MOSAIC: Unified Declarative Platform for Dynamic Overlay Composition

Yun Mao[1,2], Boon Thau Loo[1], Zachary Ives[1], and Jonathan M. Smith[1]

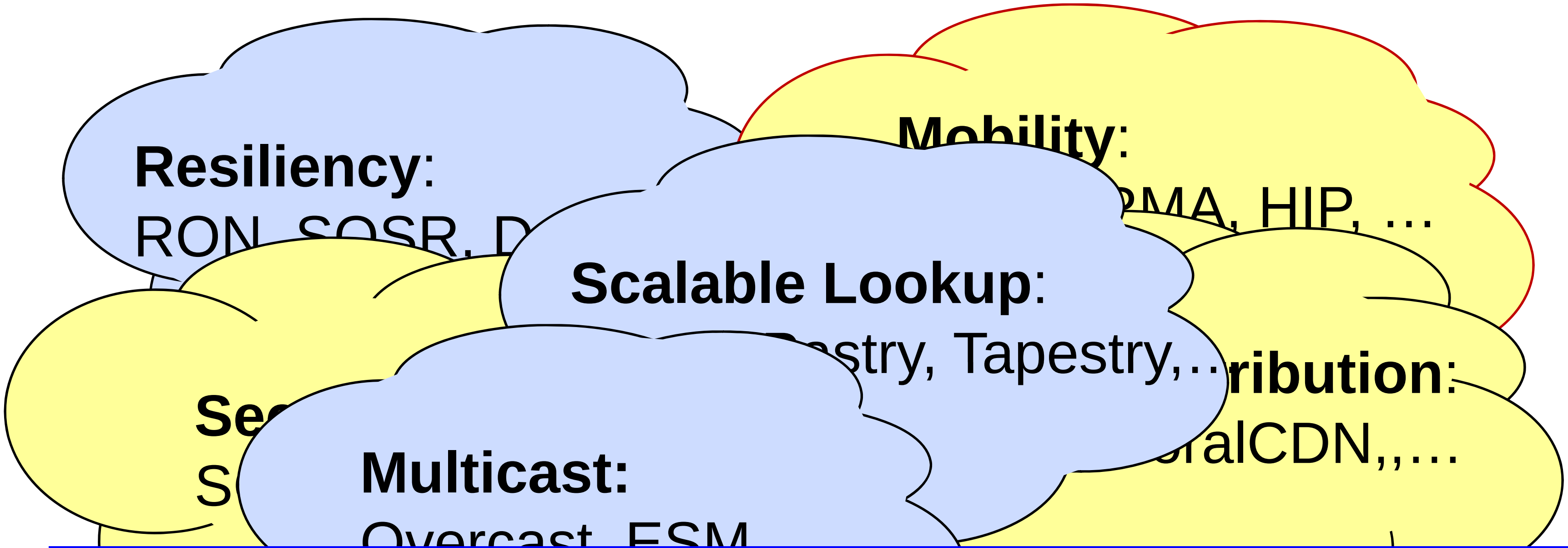
University of Pennsylvania, AT&T Research

Challenges to Today’s Internet

- New applications demand new capabilities
  - Mobility, quality of service, content-based routing, anycast, multicast, …
  - Many applications require more than one capability
- Challenges
  - Unwanted and harmful traffic
  - Complexity and fragility of inter-domain routing
- Hard to address in the current architecture
  - Changing the core is hard
  - Protocols are deeply coupled to their implementations

