Using the Network Layering DAG
CS 118
Computer Network Fundamentals
Peter Reiher

Outline

- The DAG
  - A closer look at tables
  - A closer look at directed links
  - A closer look at FSMs and their state
  - Optimizations and equivalences
- DAG walks

Reminder

- This is an *conceptual approach*
  - There's no common implementation approach
  - Most code is designed as a fixed structure
    - And what are shown as a separate tables and FSM state is often mangled together
  - There are programmable communication systems
    - That COULD be programmed like this
    - E.g., Netgraph, Click

Another reminder

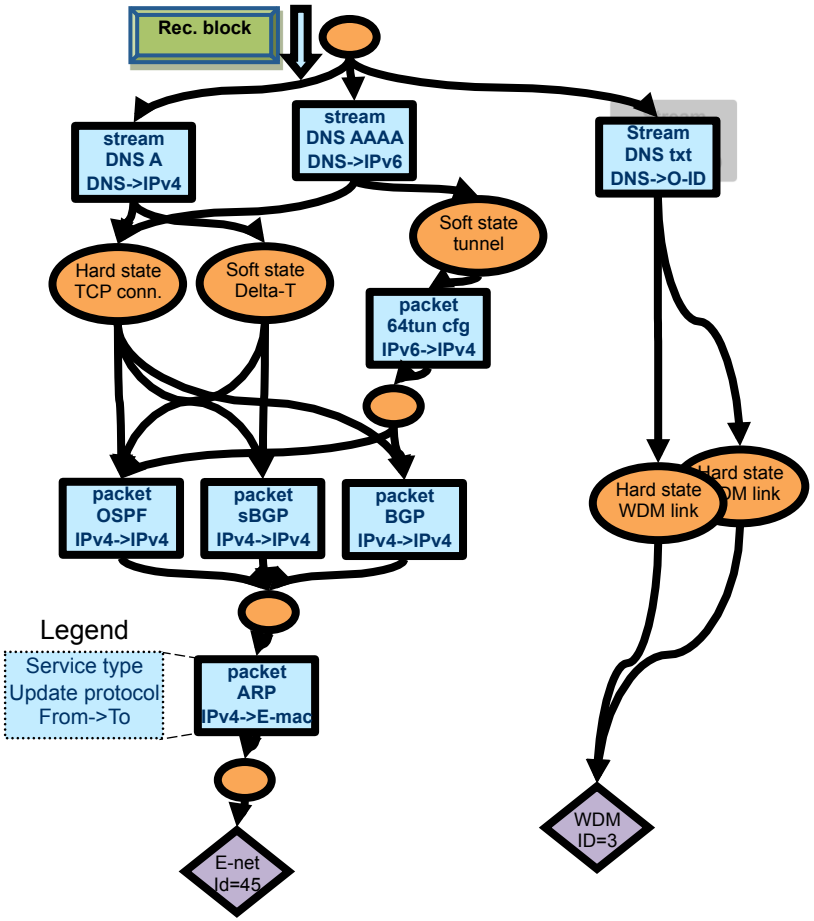
- The DAG describes motion through network layers
- Not motion between network nodes
- Each node on a path has its own DAG of this type
  - Simple or complex

The DAG

- Components
  - Nodes
  - Arcs
- Rules (constraints)

10,000 ft view

- Alternating nodes
  - FSM + state
  - Tables
- Directed arcs
  - With rules governing what they connect



Components

- Tables
- FSMs and their state
- Directed links

Tables

- Name translation
  - Bridge identities between layers
  - …which indicates choices of layers (or not)
  - …which indicates network paths

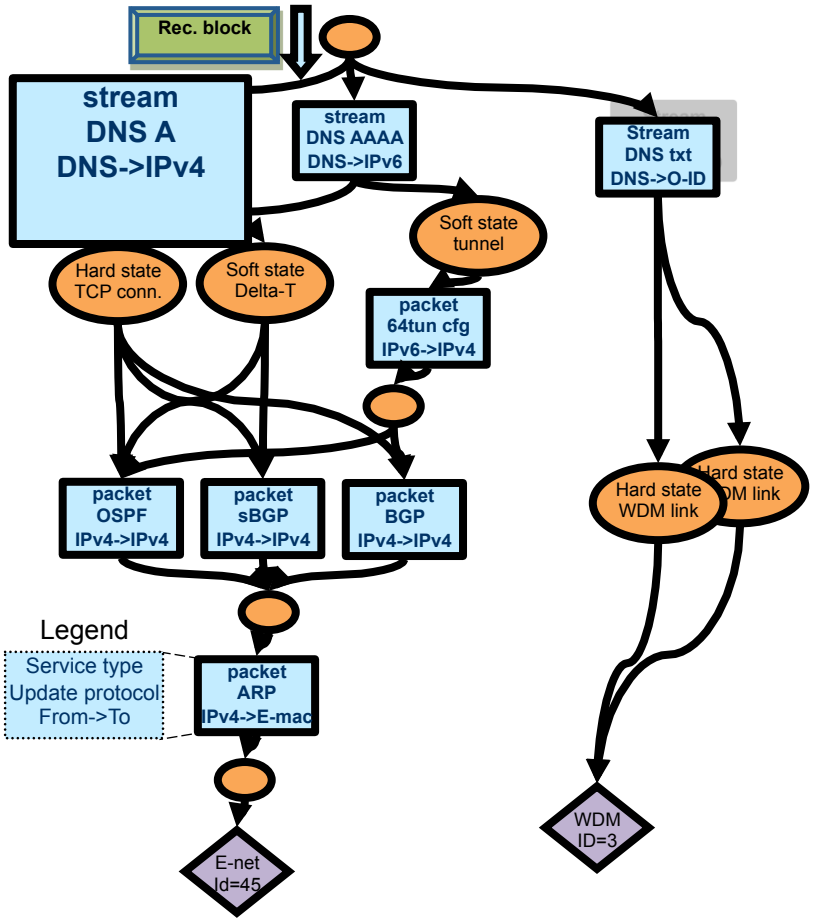
For example,

This table translates the DNS name to an IPv4 name

We could have chosen tables that translate the DNS name to an IPv6 or OID name

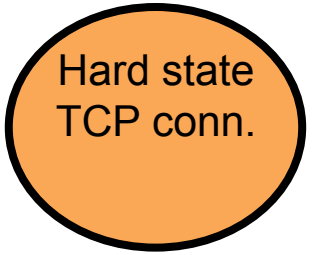
Implying different layer choices

And different paths through the network



FSMs and their state

- Represents the protocol
  - State
    - Representing a running FSM
  - Waiting for
    - Tape-in (from FSM above, i.e., “upper layer”)
    - Messages (from FSM below, i.e., “lower layer”)
    - Timer events
  - Emitting
    - Tape-out (to FSM above, i.e., “upper layer”)
    - Messages (to FSM below, i.e., “lower layer”)
  - Rules
    - Governing the relation of the above



