The basics of file systems

Public clouds

Public clouds

For the purposes of our classes we will define a public cloud as a collection of services (VM hypervisors, DBs, etc.) that one can rent from a company called a cloud provider.

It is a responsibility of a cloud provider to

1. deploy and maintain hardware where the services run,
2. guarantee high availability of services,
3. make those services scalable,
4. provide APIs for managing and using services.

Public clouds

For the purposes of our classes we will define a public cloud as a collection of services (VM hypervisors, DBs, etc.) that one can rent from a company called a cloud provider.

It is a responsibility of a cloud provider to

1. deploy and maintain hardware where the services run,
2. guarantee high availability of services,
3. make those services scalable,
4. provide APIs for managing and using services.

The biggest cloud providers are Amazon AWS, Microsoft Azure and Google Cloud Platform.

We will work with GCP, and will be interested in the following two services:

1. Object storage,
2. Virtual machines.

Object storage

Storage is typically presented as object storage.

An object storage is a storage system that provides a REST API that has at least the following operations:

1. Upload an object (think of “upload a file”),
2. Download an object or a range of bytes in an object,
3. Delete an object,
4. List objects.

Objects are identified by string names. There is no such thing as a directory. An object storage provides only files.

The major cloud providers have their implementations of object storages:

1. Amazon S3: https://aws.amazon.com/s3/,
2. Azure Blob Storage: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction,
3. Google Cloud Storage: https://cloud.google.com/storage.

Object storage

Storage is typically presented as object storage.

An object storage is a storage system that provides a REST API that has at least the following operations:

1. Upload an object (think of “upload a file”),
2. Download an object or a range of bytes in an object,
3. Delete an object,
4. List objects.

Objects are identified by string names. There is no such thing as a directory. An object storage provides only files.

Important differences from file systems:

No random writes. One can only upload a whole object, or replace a whole object.

The operation “List object” is very rarely used and may be slow. This makes object storages easier to scale. A typical scenario is to use object storage as a content-addressed storage (the name of an object is the hash of the content). See

- Git,
- Composefs (https://lwn.net/Articles/933616/).

Content-addressable storage has an added benefit of automatic data deduplication.

Object storage

Object storages have no random writes. One can only upload a whole object, or replace a whole object.

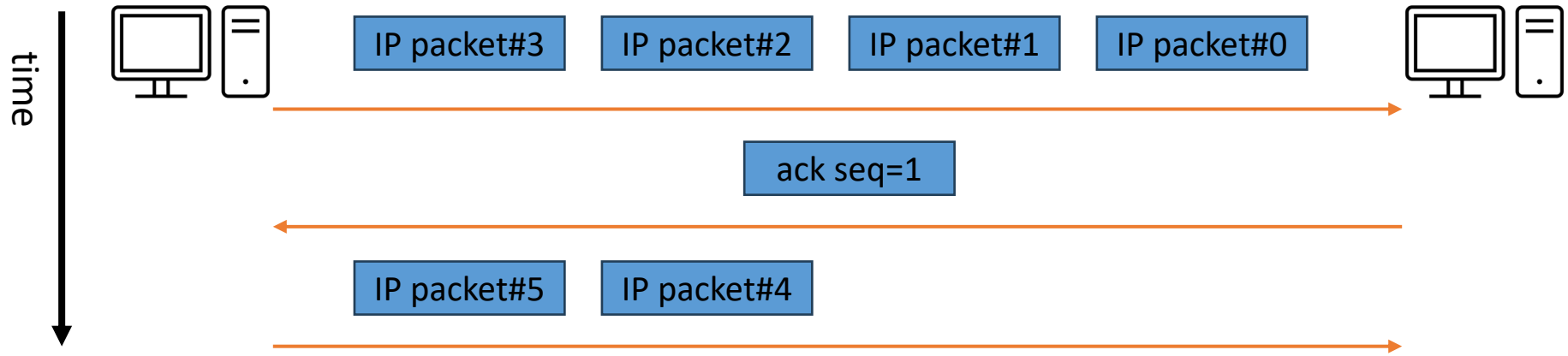
This does not work well with the Internet which is unreliable. Suppose one wants to create an object that is 10G long, and the network connection breaks halfway through. How does one resume an upload?

Uploading big objects

S3 and Azure	GCS
1. `CreateMultipartUpload()`, 2. `PUT Object` to upload parts, possibly in parallel, 3. `CompleteMultipartUpload()`.	1. `POST /upload` to start a resumable upload session, 2. Issue a sequence of PUTs that specify Range: PUTs can't run in parallel, ranges must be adjacent. 3. The last PUT has a special flag that completes an upload.

Bandwidth-delay product and the speed of TCP

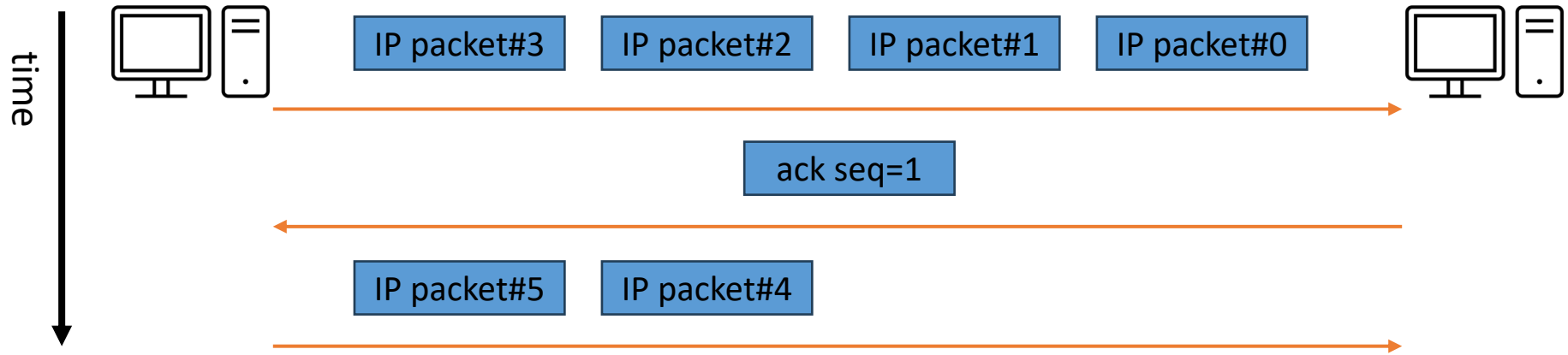
Bandwidth-delay product and the speed of TCP



