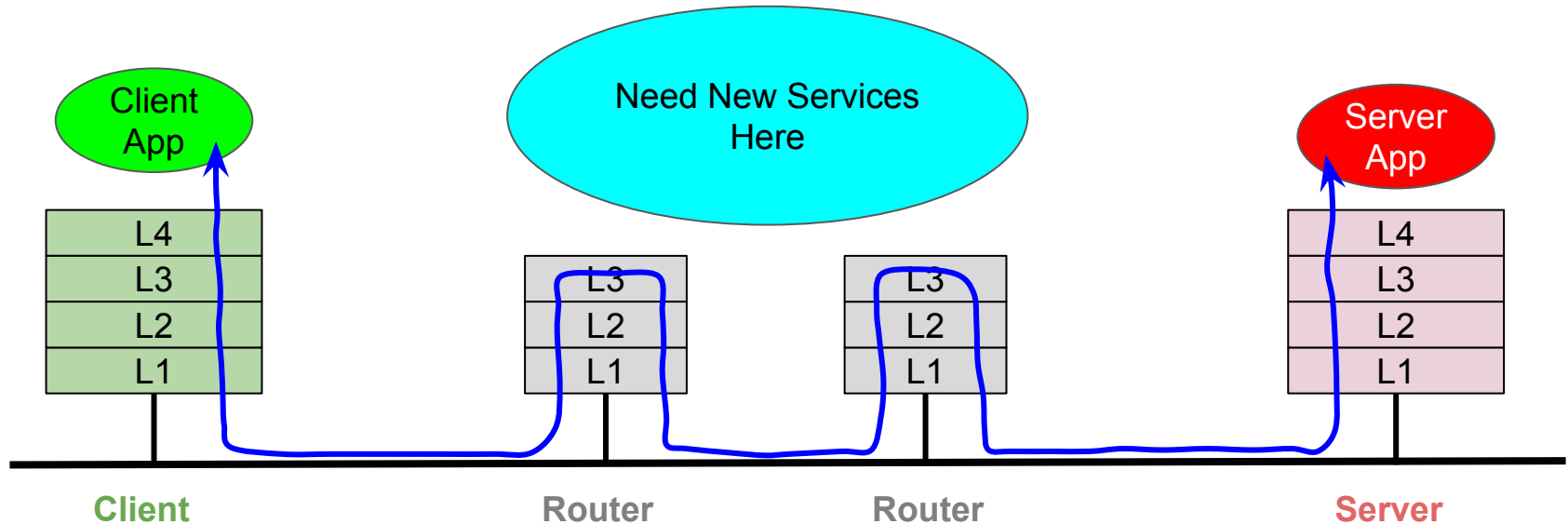
The Bad News

FABRIC requires users to design user-defined network services “from the ground up” to achieve full control

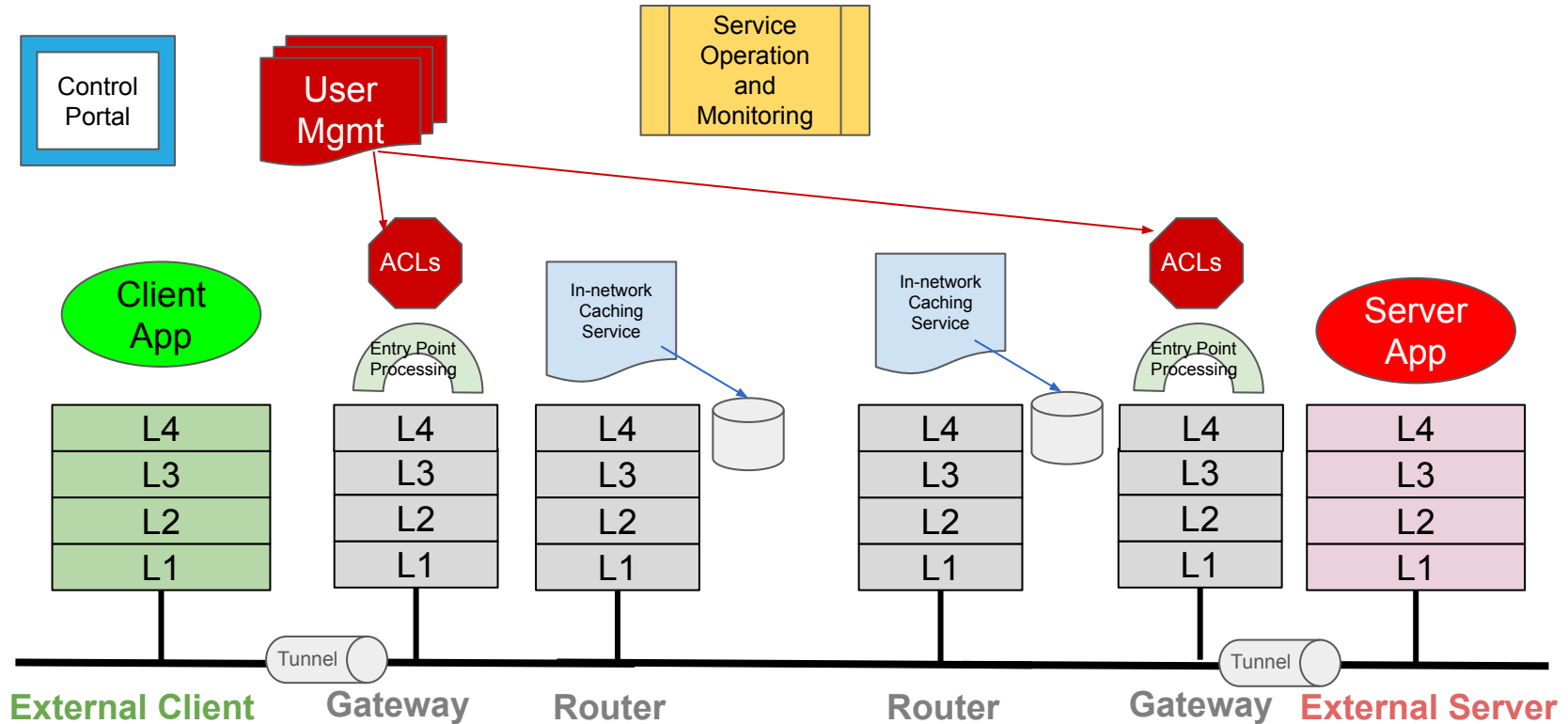
Creating custom services is now possible!! But no one said it was easy.