For example,

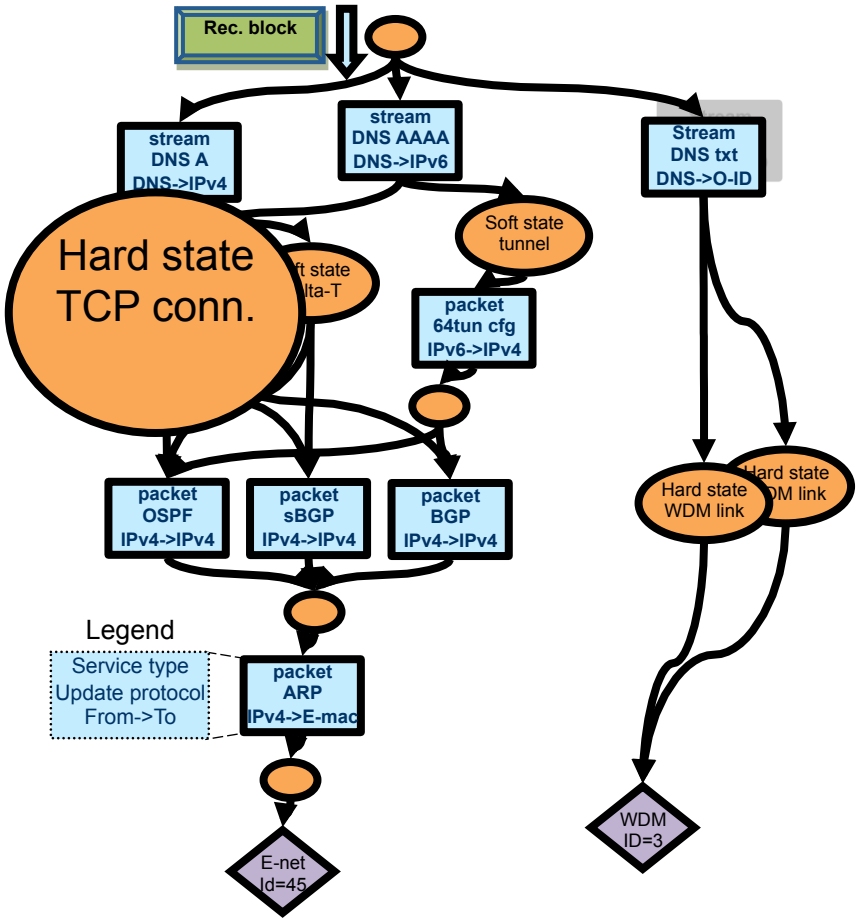
A TCP finite state machine

The layer above provided an IPv4 address (name) and a DNS name (msg)

It will emit messages to the lower layer

The lower layer will eventually provide a response from the DNS server

Which the TCP layer will emit to the upper DNS layer



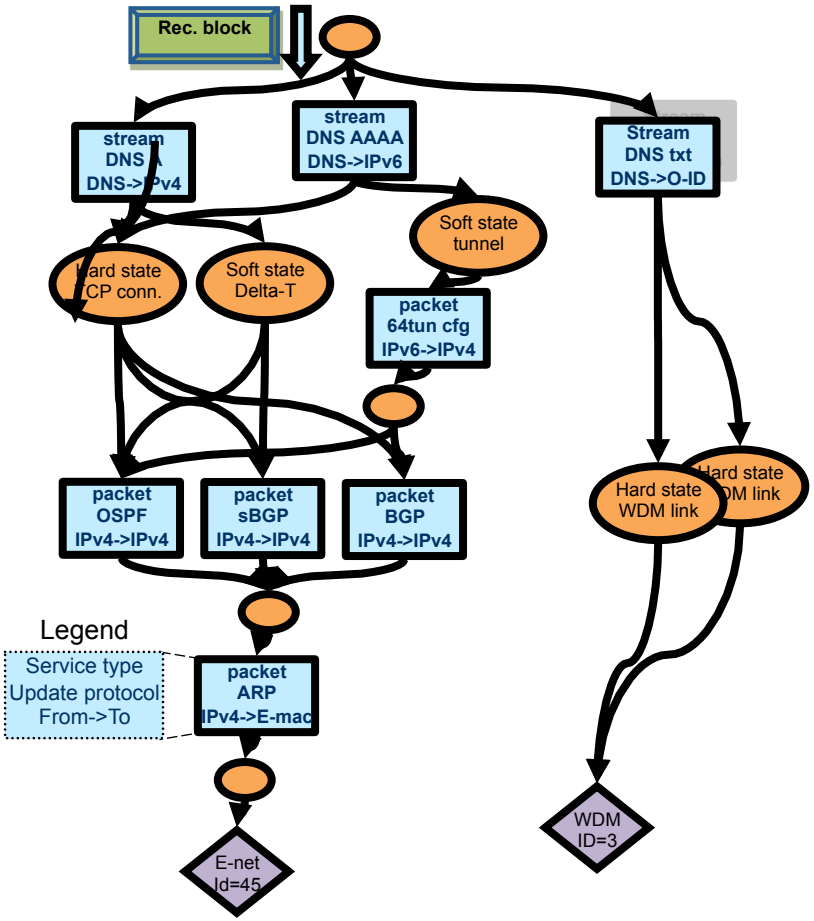
Directed links

- Describe relationships between
  - Running FSMs (protocols in progress)
    - Including "users" (information source/sink)
    - Including links (physical information conduits)
  - Translation tables

For example,

Describes relationship between DNS ➔IPv4 table and TCP FSM

That table outputs serve as inputs to TCP FSM



Rules (graph constraints)

- Structure
- Nodes
- Links
- Paths

Node rules

- Table rules
  - *Root* tables
  - *Leaf* tables
  - Other tables
- FSM + state rules