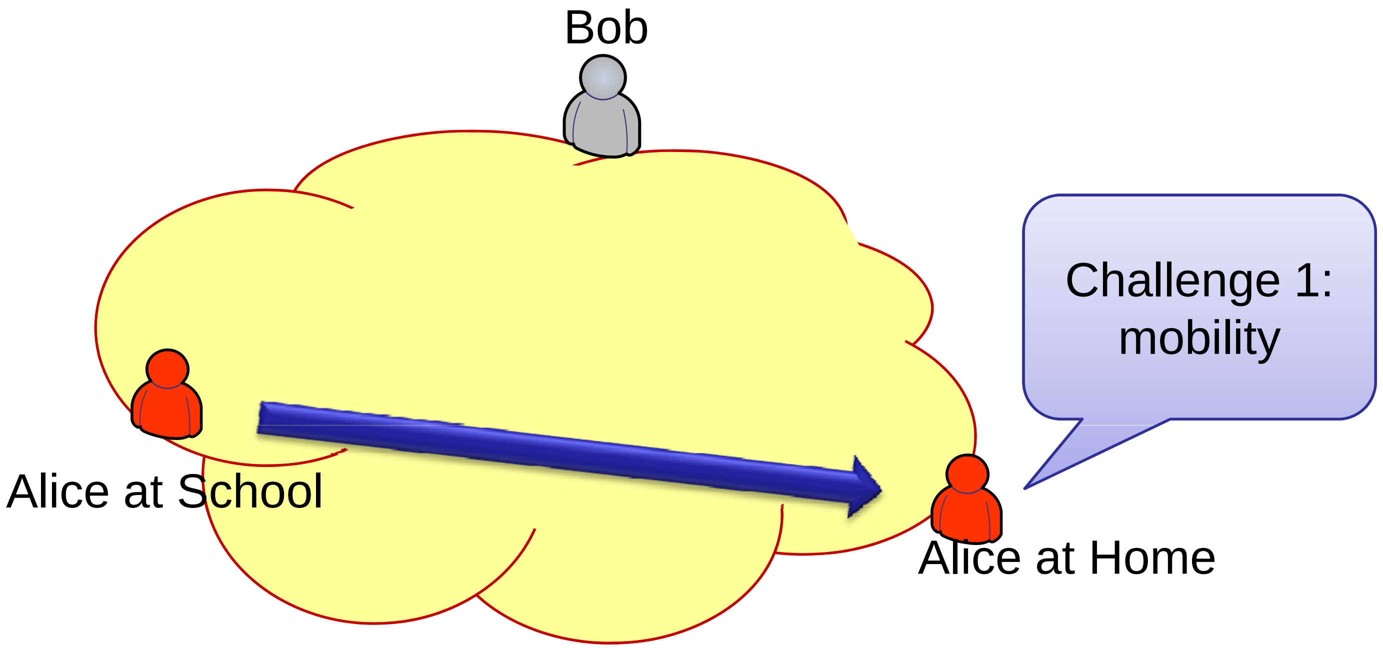
Overlay Networks

- Application-level networks that achieve new functionality without changing the infrastructure:

