

# Recap

Authentication Protocol based on Asymmetric Crypto System

Principles for Secure System Design:

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Separation of privilege
- Least privilege
- Least common mechanism
- Acceptability
- Defense in Depth

# Design Principle: Layered Trust

A hierarchically designed system has layers of trust

Layers are isolated to limit effects of problems in one layer

Level	Functions	Risk
2	Noncritical functions	Few disasters likely from noncritical software
1	Less critical functions	Some failures possible from less critical functions, but because of separation, impact limited
0	More critical functions	Disasters possible, but unlikely if system simple enough for more critical functions to be analyzed extensively

# Trusted Systems

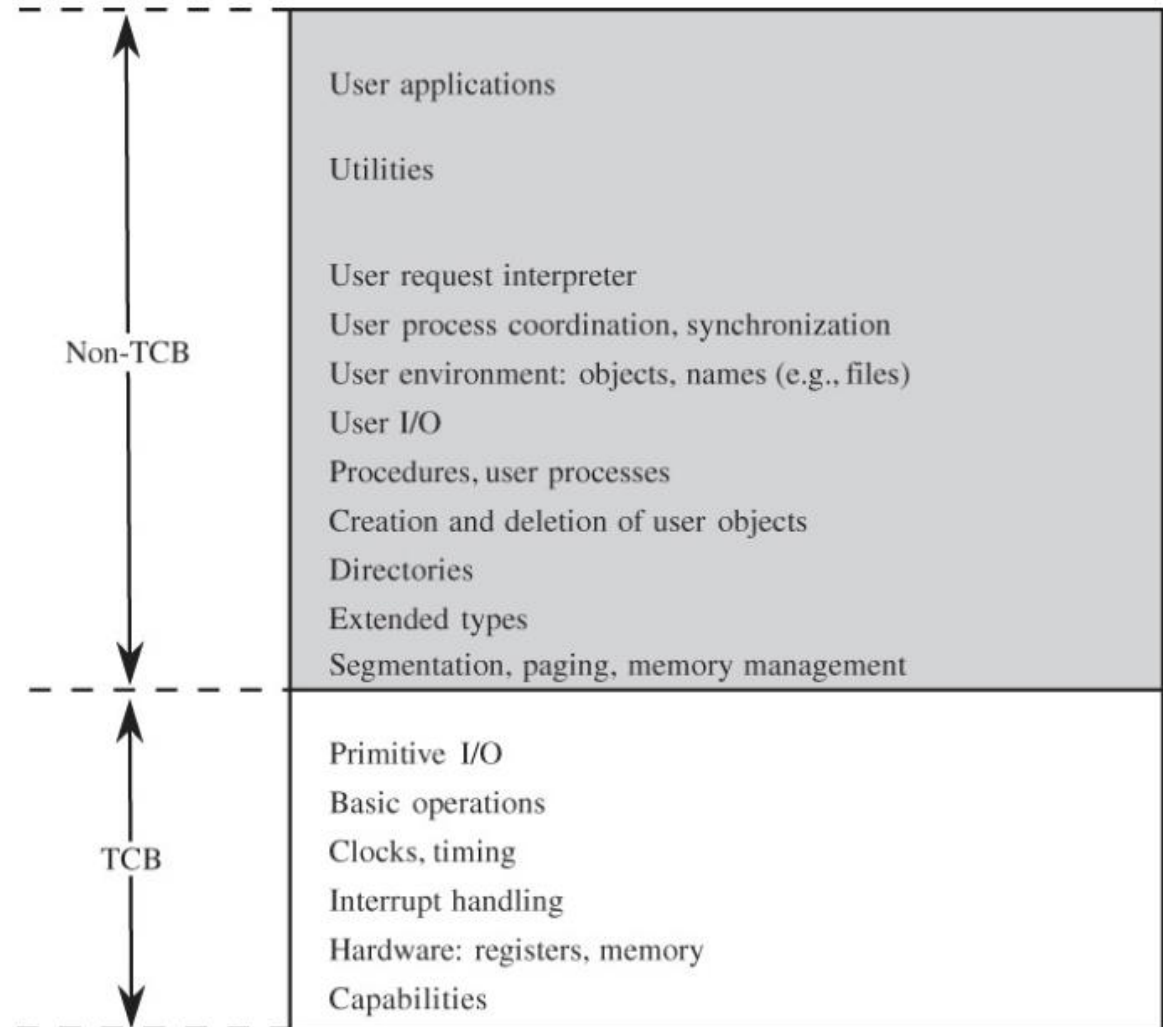
A **trusted system** is one that is relied upon to a specified extent to enforce a specified security policy

