T-110.5140 Network Application Frameworks

Overview

Dr. Tancred Lindholm
Based on slides by
Prof. Sasu Tarkoma and Prof. Pekka Nikander

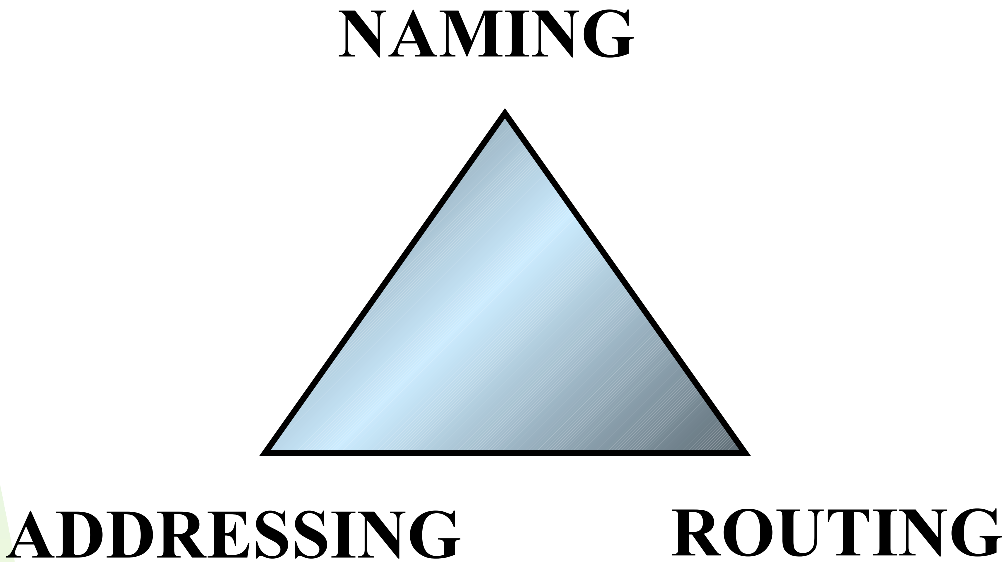
NAF Overview

- Overview
  - Networking: naming, addressing, routing
  - Multi-addressing:Mobility, multi-homing
  - Security: Trust, risks, protocols, keys
  - Objects: Encapsulation, XML, frameworks
  - Performance: bandwidth, delay, bottlenecks
- Connections between aspects
- Examples

Networking

- Communication between distributed entities
- What are network entities?
  - How are they named?
  - How are they connected?
- How are resources allocated?
- Where is state?
  - How it is created?
  - How it is removed?
  - How it is maintained?

Naming, Addressing, Routing



- Fundamental design aspects of a network

Naming, Addressing, Routing

- Naming
  - Simply the name of an entity
  - For example: a domain name
- Addressing
  - The address of an entity
  - For example: geographical location, IP-address
- Routing
  - How to relay messages between addresses
  - Examples: IP-routing, overlay-routing
- Naming, addressing, and routing
  - may be applied on many levels and for different purposes