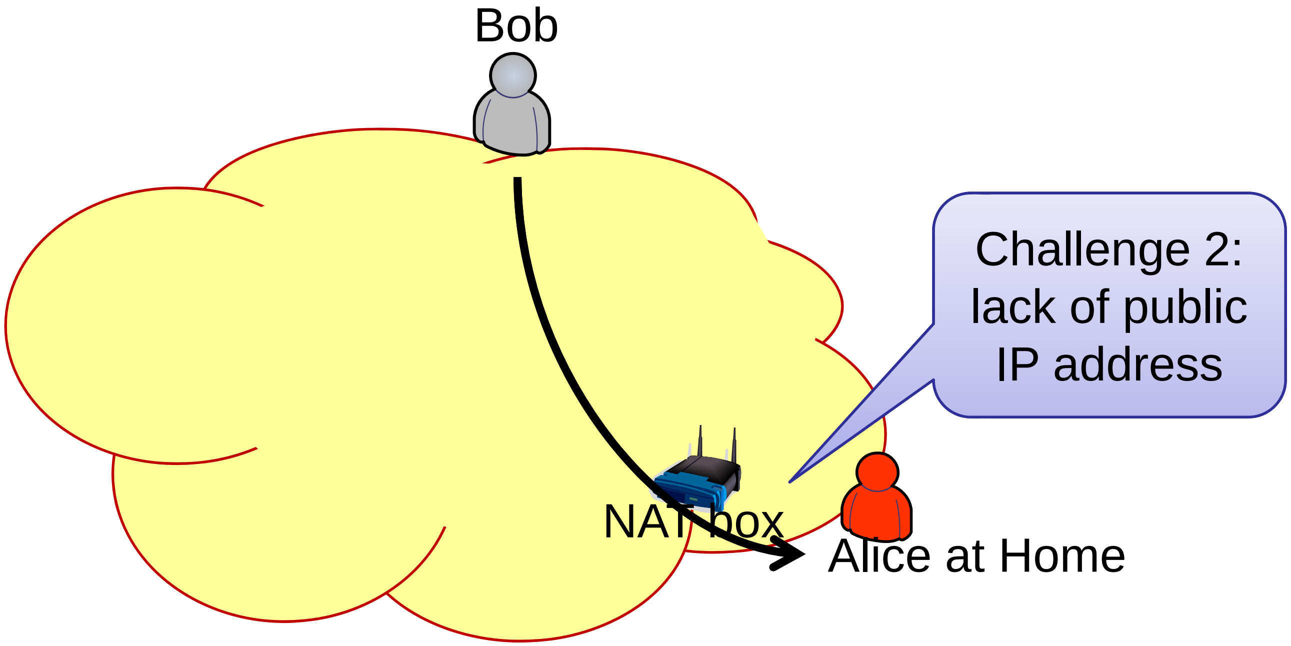


- Do not fully address challenges and evolutionary needs
- Limited to application-specific domains
- Not easy to “mix-and-match” to support new applications