Root table rules

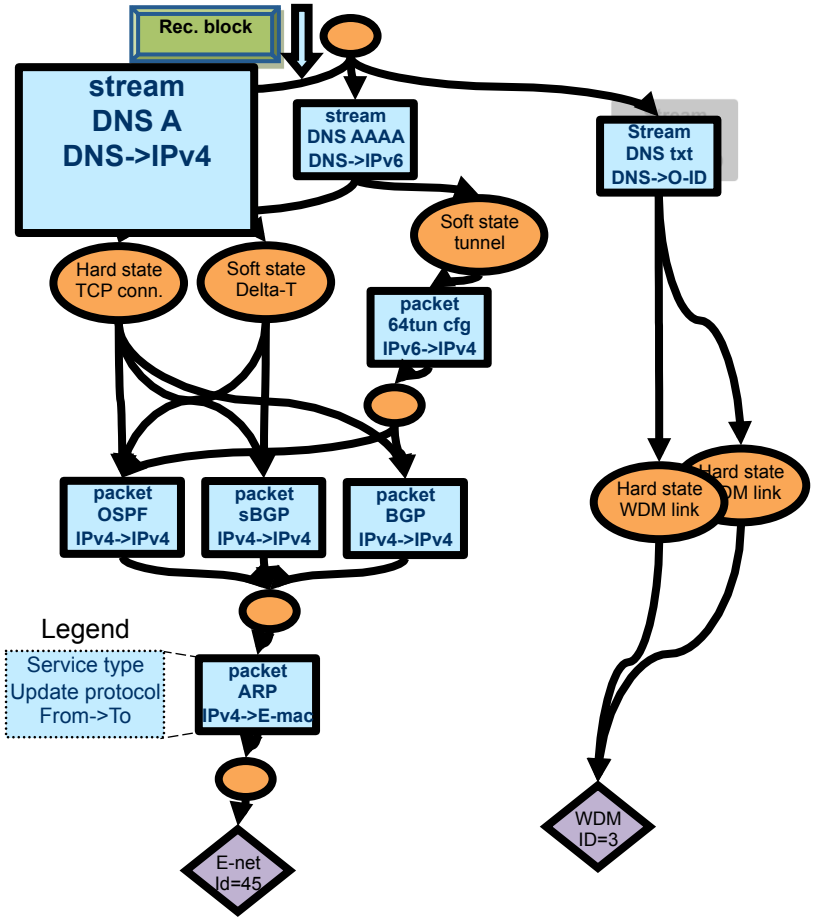
- Table just below the "user FSM"
  - User FSM represents input/output to the communication system
  - User FSM isn't really part of the system
  - "User" is not necessarily a human user
- Translates user-provided names to the first protocol name
  - User-provided names are local to the OS

For example,

The user or application provides a DNS name

This table translates the DNS name to an IPv4 name

For the use of the next protocol down (TCP, here)



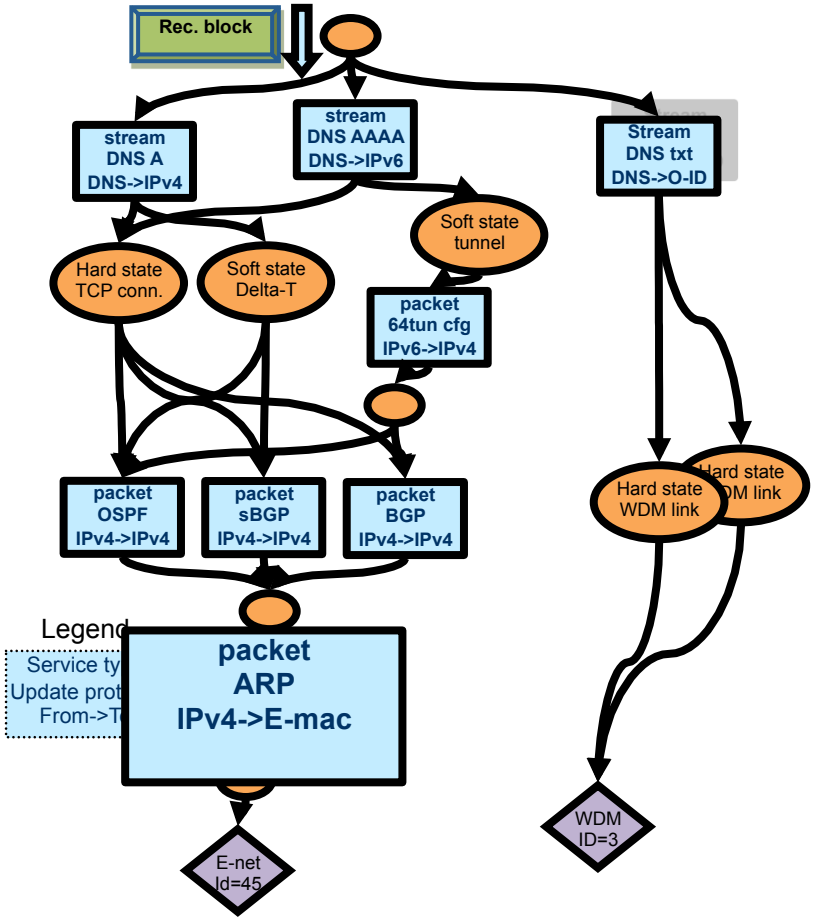
Leaf table rules

- Table just above the "link FSM"
  - Link FSM represents input/output to the physical link
  - Link FSM is the only part that's "real" (all else is emulated)
- Translates
  - Protocol names to physical encodings
  - Computation to communication

For example,

This table translates the IPv4 address to an Ethernet MAC address

Which the leaf protocol can use to physically send a message across an Ethernet channel



Intermediate table rules

- All other tables except root and leaf
  - Represents message in/out to FSM above
  - Represents tape-in/-out to FSM below
- Translates
  - Names in FSM above to names in FSM below