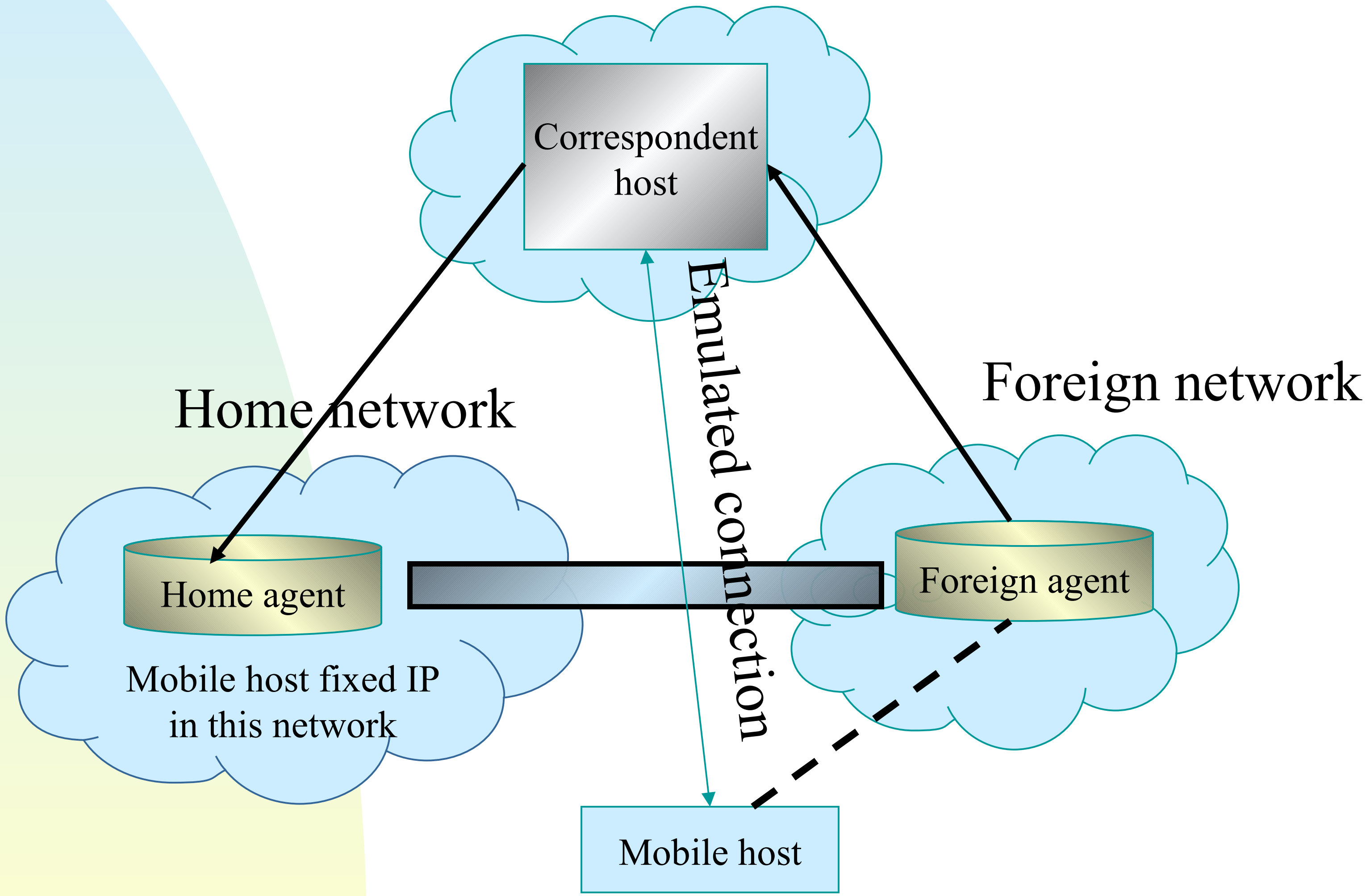
Naming, Addressing, Routing

- For the remainder: we assume that addresses are assigned topologically
  - Prefix-routing (network part, host part)
    - High-level routing between networks
    - Final destination routing inside network
    - More flexibility: CIDR (variable length prefix)
  - Keeps routing tables manageable
    - Addresses depend on location, i.e., addresses sharing network prefix are co-located inside that network

Mobility and Multi-addressing

- Multi-addressing
  - Entities may have multiple addresses
- Mobility requires support for address change
  - In mobility the topological location (access point) changes --> the address changes
- Mobility
  - Mobile nodes
  - Handover terminology
    - make-before-break / soft handover (e.g. 3G)
    - break-before-make / hard handover (e.g. 2G)