Example 1: Alice&Bob

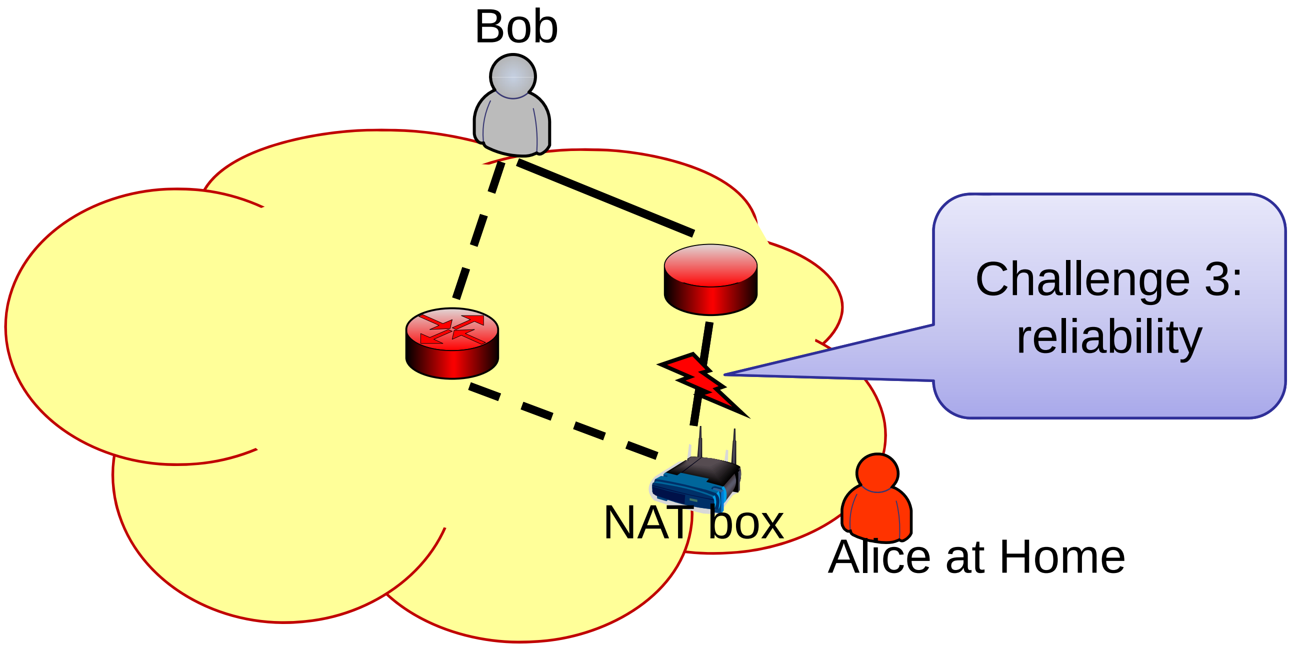


Example 1: Alice&Bob



Challenge 2: lack of public IP address

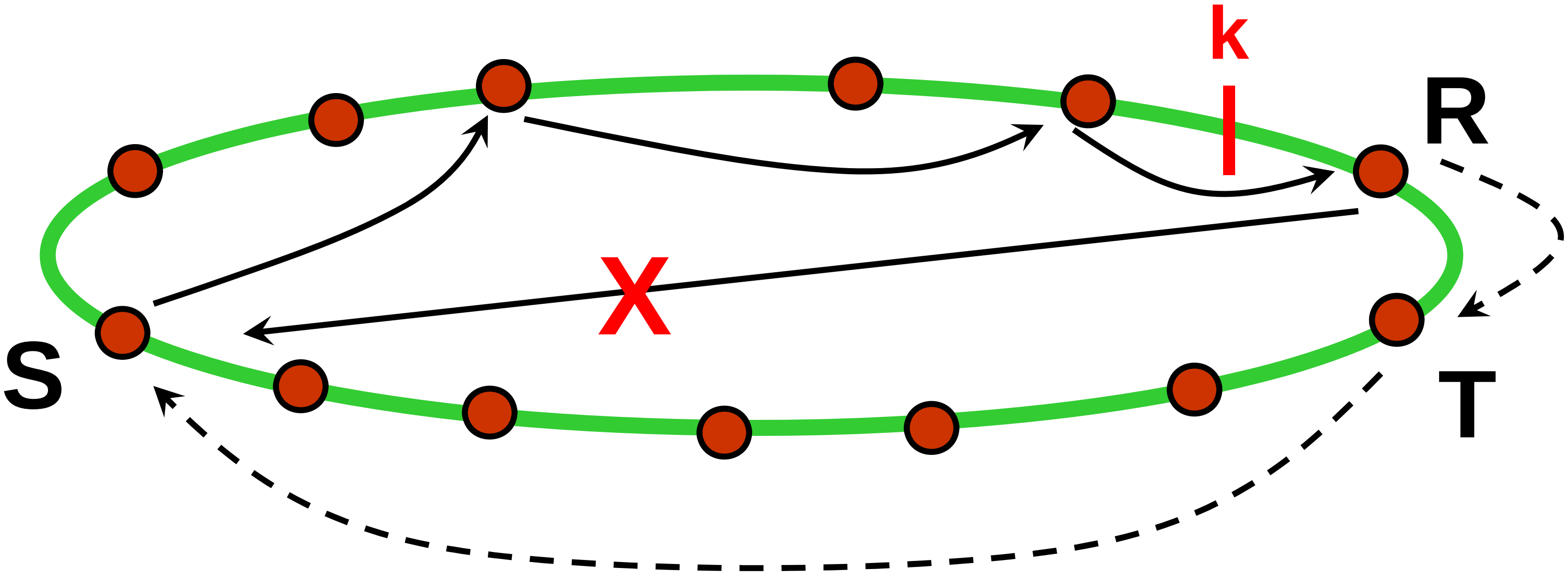
Example 1: Alice&Bob



Challenge 3: reliability

- Each individual challenge has solutions based on overlays, but none of them solve all the problems at once
- Change in environment (trust level, connectivity, etc) may invalidate chosen overlay

Example 2: Distributed Hash Table (DHT) with Network Failures



- Intermittent network failures result in broken return paths, and other issues
- Layer DHT over a resilient overlay can help!