FSM + state rules

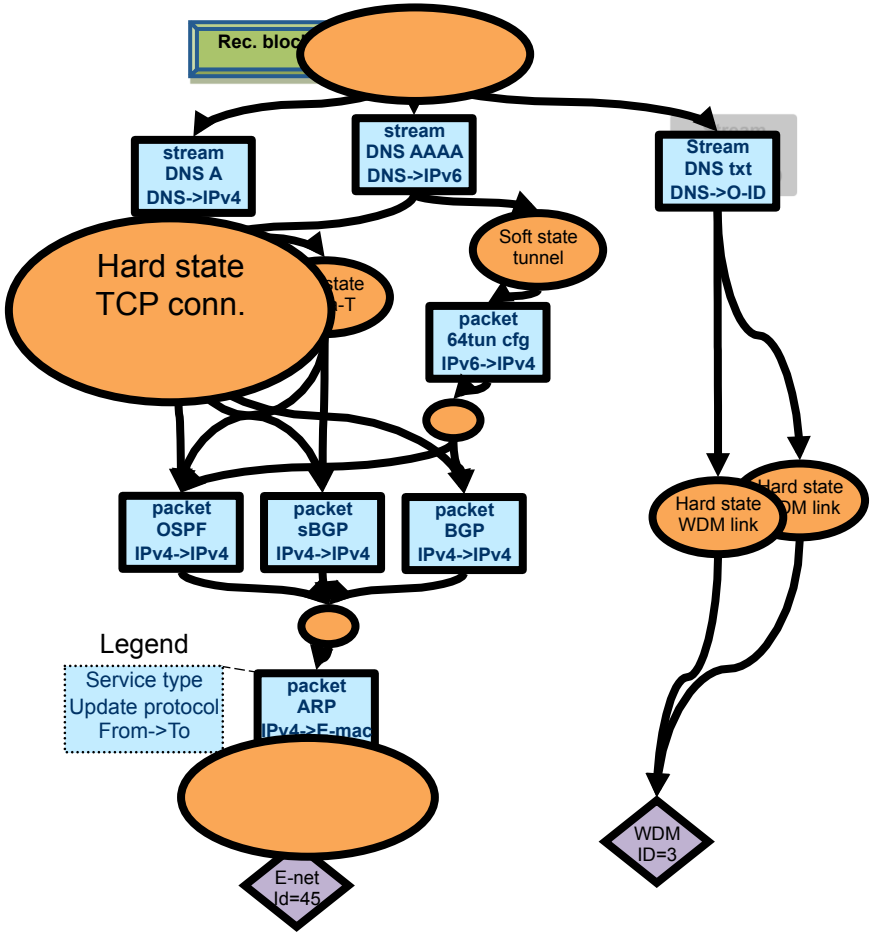
- FSM represents the protocol
  - Operates in a single namespace
- Real FSMs (part of the system)
  - Matches names in table above AND table below
- Virtual user FSM (top/root)
  - *Represents* system in/out
- Virtual link FSM (bottom/leaf)
  - *Represents* a physical link

For example,

A virtual user FSM

A real FSM

A virtual link FSM



General node rules

- Roots
  - FSMs that represent user programs
  - Link *to* root tables
- Leaves
  - FSMs that represent physical links
  - Linked *from* leaf tables
- FSMs
  - Operate in a single name space
  - Refer to tables “below”

Paths

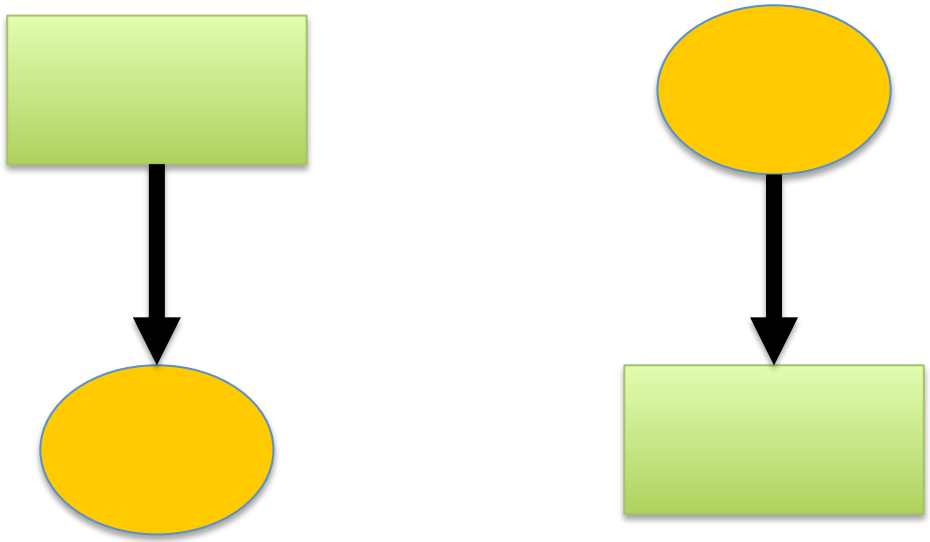
- Meaning
- Rules (constraints)

Linking nodes

- Orientation
  - Directed
  - Acyclic
- Head/tail
  - Connects different types of graph nodes:
    - If head is a table, tail is a FSM
    - If head is an FSM, tail is a table

Head/tail rule

- Tables and FSMs "alternate"
  - In our pictures, looks like this:



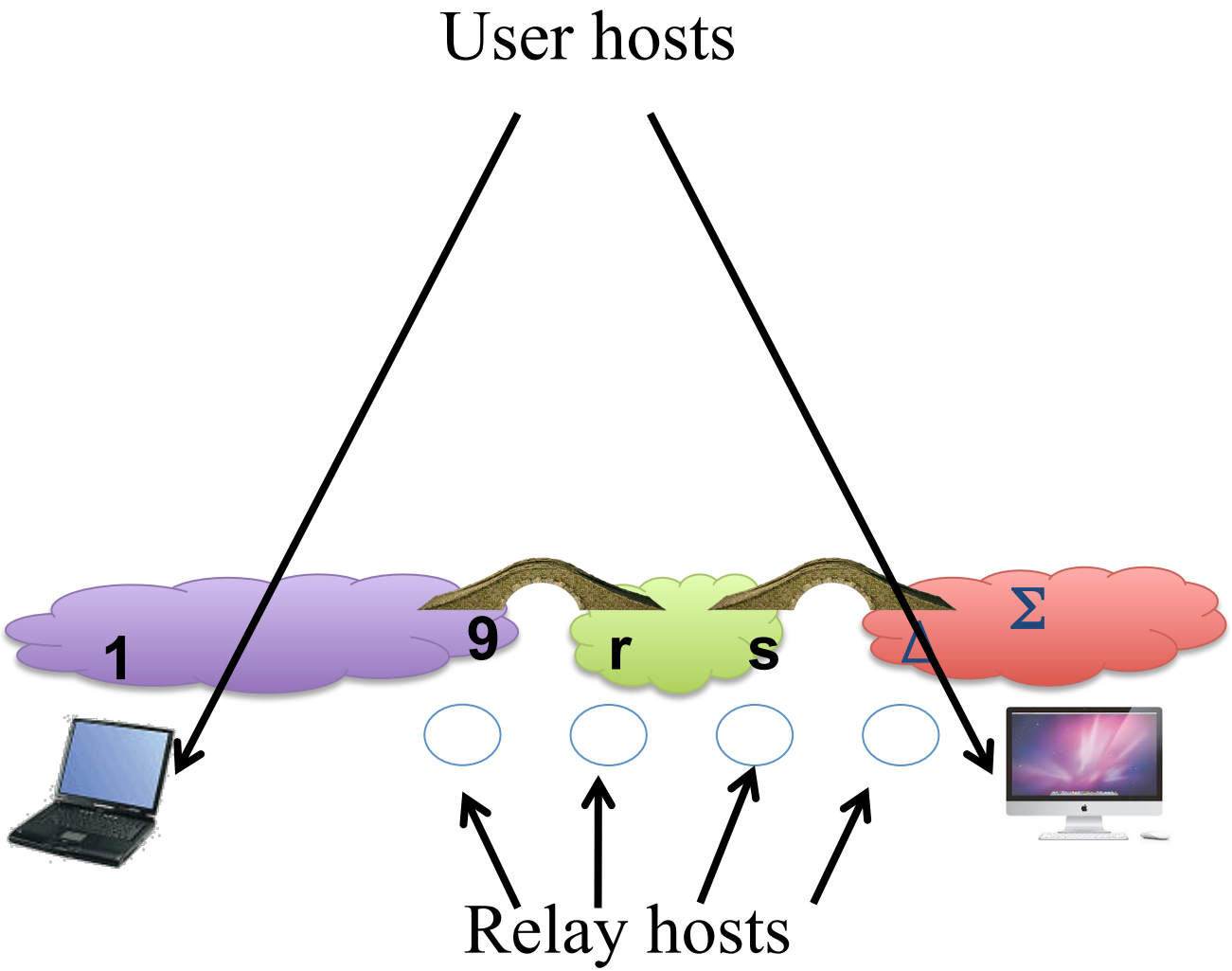
Path meaning

- Creates communication from networking
  - Composed of layered capabilities
- Describes the "stack" of layers
  - All communication is pairwise
  - Nested composition = linear path (the "stack")
  - Each message traverses a *single* path

Path rules

- View of a user host
- View of an intermediate (relay) host
- User to user view

The different host types



Path in a user host

- Source (push)
  - Enter at top
  - Exit at bottom
- Destination (pop)
  - Enter at bottom
  - Exit at top

Implications of host path rules

- Tables must tie user to a physical link
  - Must start with a name a user knows
  - Must follow a continuous chain of tables
  - Must end with a physical link

The maps tell you the DAG structure

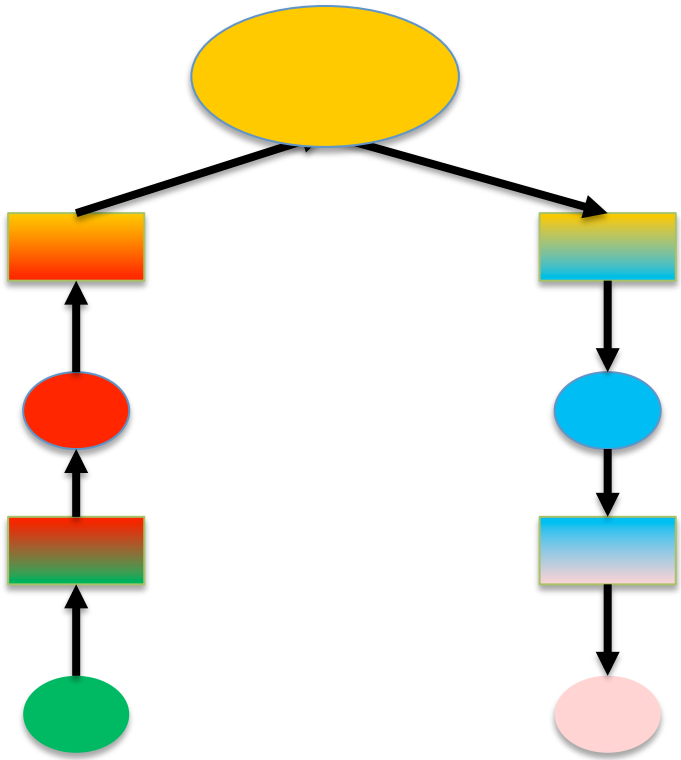
(real or implied)

DAGs and links

- Links are the physical network connections
- DAGs exist at the nodes at each end of a link
- We can hook together DAGs of different nodes together
- Describing the overall network path through layers on multiple nodes

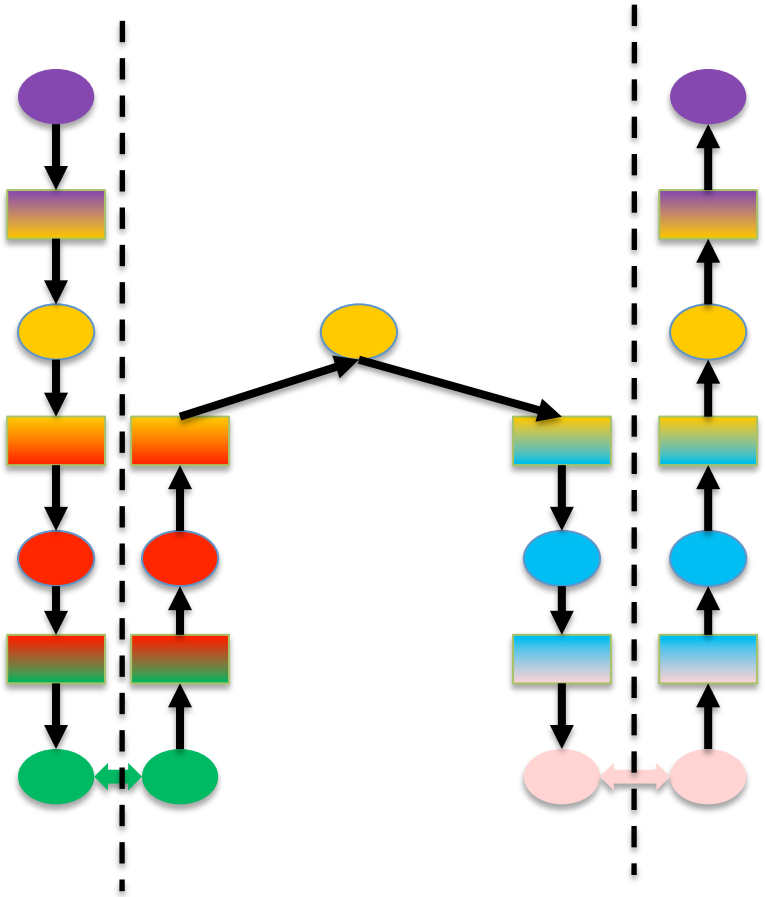
Path in a relay host

- Messages
  - Enter at a leaf (pop)
  - Exit at a leaf (push)
  - Share a common FSM
- No user input/output