Mobility Example:Mobile IP Triangular Routing



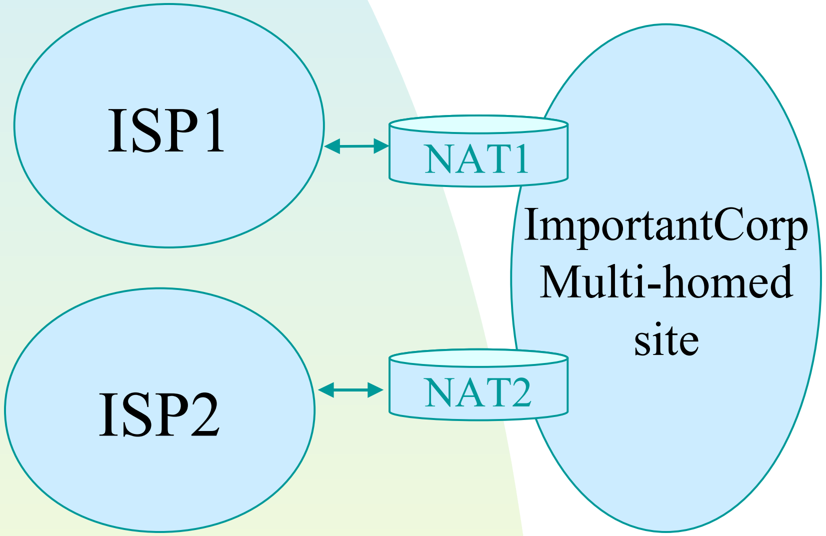
Triangular routing

Multi-addressing

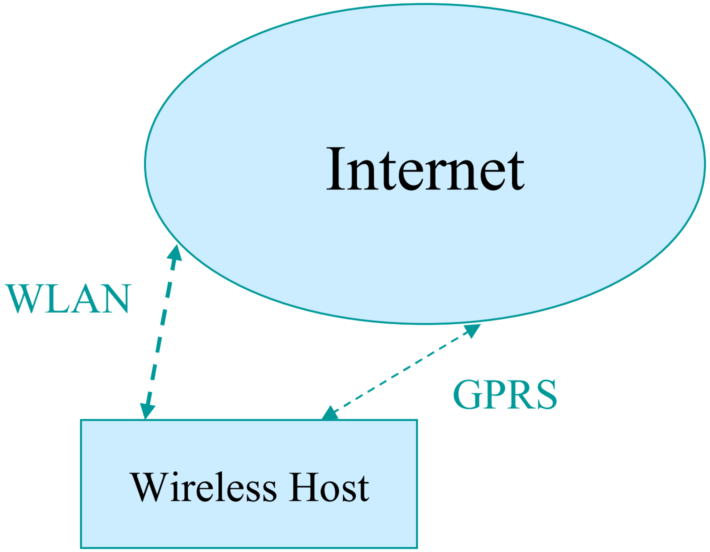
- Multi-homing
  - "on multiple home networks"
  - Server has multiple addresses on different networks for increased reliability
  - Client has multiple addresses
- Multi-homing supported by active address change (to switch link)
- Topology change can cause renumbering
  - From old prefix to new prefix
  - Changing the IP host addresses of each device within the network
  - Related with multi-homing and must be supported by mobility protocols

Multi-homing Examples

Site multi-homing



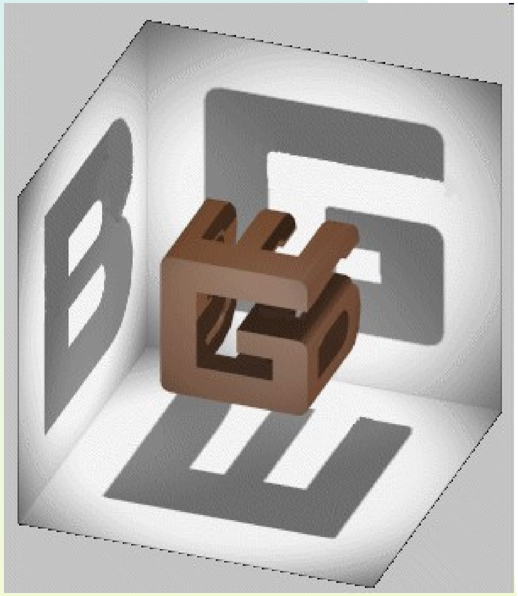
End-host multi-homing



Multi-layer Operation

- Mobility and multi-homing can be realized on different layers
  - Network (Mobile IP)
  - Between network and transport (HIP)
  - Transport (SCTP)
  - Application (SIP, overlays)
- Best case: mobility / multi-homing solved on one layer
- Worst case: mobility / multi-homing resolved on each layer

Facets of Distributed Systems



- There are many, many ways to look at a distributed system (and systems in general)
- Let's skim some of these (and return later)

User POVs of Distributed Systems

- User view point
  - Services that work
  - 24/7, anywhere
  - Usability, security
- Developer view point
  - Easy to develop and debug
  - Fast time-to-market
- Administrator view point
  - Easy to deploy and maintain
  - Scale well
  - Secure

Security

- Requirements
  - Confidentiality
  - Authentication
  - Authorization
    - Rules, policies, ACLs
    - ticket-based schemes
  - Non-repudiation
  - Auditing and logging
  - Availability

Security

- Physical network operated by many parties
- Not all operators can be trusted
- Protecting subnets
  - Firewalls, NATs, middleboxes
    - Connectivity problems
- Need for cryptographic protection
  - Integrity and confidentiality of data
  - Identification, access control, and authorization
  - Key distribution and trust creation/evaluation

Programming with Objects

- Information hiding for programmers
- Extend a familiar paradigm to a distributed environment
- Cracks in the centralized OO model
  - Huge difference in latency
  - completely different fault semantics
  - synchronization problems
- How to name and find objects outside a single memory space?
- Using services provided by third parties?