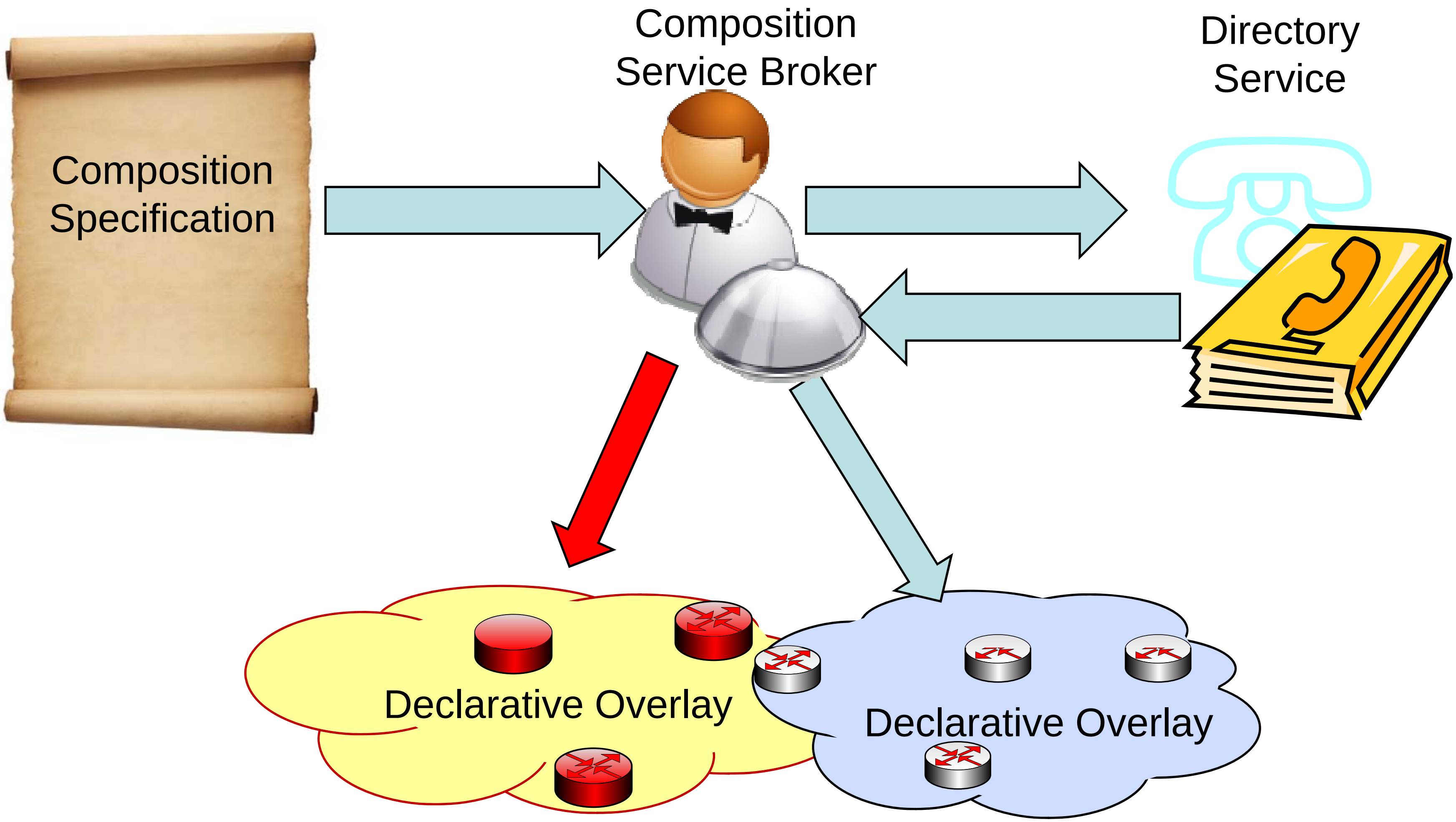
Goals of Overlay Composition

- Speed-up network evolution:
  - Novel application-specific networks built on multiple overlay compositions (bridging / layering)
  - Component reusability
  - Dynamic adaptation: modify components as requirements or environment changes.
- Support custom application needs. E.g.,
  - Alice&Bob example:
    - i3+RON+bridges = mobility + reliability + NAT traversal
  - DHT in failure-prone networks
    - Chord DHT over RON = better lookup performance
  - Secure mobility:
    - i3 over secure overlay = mobility + eavesdropping prevention
  - …