Rigorously developed and analyzed, trusted to be tamper-proof, will not execute unauthorized code

Used to act as a kernel or provide critical services (e.g., user authentication, secure boot, digital signature, encryption, etc.)

# Trusted Computing Base (TCB)

A **trusted computing base (TCB)** is the name given to everything in the trusted operating system that is necessary to enforce the security policy



# Trusted Platform Module (TPM)

A **Trusted Platform Module (TPM)** is hardware that controls what can be done on the machine, for example, it assures booting into intended operating system

- Hardware random number generator
- Secure generator of cryptographic keys
- Remote attestation
- Sealing/unsealing

## Other Operating System Tools for Security

**Virtual Machine** – present to the users only the resources they need, giving the user the impression their program is running on its own machine

**Hypervisor (virtual machine monitor)** – software that implements a virtual machine

**Sandbox** – similar to virtual machine or container, a protected environment in which a program can run and not endanger anything else on the system

**Honeypot** – a fake environment intended to lure an attacker

# Access Control

**Access control** is regulating what actions subjects can perform on general objects (e.g., files, tables, hardware, network connections, etc.)

**Objects** are resources on which an action can be performed: files, tables, programs, memory objects, hardware devices, strings, data files, network connection processors, etc.

**Subjects** human users or agents (e.g., processes) that represent users from which objects are protected

**Access mode** are controllable actions of subjects on objects, for example: read, write, modify, delete, execute, create, etc.

# Traditional Unix File System Permissions

Each file/directory has an owning user and group

Read, write and execute permissions are specified for the user, group and all users

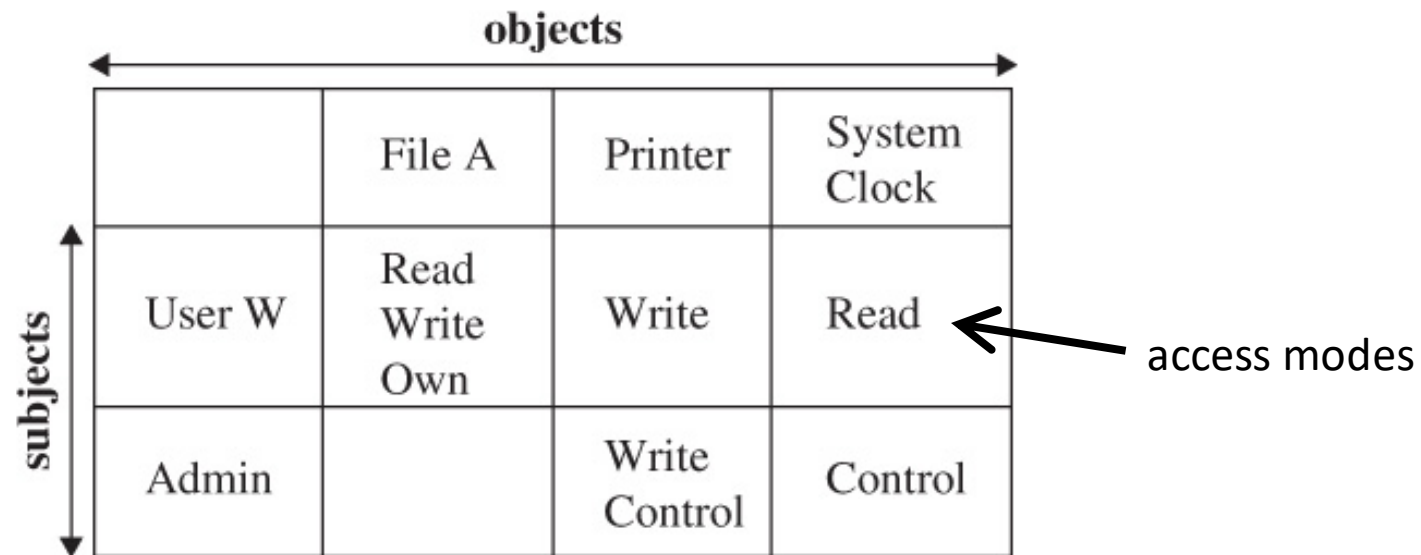
Has many limitations, for example, cannot put file into multiple groups