Need “higher level” building blocks that make FABRIC Service deployments easier.

Example Classic/Textbook Networking Stacks



How hard is it to develop an in-network caching service?



Types of Deployable FABRIC Services

Community FABRIC Services:

- Generally useful services needed by large communities and approved by FABRIC
- Single shared instance per service
- Continually running services
- Instantiated and maintained by a responsive service operator
- Offers some level of service guarantees
- Of sufficient value to consume FABRIC resources that are in demand over long time scales

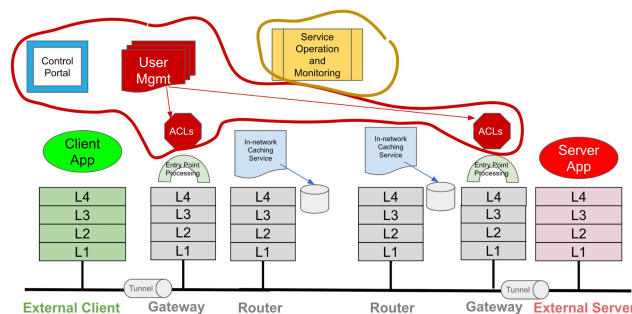
Application-specific FABRIC Services:

- Custom/Tailored services needed by specific applications or experimenters
- May be multiple instances of a service
- Could be long running, but often will be deployed on-demand as needed
- Instantiated and operated by experimenters (users)
- Limited, if any, service level guarantees
- Of value to specific users/communities => must weigh/limit cost of FABRIC resources consumed

Building Blocks for FABRIC Services*

User Interface/Usage building blocks

- “FABRIC Service” **User Accounts** [Coming Soon]
 - Service users need not be FABRIC Experimenters
 - Service users will need to request a FABRIC Portal account
 - Service users will be a special type of FABRIC user denoted in their FABRIC profile
- FABRIC Service **Registration Portal**
 - Service users must register for the services they plan to use
 - Service users will be given a token to use a service
 - No plans, at present, to support “open” services
 - Registration will enable and setup/configure the service for the user



*These services are in various stages of development.
Feedback and comments are welcome.

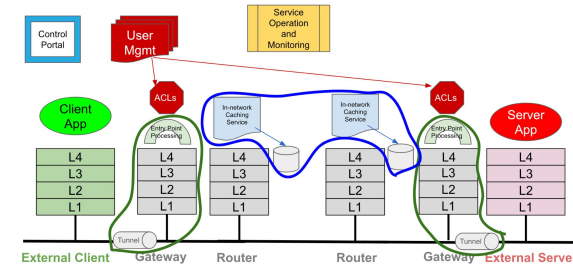
Building Blocks for FABRIC Services (continued)

Application interface building blocks

- **Service Entry Points**
 - Used by (External Internet) Applications to invoke FABRIC services
 - Typically take the form of facility ports providing access to IP address/Port #s
 - Services can be invoked by Packets sent to an entry point (IP address/Port #)
 - Services can be invoked by API calls sent to an entry point (IP address/Port #)
- **Tunneling Service**
 - Automatic setup of a tunneling service between user's machine and entry points
 - Container downloaded to, installed, and run on the user's machine to simplify the tunnel setup process

Application development building blocks

- Long-running slices and resource reservations
- Services packaged as (layered) Artifacts that can be easily deployed in a slice
- Additional services to come



*These services are in various stages of development.
Feedback and comments are welcome.