Performance

- Network Quality of Service (QoS) characteristics
  - End-to-end latency
  - Bandwidth
  - Jitter matters
- A dynamic phenomenon if packet switched
  - Congestion leads to drops or delays
- Different paths have different QoS properties
- Two worlds: wireless and wired

Delay and failure model matters

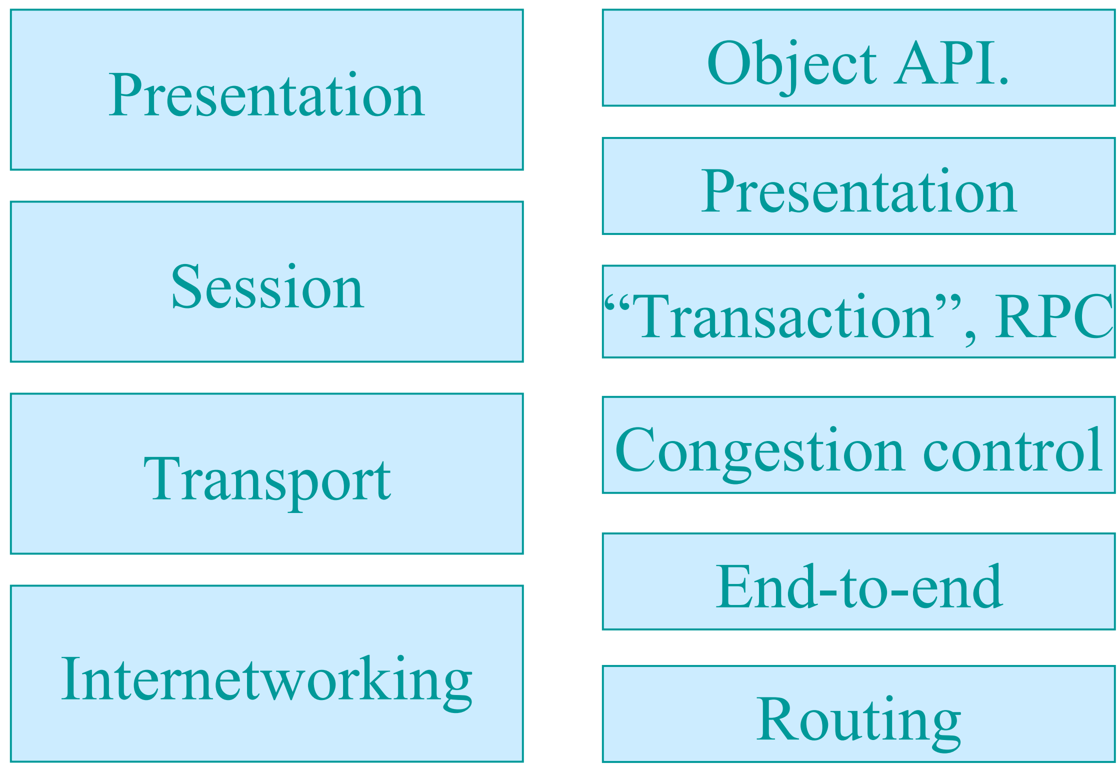
- A single process
  - succeeds or fails
  - method call takes nanoseconds
  - all-or-nothing delay and failure model often adequate
- In a network,
  - round trip latency may be ~ 100ms
  - end-nodes may fail rather often → fail complete process?
  - a path between two end-nodes may fail
  - performance may fall to unacceptably poor level