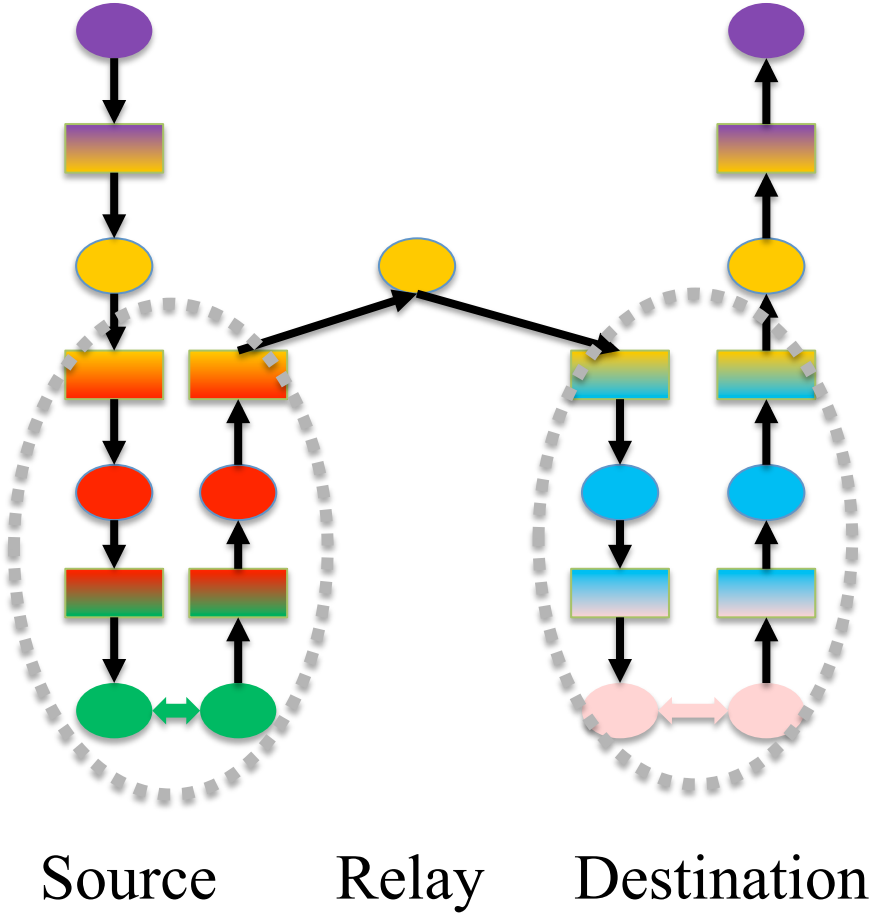
Path matching

- Links match tails



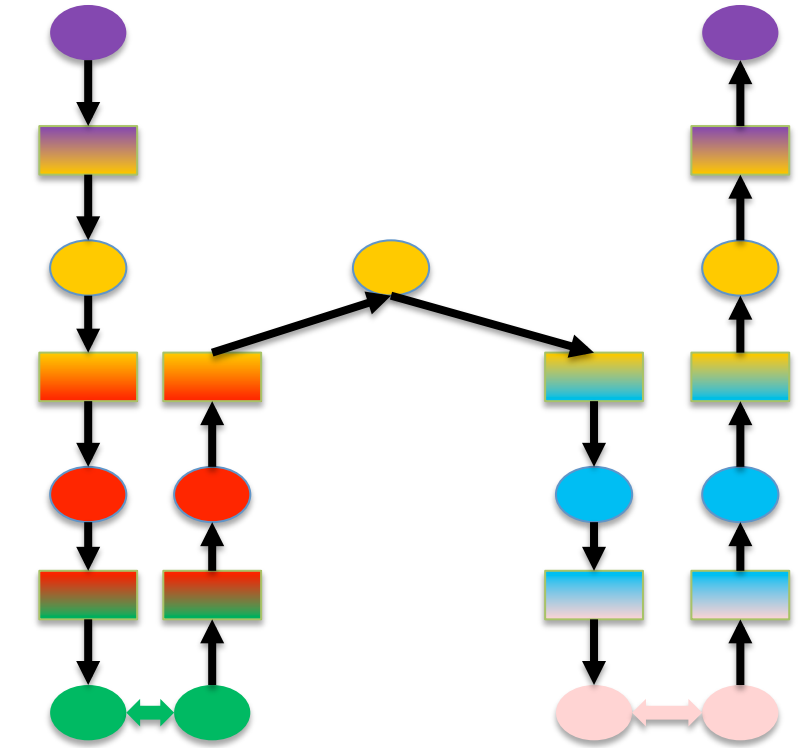
Path matching

- Links match tails
  - Links must match
  - Tails must match



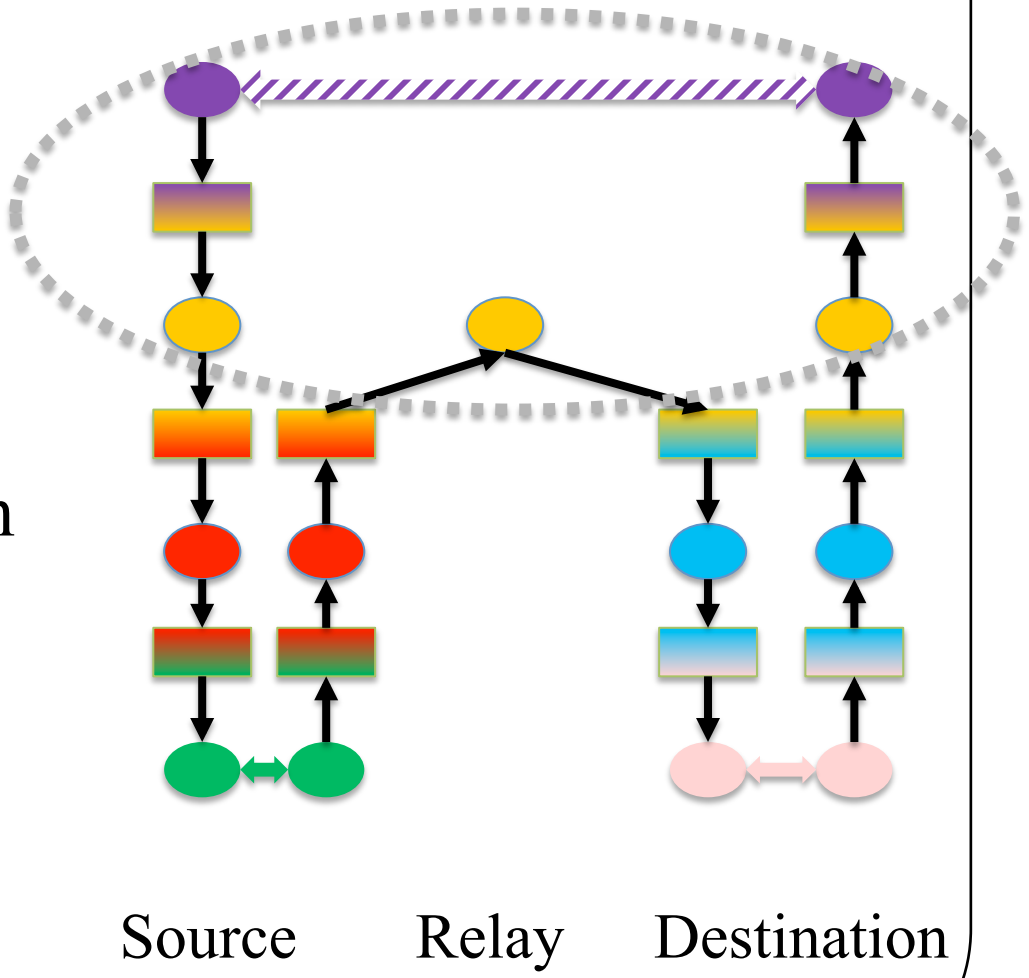
Path matching

- Links match tails
  - Links must match
  - Tails must match
- Ends *and* hops match heads



Path matching

- Links match tails
  - Links must match
  - Tails must match
- Ends *and* hops match heads
  - Emulate end-to-end
  - By relaying through shared node



Pushing and popping

- Each FSM
  - On the way down, adds info needed …
  - For matching FSM on the way up
- Push/pop
  - Stacked layers make messages and FSMs work like one large, distributed FSM

A deeper look…

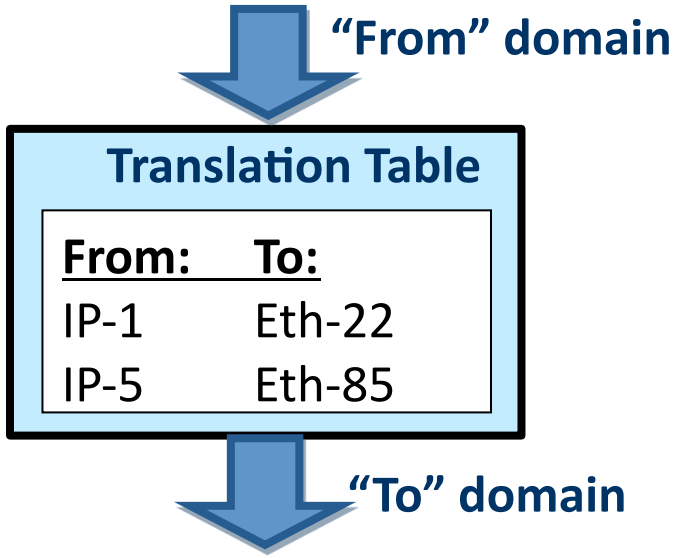
- Tables
- FSM + state

Tables

- Name translation
- Relay support

Name translation

- Map FROM domain into TO domain
  - "Domain" is another term for "namespace"
  - Map can have:
    - Aliases (N:1)
    - Proxies (1:N)
    - Additional information



Relaying support

- Governs when can you relay
- Provides additional information
- Table origins

When can you relay?

- When a table entry exists
  - When it's already there (cached, "push")
  - When you can fill it in (i.e., "pull")

Does an entry ensure connectivity?

- Nope!
- DAG inside one host *should* match, though
  - E.g., remove "thread" of entries "above" when the "bottom" has no entry

Other table information

- Resolution context