An Example Community FABRIC Service

The FAB-Cache Service

FAB-Cache Service Motivation

- (Big/Massive) Data drives much of today's research

Researchers from all research domains are now leveraging big data sets for AI, data analysis, simulation/modelling, etc. These big datasets are often (redundantly) requested by many processing nodes on the network. For example the same data may be used by researchers all over the world, or by many nodes in a cluster processing different parts of the data.

- Need smart/intelligent ways to access/transfer research data

Frequent redundant downloads of large datasets (GBs to TBs) from remote repositories strain network bandwidth, increase costs, slow down access times, and overload source servers, leading to availability and performance issues.

FAB-Cache Approach: Distributed In-Network Caching

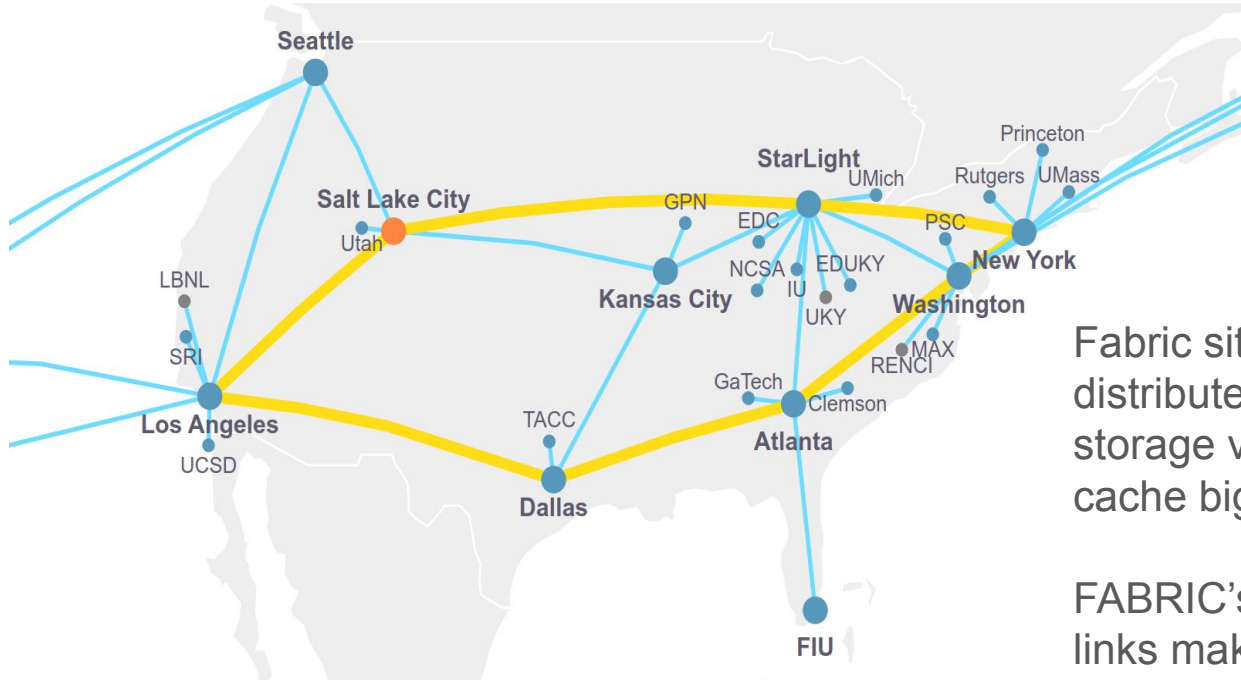
Automatic In-Network Caching:

- Cache data at network routers as data flows from a server to a client
- Network provides data to clients if it has it cached at a router on the path to the server.

Benefits:

- **Reduce Bandwidth Usage:** Avoid redundant downloads.
- **Faster Access:** Retrieve datasets from nearby cached copy.
- **Less Load on Data Providers:** Improve availability of public datasets.
- **Scalability:** Support more users without network congestion.