System Models

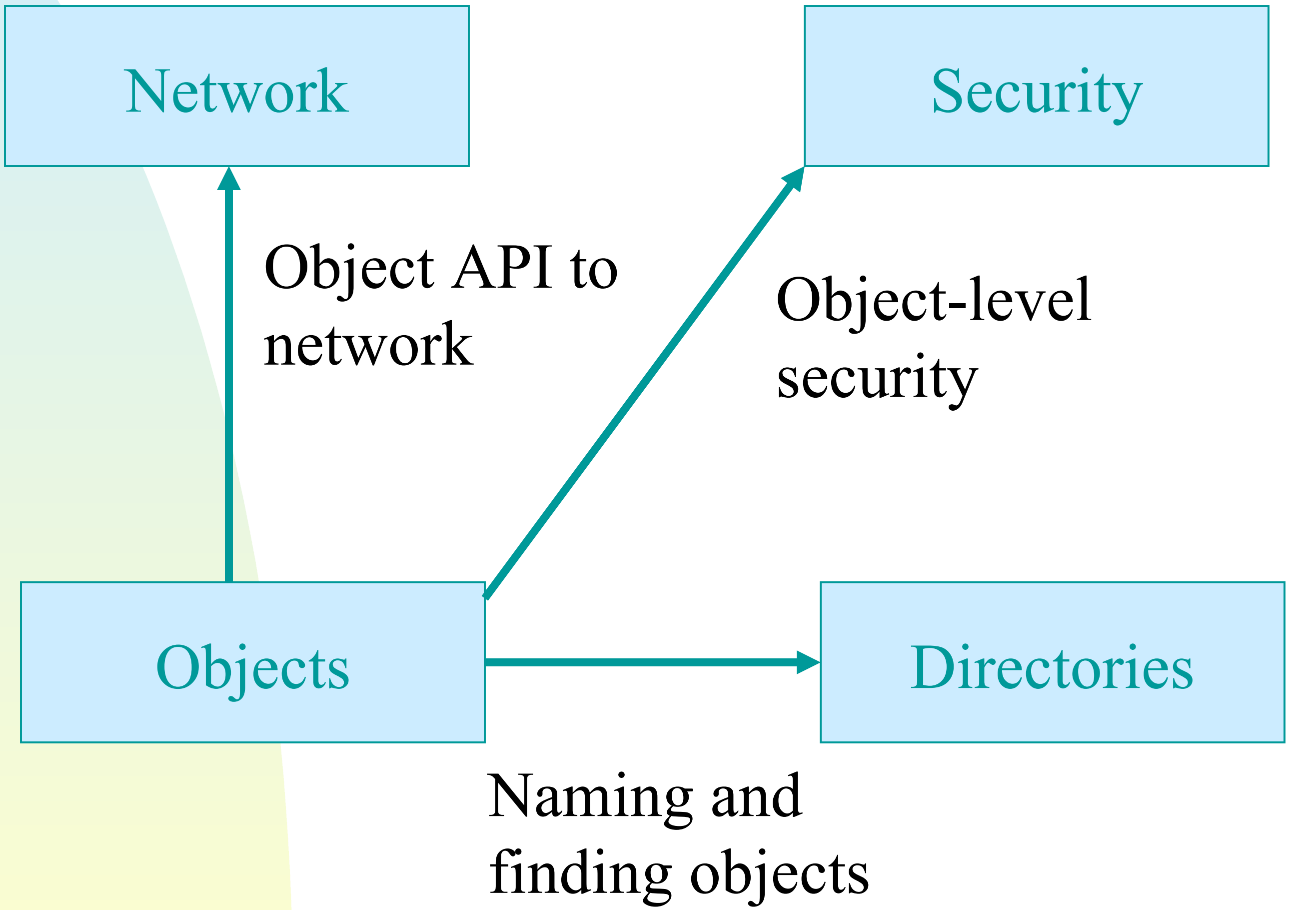
- Layered model
- Object centric view
- Network centric view

Layered Model

- No unambiguous layering



Object centric view (High Level)



Naming and finding objects

- Objects need to have out-of-process names (i.e., not just memory addresses)
  - Each type (class) needs to have a name
  - Each method (action) needs to have a name
- Objects may be mobile, replicated, ephemeral, or permanent(persistent)
- How to find an object in the network?
- How to maintain a consistent view of types when they evolve?