  - Cost, weight, etc. to differentiate proxies
- Table maintenance
  - Expiration time
  - Origin (for refresh, duplicate detection, etc.)

Table origins

- Manual
- External

Manual table configuration

- Preconfigured
  - Config files, boot files, etc.
- Manual manipulation
  - User commands

External table configuration

- Host configuration
  - DHCP
- Dynamic update
  - Routing protocols (covered next week)

Directed links

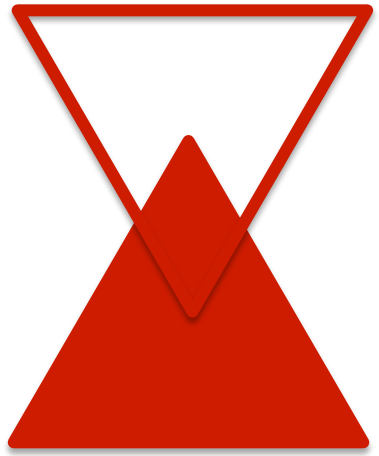
- As part of the table
- Defining alternates
- Defining protocol reuse and sharing

Links in the table

- What needs to be there?
  - Name you have
  - Name you want
  - A way to get to the next FSM
    - Via identifier of the namespaces
    - Via separate identifier (OS pointer, e.g.)
    - Or implied
      - In the picture so far, a table is defined as having one TO namespace, so the "next FSM" is implied

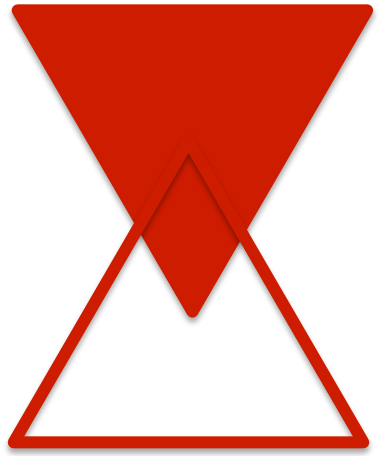
Alternates

- Multiple entries for a name to resolve
  - I.e., proxy
  - Weights or costs
    - Allows us to prioritize selection
  - Indicators of usage
    - Use only one
    - Use one until one works
    - Use multiple at the same time
      - Forces “parallelism” for fault tolerance, e.g.
  - I.e., the bottom half of the hourglass



Reuse

- Many FSMs point to a single table
  - All share the same FROM namespace
  - Allows the rest of the path down the DAG to be shared/reused by other protocols
  - I.e., the top half of the hourglass