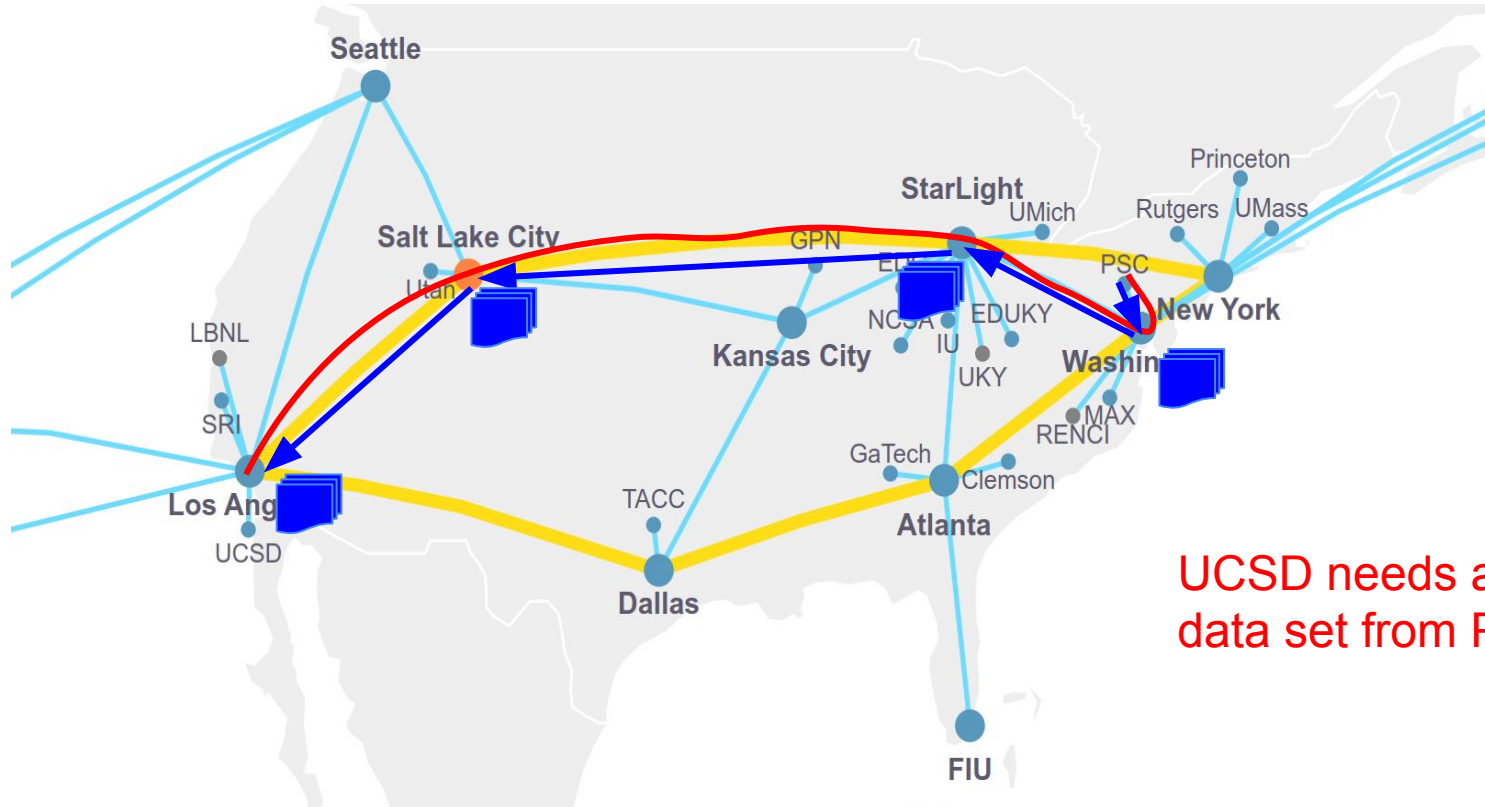
Leveraging FABRIC for In-Network Caching



Fabric sites are geographically distributed and each site has large storage volumes that can be used to cache big datasets.