Providing an Object in the Network

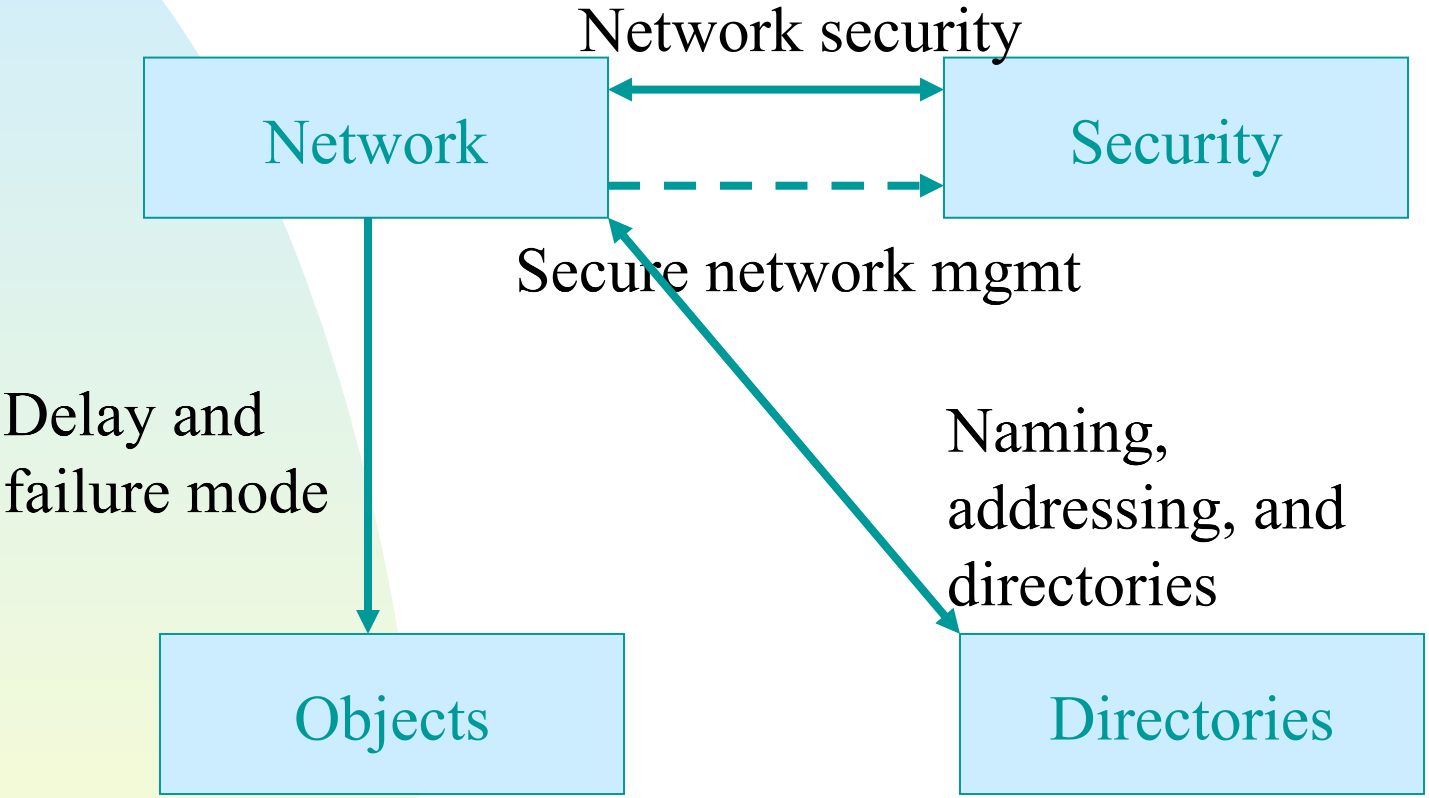
- Mostly a matter of providing naming
  - How to find the object?
  - How to find type and method meta-data?
  - How to refer to remote objects?
- Also
  - How to move objects over the network?
  - How to synchronize replicated objects?
  - Abstraction of delay and faults

Object Security

- Objects represent reactive data storage
  - May implement access control logic
- Threads of execution act upon them
  - <Thread ID, Call History> has {permissions}

- How to trust a remote node?
- How to represent permissions over the network?

Network centric view (Lower Level)



Network security

- Large networks are physically vulnerable
- Cryptography for integrity and confidentiality
  - Need to solve the key distribution problem for authentication
- Not everybody is equally trusted
  - Need to have identities and credentials
- Security and Availability
- Security and Privacy
- Balance: security vs. ease of administration vs. performance

Naming, addressing, and directories

- Network entities are named
  - DNS names: www.example.org
- Names need to be translated to addresses
  - Network only how to forward to an address!
- A directory provides translation information
- Avoid mutually dependent design
  - Make sure that basic networking works even without directories.