FSMs and their state

- Soft vs. hard
- The base case as an FSM

Soft state

- Not critical to FSM operation
  - Recoverable
- Cached optimization
  - MUST be equivalent to “null start”
  - I.e., reaches the same result as an FSM that starts in the idle state

Hard state

- Critical to FSM operation
  - Cannot be recovered; results in error/failure
- Persists between messages
  - E.g., represents shared state with the other end of the channel
  - Provides context for FSM operation
  - Acts like a "paused" FSM

Soft vs. Hard

- Soft
  - Fault tolerant if lost
  - More efficient if it can recover/refresh
- Hard
  - Requires guarantees, backups, etc.
  - No communication until restored (if possible)

The base case as an FSM

- The base case FSM isn't really there
  - The channel is treated like an FSM

DAG walks

- Early binding
- Late binding

Early binding

- Setup path through DAG on first use
  - Record that path (e.g., socket data structure)
  - Reuse that path (forever)
  - Independent of FSM hard/soft state

Late binding

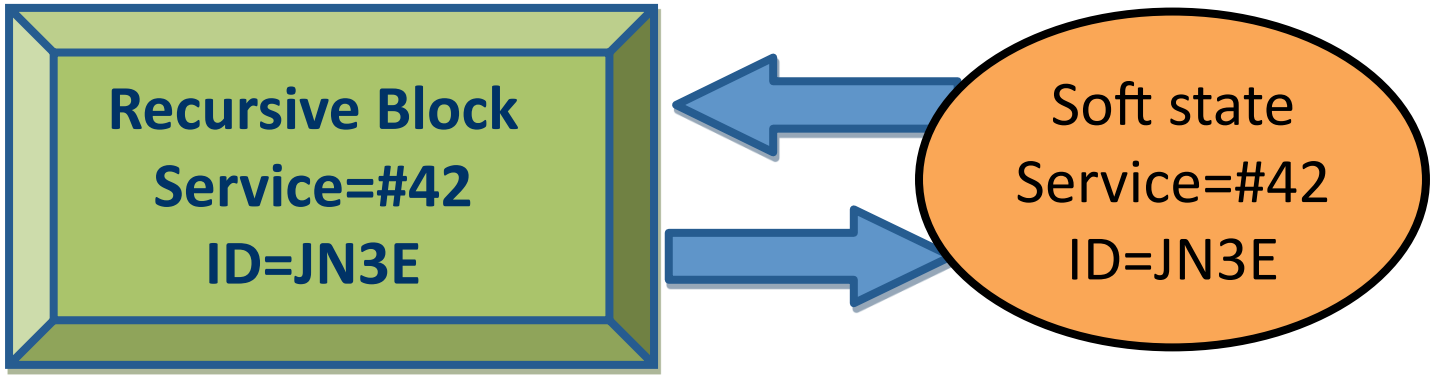
- Every message finds its own way
  - Ignore path of previous messages
  - Each message finds its own
  - Independent of FSM hard/soft state

Early vs. late

- Early (static)
  - Pros
    - Ensures stability
  - Cons
    - Cannot adapt (suboptimal, fragile)
- Late (dynamic)
  - Pros
    - Adapts while info exchange is in progress
  - Cons
    - Can be expensive (slow, costs resources)
    - Can be unpredictable

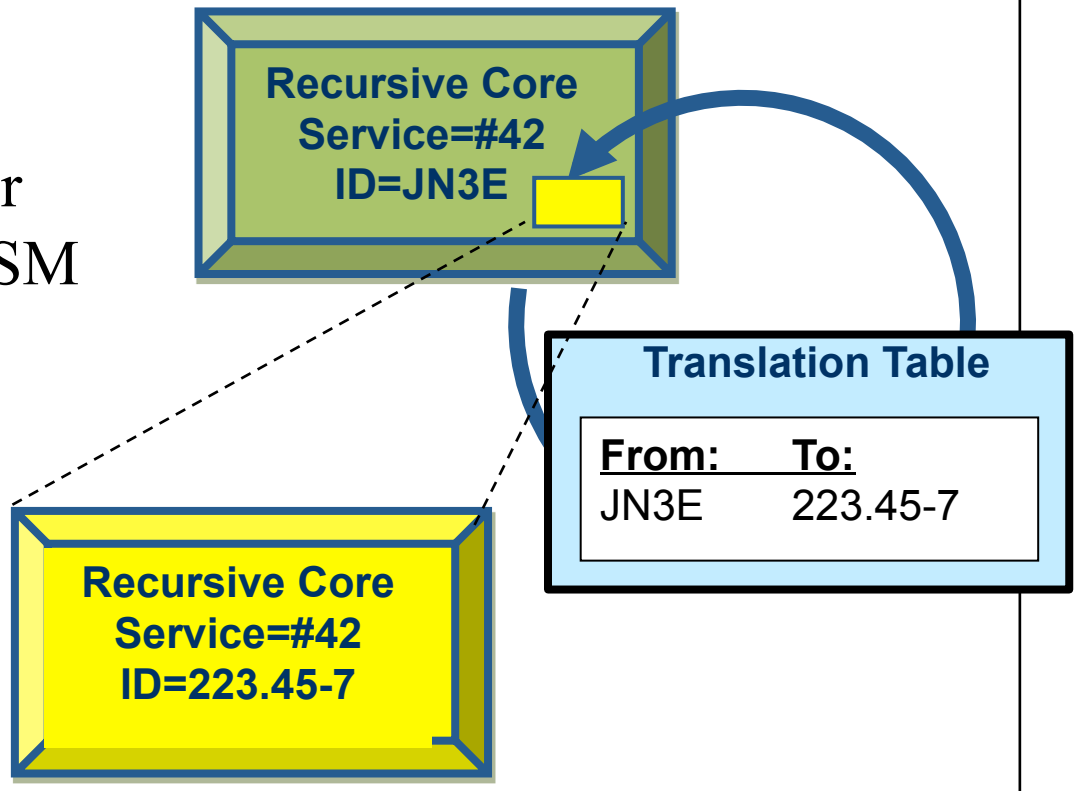
Processing

- The recursive block walks the tree
  - Using message as part of context for FSM



Graph traversal

- Each recursive step
  - Walks a layer down
  - Wraps needed state for retrieval at receiver FSM



Summary

- The DAG encodes:
  - The protocol stack
  - The network architecture
- The DAG has constraints
  - Esp. DAG portions must match
- The DAG is a concept
  - Implementations vary, grouping/partitions vary