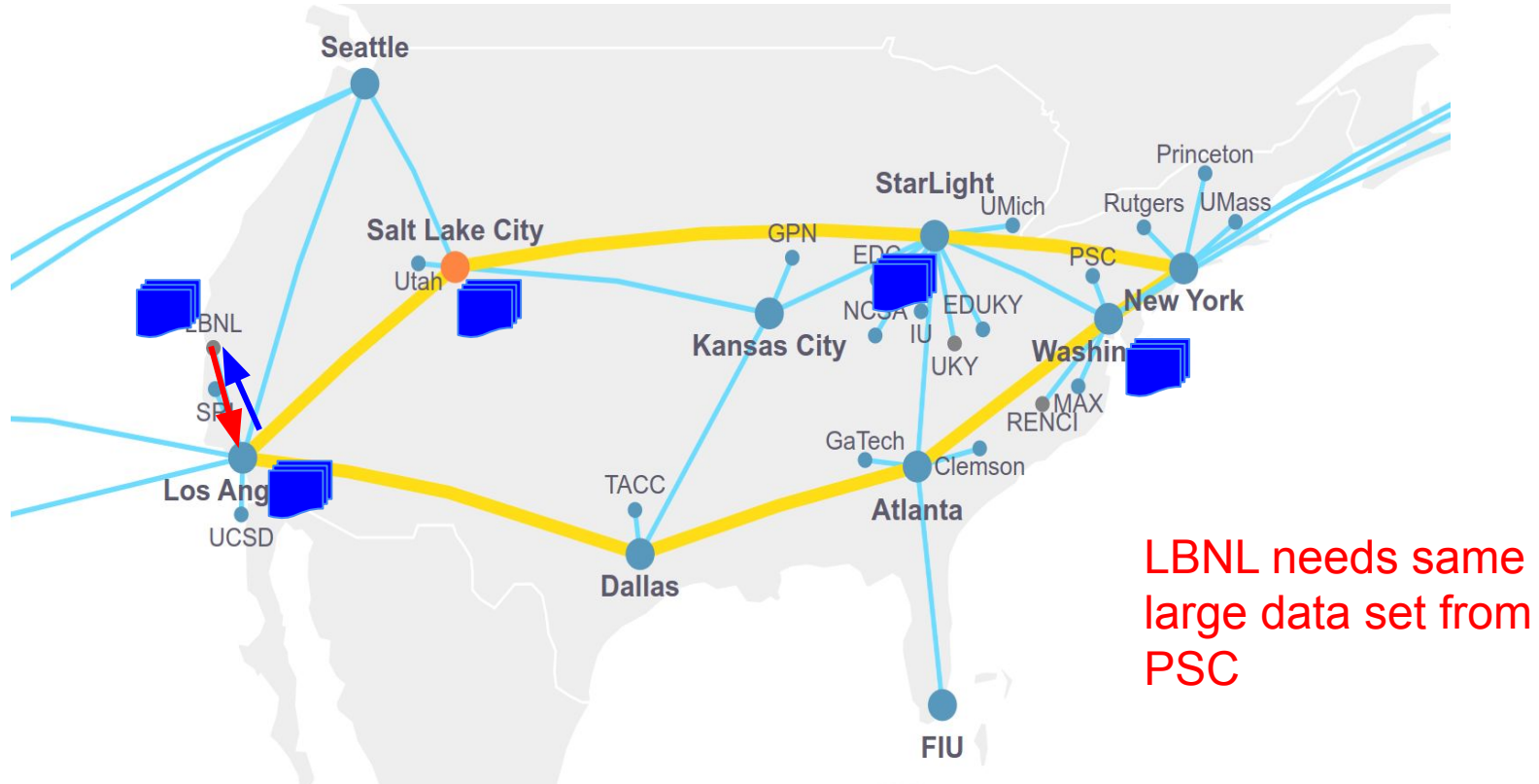
FABRIC's high performance network links make data transfer faster.