Course Concepts in the Context of Examples

- Networking: IPv4 and IPv6
- Directories: DNS
- Security: IPsec and IKE
- Objects: Java RMI

Networking: IPv4 and IPv6

- Hosts named and addressed by IP address
- IP addresses are assigned topologically
- Forwarding tables
  - Created by routing protocols
  - Converge time: minutes
- Two broad classes of state:
  - Routing and forwarding tables
  - End-to-end state

Concerns with IP networks

- Not end host state, not really routing state either: NAT
  - Should always be soft state to protect resources
- Congestion control, reliability, packet drop / retransmission, flow control
  - Traditionally handled at the transport layer
- Routing hardware design and cost
  - Complexity of next-hop lookup
  - QoS facilities, queues, traffic shaping

Directories: DNS

- Provides Domain Name to IP address mapping
  - Hosts are no longer named with IP addresses
- Replicated, hierarchical repository
- Data cached at edge hosts
  - Reduces traffic
  - Long-lived caches make update distribution slow (short lived would DOS the system)
- Partitioned into administrative domains
- Relatively poor security
  - Mostly relies on manual configuration

Concerns with directories

- Actual data storage and structure
  - Logical structure, i.e., architecture
  - Architecture + hardware structure = performance level
- Partitioning, replication, caching
- Access control: reading, modification
- Representation of relationships
- Representation of objects

Security: IPsec

- IP Security (IPsec)
- End-to-end, below congestion control
  - Authentication Header (AH)
    - Integrity and authenticity (immutable IP header+payload)
    - Problems with NATs (dst mutable)
  - ESP (Encapsulating Security Payload)
    - Transport-mode: higher level payload
      - host-to-host, IP headers not encrypted
    - Tunnel-mode: payload is IP packet
      - network-to-network, inner packet encrypted
    - Mostly in tunnel mode, VPNs
- Contains a complex policy control model

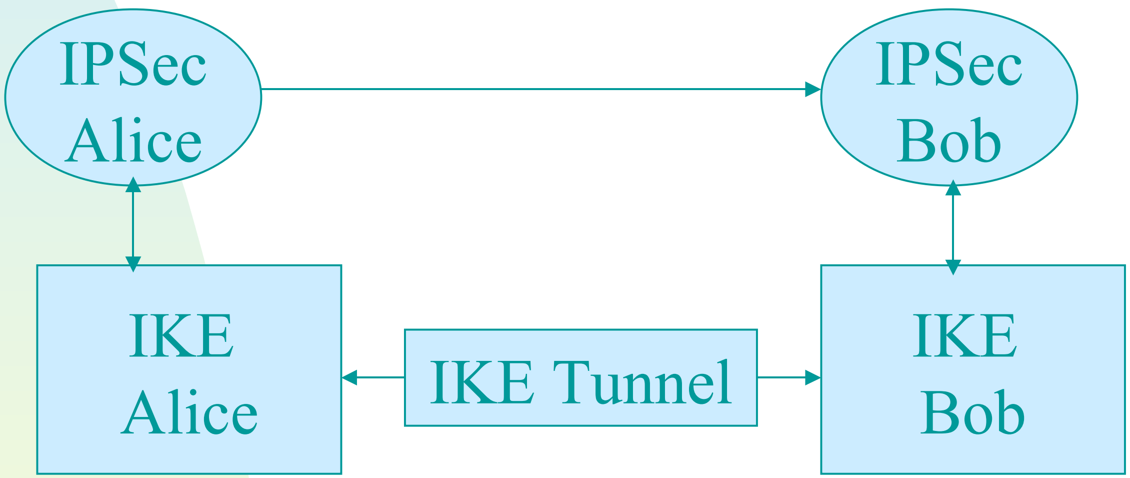
IKE

- IPSec separates key management into IKE
- *Security Association* (SA)
  - relationship between two or more entities that describes how the entities will use security services (algorithm, key) to communicate securely
- *Internet Key Exchange* (IKE)
  - negotiates the IPSec security associations (SAs)
  - negotiates the security association for IPSec
  - authentication, establishment of shared keys