MOSAIC Approach

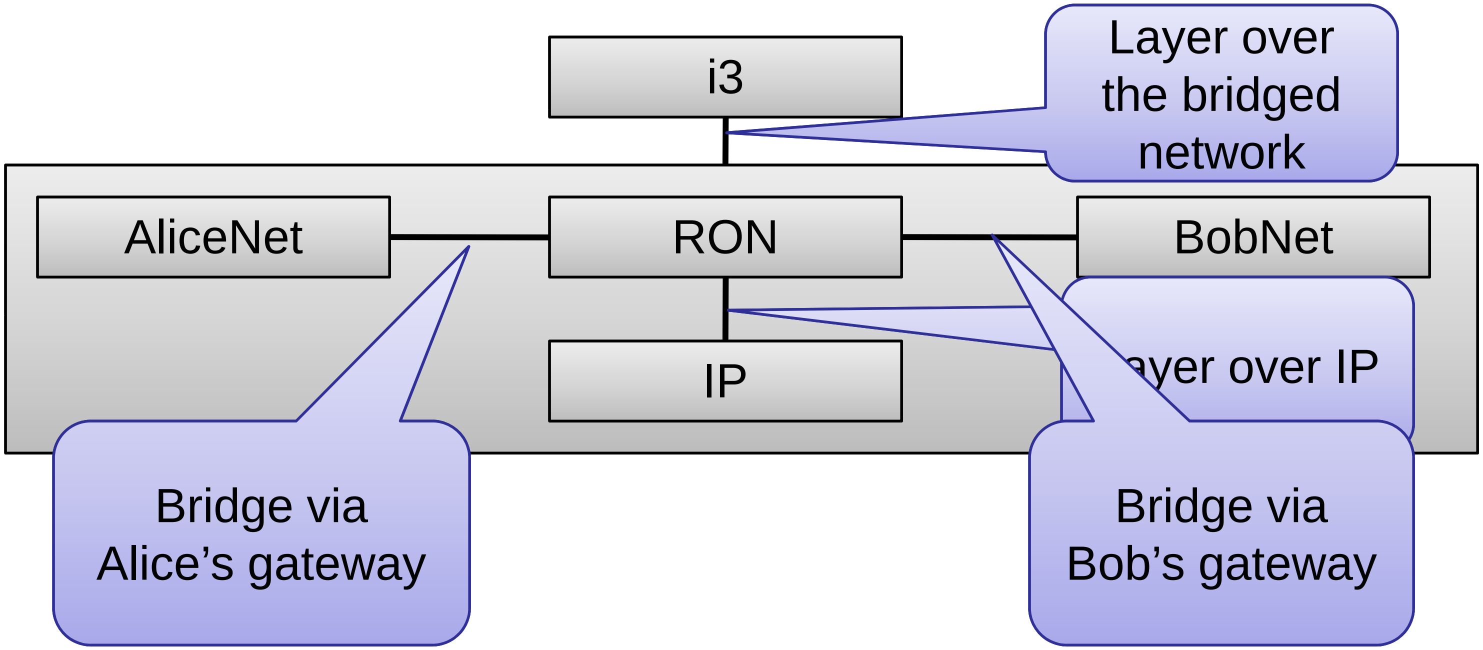
- Declarative framework for rapidly prototyping and composing overlay networks, and dynamically changing the compositions at runtime
- Leverages declarative networking [SIGCOMM ’05,SOSP ’05]*:*
  - Declarative specifications of networks using a distributed database query language
  - Distributed query engine executes specifications to implement network protocols
- Key advantages:
  - Compact and high-level representation of protocols
  - Orders of magnitude reduction in code size
  - Easy customization, sharing, and composability

MOSAIC Overview



Composition Specification

The *composition graph* of the Alice&Bob example:



i3 – Internet Indirection Infrastructure (for NAT and mobility)
RON – Resilient Overlay Network

Composition Specifications to Implementation

- Validation stage:
  - **Bridging:** two networks share common physical nodes
  - **Layering:** nodes of underlay are a super set of the overlay
- Compilation stage:
  - Declarative implementation of overlays
  - Generate the “glue code” as declarative networking rules
  - Ship rules to physical nodes for execution on a declarative networking engine (P2, http://p2.cs.berkeley.edu)

Background: Declarative Networking

Network Datalog: a distributed query language for networks

➔ R1: reachable(@S,D) :- link(@S,D)

R2: reachable(@S,D) :- link(@S,Z), reachable(@Z,D)

link(@a,b) – "there is a link from node *a* to node *b*"

reachable(@a,b) – "node *a* can reach node *b*"

"For all nodes S,D,

If there is a link from S to D, then S can reach D".