Symbolic notation	Numeric notation	English
-----	0000	no permissions
-rwx-----	0700	<b>read, write, &amp; execute only for owner</b>
-rwxrwx---	0770	read, write, & execute for owner and group
-rwxrwxrwx	0777	read, write, & execute for owner, group and others
---x--x--x	0111	execute
--W--W--W-	0222	write



# Access Control Matrix

An alternative is the **access control matrix** which describes the **access modes** granted to **subjects** on **objects**



The diagram illustrates an Access Control Matrix. It features a 3x4 grid of cells. The columns are labeled 'objects' at the top, and the rows are labeled 'subjects' on the left. The objects are 'File A', 'Printer', and 'System Clock'. The subjects are 'User W' and 'Admin'. The access modes granted are 'Read', 'Write', 'Own', 'Write Control', and 'Control'. An arrow points from the text 'access modes' to the 'Read' mode in the cell for 'User W' and 'System Clock'.

objects				
	File A	Printer	System Clock	
subjects	User W	Read Write Own	Write	Read
	Admin		Write Control	Control

# Access Control Matrix Example

	Bibliog	Temp	F	Help .txt	C_ Comp	Linker	Clock	Printer
USER A	ORW	ORW	ORW	R	X	X	R	W
USER B	R	—	—	R	X	X	R	W
USER S	RW	—	R	R	X	X	R	W
USER T	—	—	R	X	X	X	R	W
SYS MGR	—	—	—	RW	OX	OX	ORW	O
USER SVCS	—	—	—	O	X	X	R	W

Matrix may be sparse (many empty cells)

# List of Access Control Triples

A less sparse representation of the access control matrix is a list of access control triples