IPsec and IKE

1.No IPSec SA for Bob

4. Protected packets are sent to Bob



2. Alice’s IKE starts negotiations

3. Negotiation completes.

IPSec SAs in place

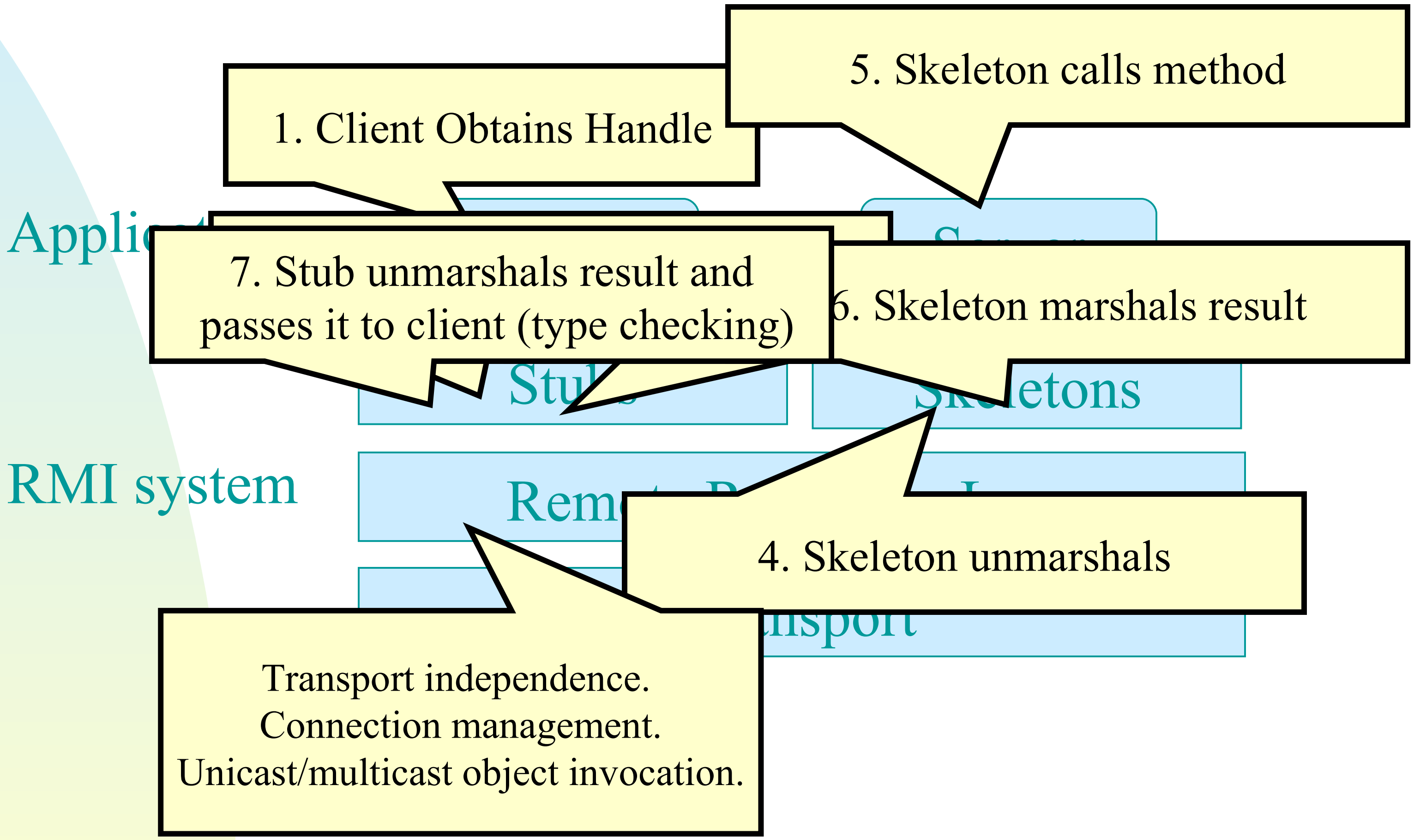
Concerns with security

- Right layer to implement?
  - E.g. crypted transport won't solve partial doc access
- What about multi-layer security?
- Privacy? DoS protection?
- Trust management?
  - How to bootstrap trust?
- Authorization and credentials?

Objects: Java RMI

- Java Remote Method Invocation
  - Objects in one VM invoke objects in a remote VM
- Remote handle = "out of process pointer"
  - From the registry name facility
  - By receiving the reference as an argument or return value of a method call
- Client needs stubs
  - stubs are proxy object in the remote VM taht forward calls over the network
- Reflection adds interesting semantics
  - Run-time dispatch instead of compile time, late binding
  - Parameters of method calls are passed as serialized objects (deep copy)

Objects: Java RMI



Concerns with objects

- Object Discovery
  - Representation of Object Handle
- Reliability and disaster recovery
  - Fault tolerance: Replication, Consistency, etc.
  - Version detection
- Marshalling and unmarshalling
- Transparency vs. efficiency
  - Easy to destroy perf if RPC is transparent
- Performance
  - Easy to send more data than necessary!
  - Distributed garbage collection
- Supporting heterogeneous environments

Summary

- Networking: naming, addressing, routing
- Multi-addressing: mobility, multi-homing
- Security: Trust, risks, key distribution
- Objects: naming, representation
- Performance: bandwidth, delay, bottlenecks

Questions / discussion