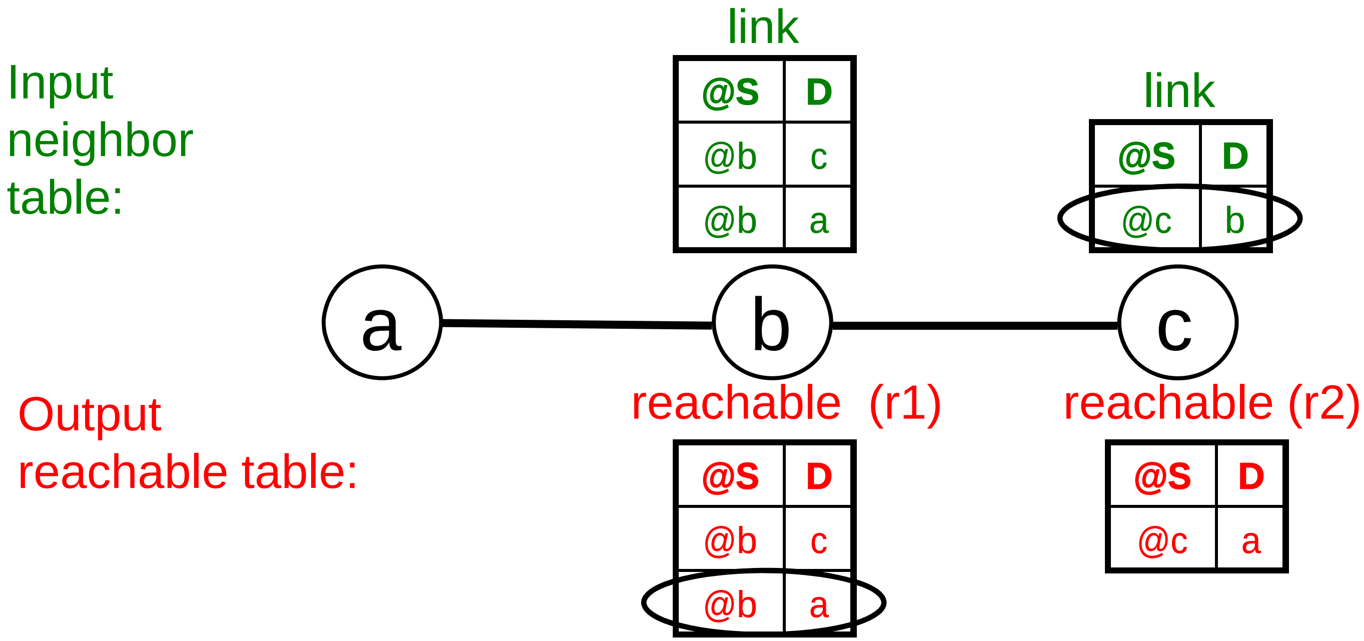
Background: Declarative Networking

Network Datalog: a distributed query language for networks

R1: reachable(@S,D) :- link(@S,D)

➔ R2: reachable(@S,D) :- link(@S,Z), reachable(@Z,D)

"For all nodes S,D and Z,
If there is a link from S to Z, AND Z can reach D, then S can reach D".



Input neighbor table:

link

@S	D
@b	c
@b	a

link

@S	D
@c	b

Output reachable table:

reachable (r1)

@S	D
@b	c
@b	a

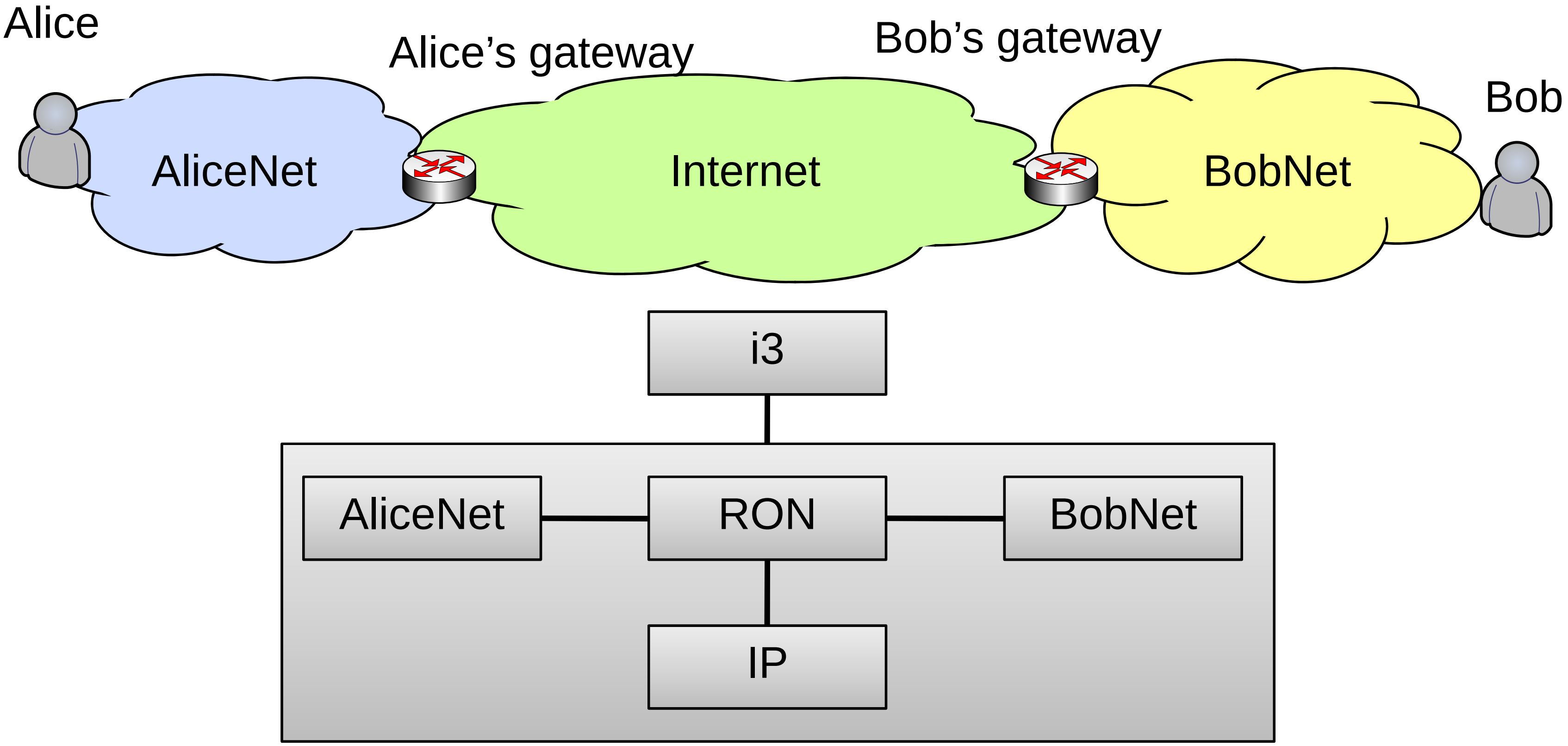
reachable (r2)