Leveraging FABRIC for In-Network Caching

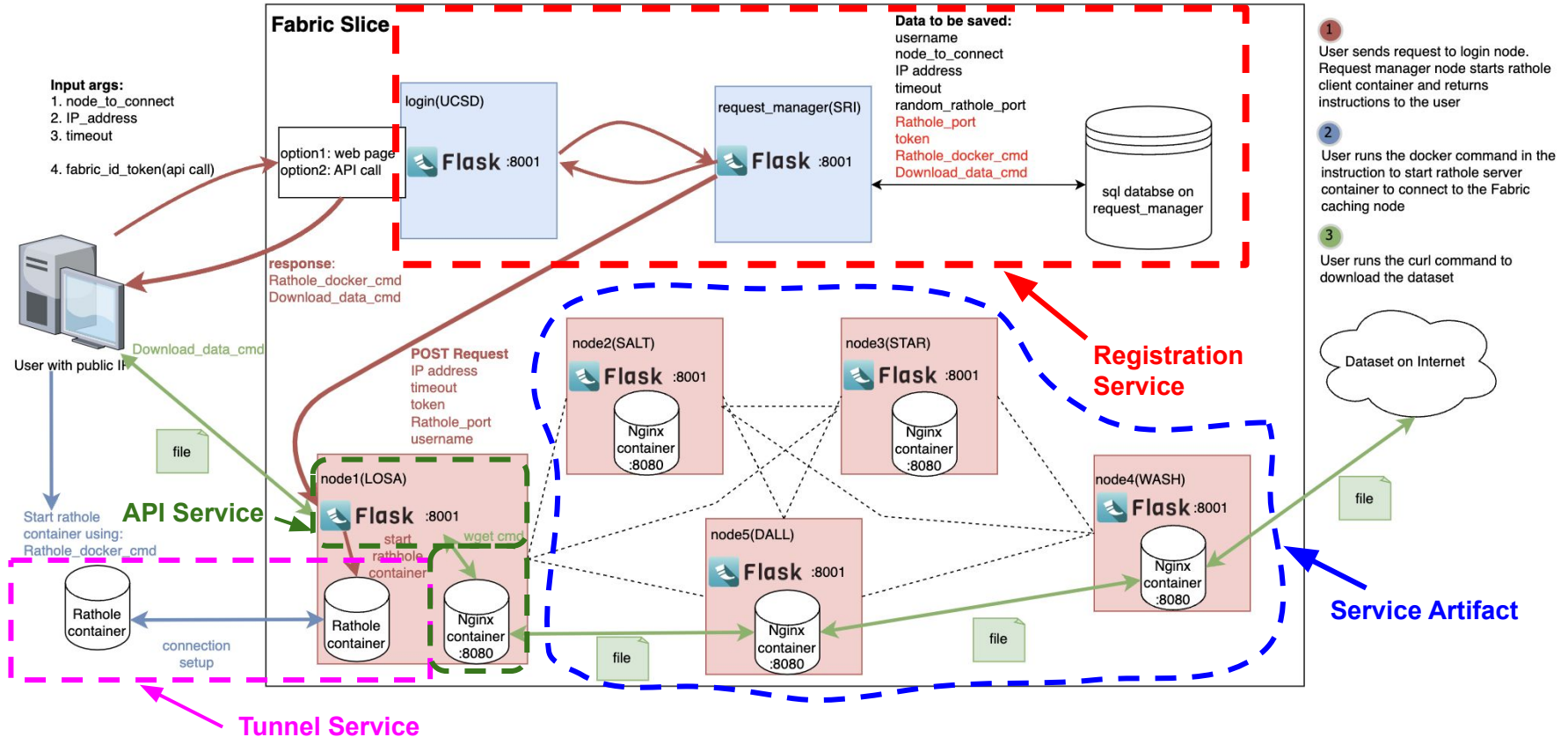


UCSD needs a large data set from PSC

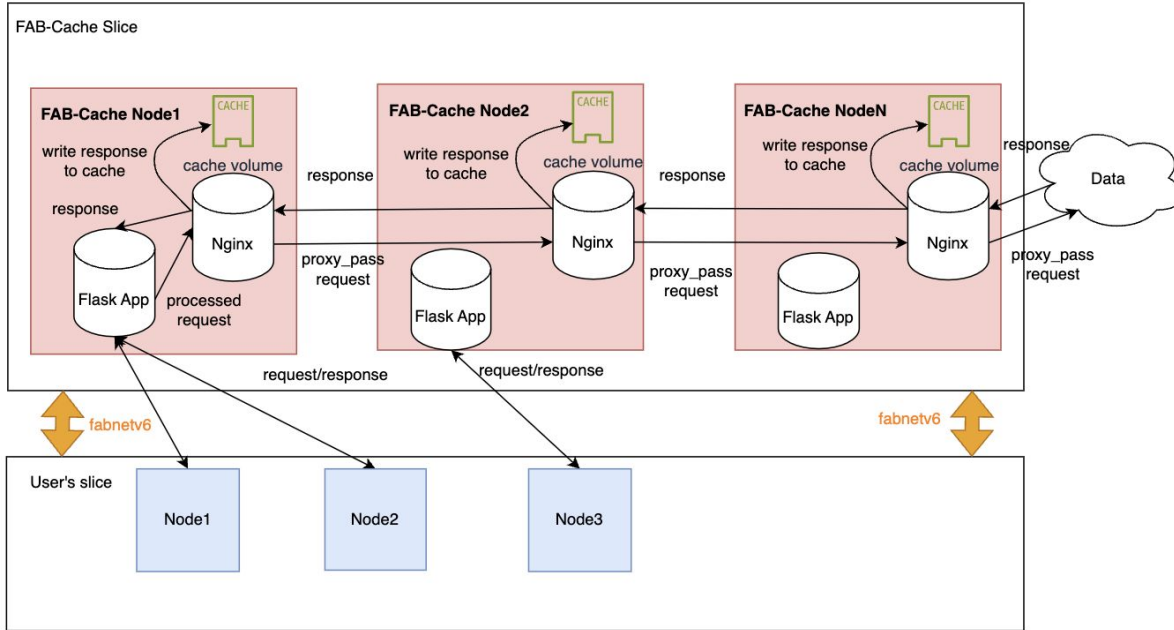
Leveraging FABRIC for In-Network Caching



Fabric In-Network Data Caching Service(FAB-Cache) Design



FAB-Cache Components(Used in this tutorial)



Nginx proxy chain:

Proxy_pass the request to the next nginx container.

Response will be cached to the volume.