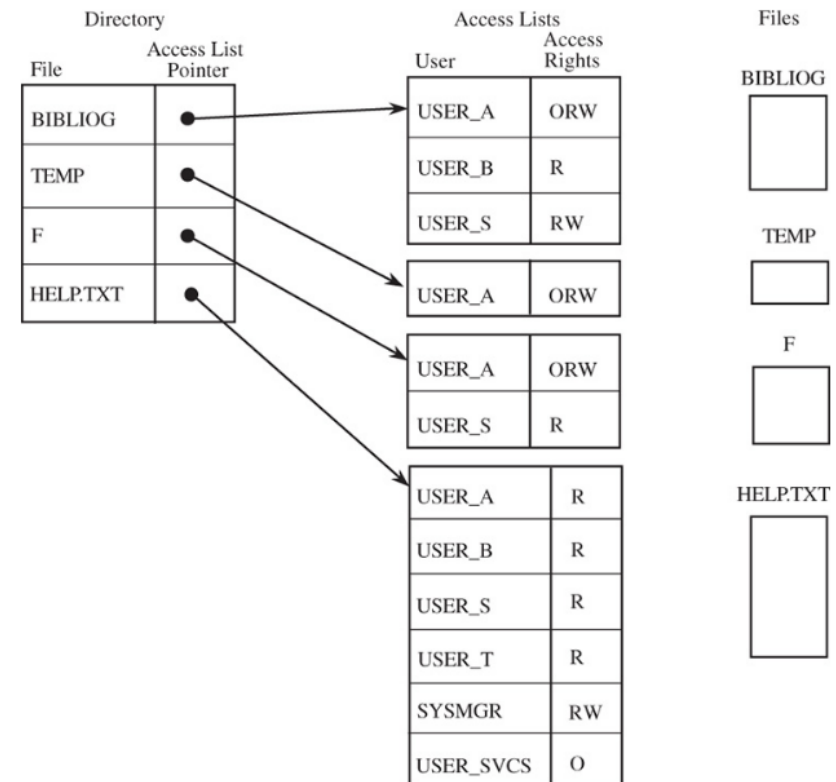
Slow search time, so not often used

Subject	Object	Right
USER A	Bibliog	ORW
USER B	Bibliog	R
USER S	Bibliog	RW
USER A	Temp	ORW
USER A	F	ORW
USER S	F	R
<i>etc.</i>		

# Access Control List

An alternative to the access control matrix is the **access control list**

Each column (object) of the access control matrix is stored in a separate list



# Access Control Matrix vs List

## Access control matrix

Pro: fast lookup for either subject or object

Con: matrix is typically sparse, wasted memory

## Access Control List

Pro: Memory efficient

Con: does not have fast lookup for both subject and object

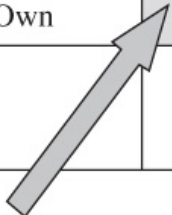
Example: What algorithm could be used to find all files a user has read access to? Consider both access control matrix and list, which is faster?

# Capability

A **capability** is a tuple of (subject, object, access mode), for example: (User W, Printer, Write)

Capabilities are like tickets some OSes use to keep track of what processes are allowed to do

	File A	Printer	System Clock
User W	Read Write Own	Write	Read
Admin		Write Control	Control



# Networking

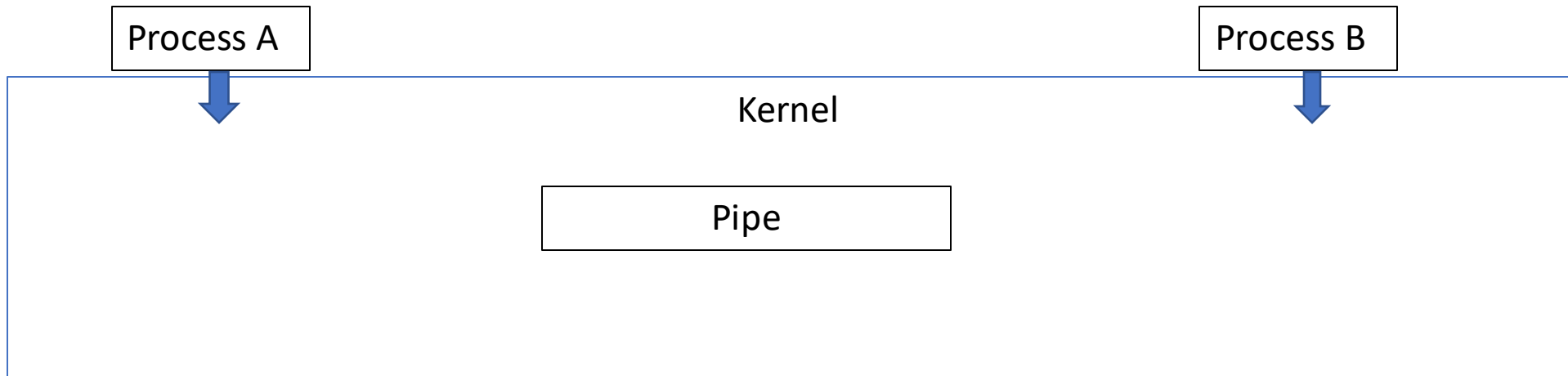
How is computer-to-computer communication implemented?