A host must be able to retransmit every in-flight packet.

Need to keep in memory: (Bandwidth x RTT x 2) bytes.

Bandwidth-delay product and the speed of TCP



A host must be able to retransmit every in-flight packet.

Need to keep in memory: (Bandwidth x RTT x 2) bytes.

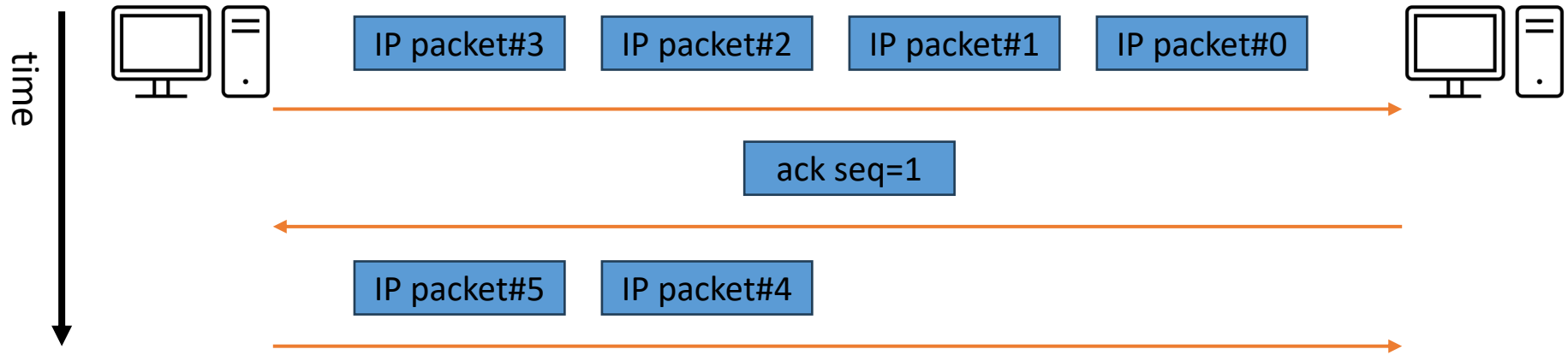
Ex.: RTT between Madrid and Paris is 20ms. A 10Gbps link needs

1.25G/s * 0.02s * 2 = 50MB

of buffering.

Even recent Linux kernels limit the socket send buffer to **6M**.

Bandwidth-delay product and the speed of TCP



iperf between Google DCs in Madrid and Paris:

1 connection → 150MB/sec
2 connections → 326MB/sec
4 connections → 578MB/sec

Even recent Linux kernels limit the socket send buffer to **6M**.

Introductory exercises

Introductory exercises

1. Go to https://cloud.google.com/ and start a free trial.
2. Download and install the gcloud CLI tool: https://cloud.google.com/sdk/docs/install.
3. Learn to configure the environment variable `GOOGLE_APPLICATION_CREDENTIALS`.

4. Create a test storage bucket.
5. Use `gsutil` to upload and download an object.

6. Create a VM:
   a) Choose “e2-micro” as the VM size.
   b) Enable “Allow full access to all Cloud APIs” in the section “Access scopes”.
7. Learn to use `gcloud` to SSH into your VM.

8. Make sure that `gsutil` works in your VM as well.

9. Create two VMs (use “n2d-highcpu-2”) in different GCP regions and
   a) Measure the RTT between the VMs.
   b) Use `iperf` to measure the bandwidth of a TCP connection between these VMs.

10. Shut down your VMs so that they do not consume your credits.

Introductory exercises

1. Read and learn to run a demo at https://github.com/dominiquelefevre/filesystems-101-jb/tree/master/proj/multiconnection/gcs-intro.
2. Read the documentation on “PUT Object” and “PUT Object Part”:
   - https://cloud.google.com/storage/docs/xml-api/put-object-upload.
   - https://cloud.google.com/storage/docs/performing-resumable-uploads.

3. Modify `gcsintro` to perform the following measurements:
   - The speed of “PUT Object”, and make the size of test objects configurable,
   - The speed of “PUT Object Part”, and make the chunk size configurable.
   - **Remark**: when measuring the speed of uploads, also measure the variance of the speed.

4. Use the modified `gcsintro` to measure
   - The speed of “PUT Object” when uploading objects that are 1M, 2M, 4M, … 128M long.
   - The speed of “PUT Object Part” when using the chunk lengths 1M, 2M, 3M, 4M, …, 64M.
   - Find the optimal chunk size for resumable uploads.

5. Measure the speeds in the following configurations:
   - A VM uploads data to a bucket that is located in the same region,
   - A VM uploads data to a bucket in a different region (e.g. from Spain to Belgium or from Germany to UK).