Flask app:

User-facing application for submitting requests and check status of FAB-Caching nodes

We will create a fabric slice with fabnetv6 and use the fabnetv6 addresses of the caching nodes for connection. (No Rathole tunnel)

Let's Get Started with the

FAB-Cache Tutorial

(Hands-on exercise omits registration due to time limits)

(We will say more about registration later)

FABRIC Notebooks and Artifacts

- FABRIC experiments are setup and run by Jupyter Notebooks
- FABRIC users can package all the materials needed for an experiment (Notebooks, config files, etc) as [FABRIC Artifacts](#).
- The Jupyter [Notebooks](#) for this tutorial have been packaged as [FABRIC Artifacts](#) that can be found at <https://artifacts.fabric-testbed.net/>.
- You can find Artifacts using any web browser. For example, if you wanted to learn about the artifact for today's tutorial, you could go to <https://artifacts.fabric-testbed.net/> and search for "KNIT 11" and then select "KNIT 11 Creating a FABRIC Service"

We will use Jupyter Hub to download today's tutorial (artifact) and run it.
See the next slide for instructions.

Accessing The FAB-Cache Tutorial Notebook

- 1) Login to FABRIC's Jupyter <https://jupyter.fabric-testbed.net> using the “bleeding edge” server option (although any of the server options will work).
- 2) Open the directory **jupyter-examples-rel1.8.1** (any rel > 1.7.1 will work)
- 3) Open and run the **artifact_manager.ipynb** notebook
- 4) Type “*KNIT 11*” into the **Title:** search field.
- 5) Find “*KNIT 11 Creating a FABRIC Service*” in the results and click the **Download** button.
- 6) The files will automatically be downloaded and extracted to **/home/fabric/work/KNIT_11_Createing_a_FABRIC_Service/vYYYY-M M-DD/tutorial_files** where YYYY-MM-DD is the version.

File Edit View Run Kernel Tabs Settings Help

+

+

+

+

Filter files by name

/ jupyter-examples-rel1.8.1 /

Name

Modified

configure_and_v...

8mo ago

docker_containers

8mo ago

fabric_examples

8mo ago

fabric_ssh_tunn...

8mo ago

artifact_manage...

10m ago

artifacts.json

8mo ago

Changelog.md

8mo ago

CONTRIBUTING...

8mo ago

LICENSE

8mo ago

Readme.md

8mo ago

requirements.txt

8mo ago

start_here.ipynb

8mo ago

artifact_manager.ipynb

Python 3 (ipykernel)

FABRIC Artifact Manager

The FABRIC Artifacts Manager is an invaluable tool for researchers and developers working with the FABRIC testbed. It facilitates the packaging, sharing, and reuse of complete, repeatable FABRIC experiments. By leveraging the FABRIC fablib, users can efficiently manage and interact with experimental artifacts in a streamlined, user-friendly manner. This not only ensures reproducibility but also enhances collaboration by making experiments easily accessible to others.

Accessing and Sharing Artifacts

Artifacts managed through the FABRIC Artifacts Manager can be easily accessed by anyone with the appropriate permissions. Whether you're looking to share your experiment with collaborators or make your work publicly available, the Artifact Manager simplifies this process.

- Public Artifacts:** Any user can read and download public artifacts directly from the Artifact Manager website or via the user interface generated by `ArtifactManagerUI`.
- Project-Specific Artifacts:** Artifacts that are tied to specific projects can also be accessed by project members, ensuring that all collaborators have easy access to the data and resources they need.

This setup ensures that your experiments are not only repeatable but also sharable, fostering collaboration and innovation within the research community.

Code Example

The following Python code demonstrates how to utilize the `FABRIC fablib` and `ArtifactManagerUI` to manage experimental artifacts. This example provides a foundation for interacting with and managing these artifacts directly through a user interface in the Jupyter Hub Environment.

```
[1]: from fabrictestbed_extensions.fablib.fablib import FabLibManager
from fabrictestbed_extensions.ui.artifact_manager import ArtifactManagerUI

artifact_manager_ui = ArtifactManagerUI(fablib=fablib)
artifact_manager_ui.create_ui()

User [redacted] bastion key is valid!
Configuration is valid
Title: KNIT 11
```

Tag: Enter tag keyword

KNIT 11 Creating a FABRIC Service

Learn how to create a long running FABRIC service

example, tutorial

v2025-10-13

Download

Charles Carpen

Hussamuddin N

James Griffioen

Shi

Artifact 'KNIT 11 Creating a FABRIC Service' version: 'v2025-10-13' downloaded to /home/fabric/work/KNIT_11_Creating_a_FABRIC
Artifact 'KNIT 11 Creating a FABRIC Service' version: 'v2025-10-13' extracted to /home/fabric/work/KNIT_11_Creating_a_FABRIC