# Recall: Process-to-process Communication

Logic View



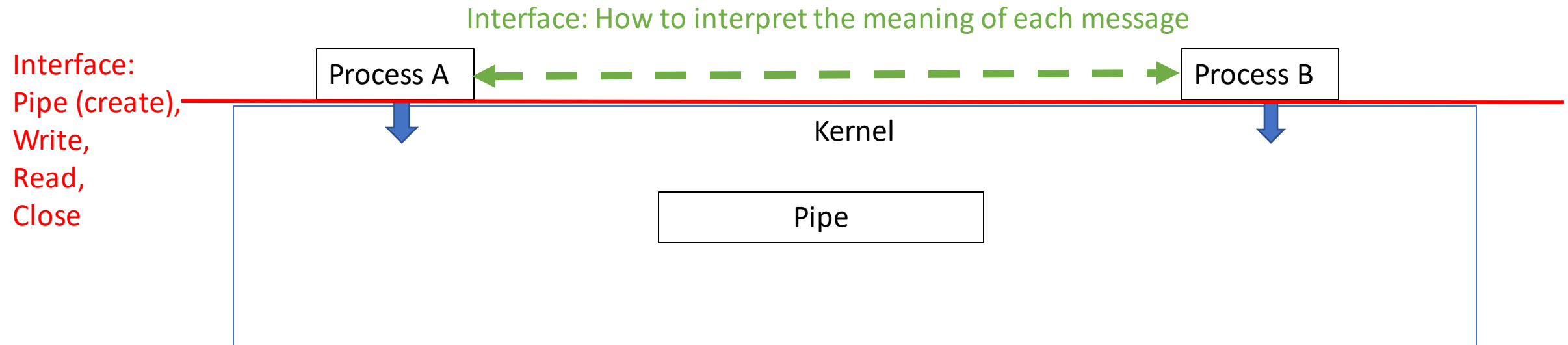
Implementation View





# Recall: Process-to-process Communication

## Implementation View



### Take aways:

- Communication between **peers** (who don't directly interact with each other) are implemented based on **layers** and the bottom layers are able to directly interact.
- Need **between-layer interface** and **between-peer interface**.
- Communication between applications residing on different machines is similar though more complex!

# Analogy: Person-to-Person Communication in Traditional World

From: Alice, 1 Lincoln Way, Ames, IA To: Bob, 2 Michigan Ave, Chicago, IL

Interface: they understand the same language  
to interpret each other's messages

But (suppose) they can't directly  
(physically) reach out to each other

From: 1 Lincoln Way, Ames, IA To: 2 Michigan Ave, Chicago, IL

From: Ames, IA To: Chicago, IL

From: IA To: IL

Human/vehicular Carriers

Rely on  
house  
mailbox  
to  
communi  
cate

Rely on  
house  
mailbox  
to  
communi  
cate

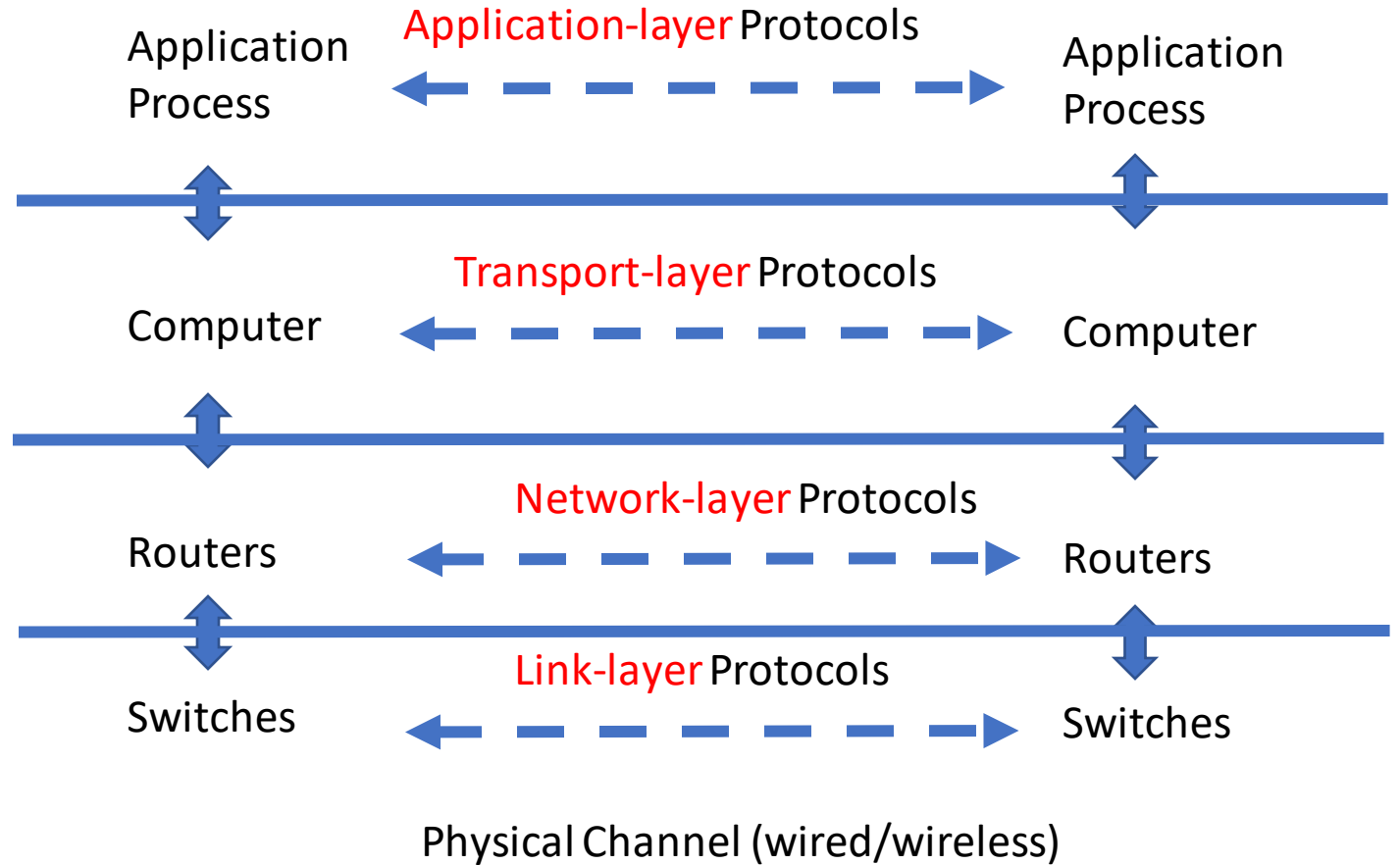
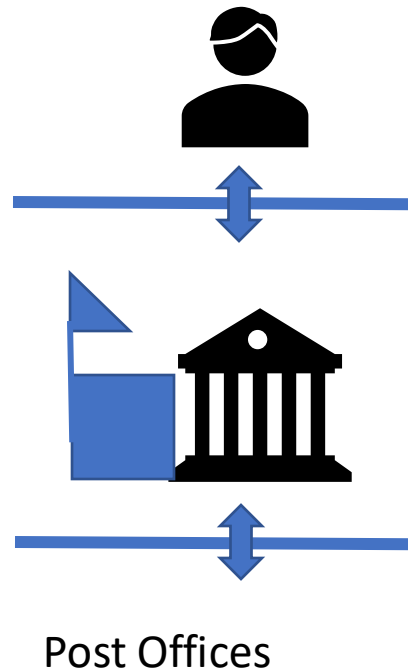
Rely on  
post  
office to  
communi  
cate

Rely on  
post  
office to  
communi  
cate

Post Offices  
(local, regional,  
national,  
international)

Post Offices  
(local, regional,  
national,  
international)

# Mapping to Internet



Note: interface is called protocol

# Simplified Topology of Internet