1 Use Double Arrows to run the notebook.

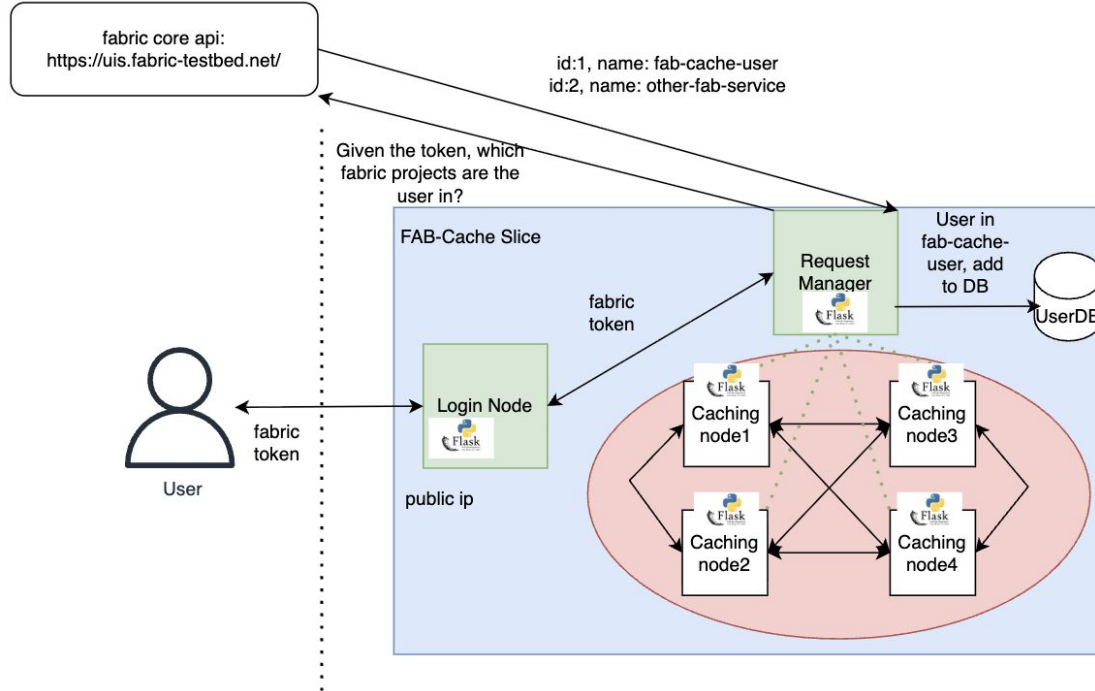
2 Search for KNIT 11

3 Select the latest version and click Download

4 Downloaded files location will be shown here

A Closer Look at the Registration Service

FAB-Cache User Management



(1) We added more components in the FAB-Cache slice:

- **Login node**: with public IP address
- **Request Manager Node**: the backend that handles authentication and user requests.

With public fabnetextV4, users can access the web page on the login node. Any request they make will be handled by the Request Manager Node. This gives **controlled access** to users.

How to Use the FAB-Cache Login Page: Project and token

1. Users request joining the “Service - FAB Cache - Users” project.
2. After being added to the project, go to the FABRIC Credential Manager to generate the token.
3. Once the token is created, you can copy the token in the box below

The screenshot displays the FABRIC web interface. The top navigation bar includes links for Resources, Experiments (highlighted with a red arrow and '1'), Knowledge Base, JupyterHub, Contact Us, About, and Community. The left sidebar shows a list of options: PROJECTS & SLICES, MY SLICES, MANAGE TOKENS (highlighted with a red arrow and '2'), MANAGE SSH KEYS, and ARTIFACT MANAGER. The main content area is titled 'Manage Token' and contains a blue box with the text 'Please consult this guide for obtaining and using FABRIC API tokens.' Below this is a button labeled 'Open FABRIC Credential Manager' (highlighted with a red arrow and '3'). To the right, the 'FABRIC Credential Manager' section is visible, featuring a 'Log out' button and a blue box with the text 'Please consult this guide for obtaining and using FABRIC API tokens.' Below this is the 'Create and List Tokens' section, which includes a 'Select Project' dropdown menu set to 'Service - FAB Cache - Users'. A green box contains the text: 'The default token lifetime is 4 hours. To obtain long-lived tokens for the selected project, please request access from FABRIC Portal.' Below this, there are input fields for 'Lifetime' (set to 4), 'Unit' (set to Hours), 'Comment (10 - 100 characters)' (set to Created via GUI), and 'Select Scope' (set to All). A 'Create Token' button is located at the bottom right of this section, highlighted with a red arrow.

How to Use the FAB-Cache Login Page

Fabric Distributed Cache System

Admin Login

Username

Password

Log In as Admin

OR

FABRIC User Login

FABRIC API Token

Paste your FABRIC API token here

Log In with Token

1. Go to <https://23.134.232.210/login>, paste the token in the box and click “Log In with Token”
2. Or call the API:

```
curl -k -X POST https://23.134.232.210/api/register_with_token \
-H "Content-Type: application/json" \
-d '{"id_token": "<your id token here>"}
```


Questions? Comments?