@S	D
@c	a

Large Library of Declarative Protocols

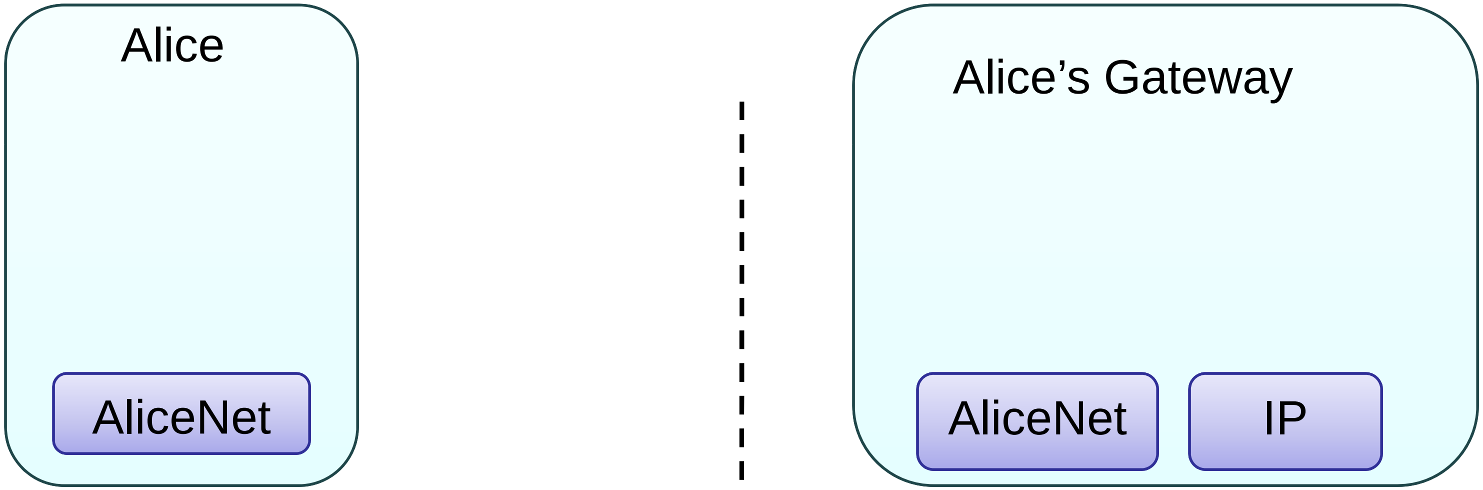
- Example implementations to date:
  - Routing protocols (DV, LS, DSR, AODV, OLSR, etc.)
  - Chord Distributed Hash Table
  - Resilient overlay network (RON)
  - Internet Indirection Infrastructure (i3)
  - Others: sensor networking protocols, replication, snapshot, fault tolerance protocols
- Language extensions:
  - Logical location specifiers (not just IP addresses)
  - Composible views (grouping rules together as components)
  - Legacy application support (via tunneling)

Composition Example: Alice & Bob



- Composition – 69 rules
  - Chord DHT – 35 rules, i3 – 16 rules, RON – 11 rules
  - Auto-generated composition "glue" – 7 rules (for layering and bridging)

Alice’s Initial State



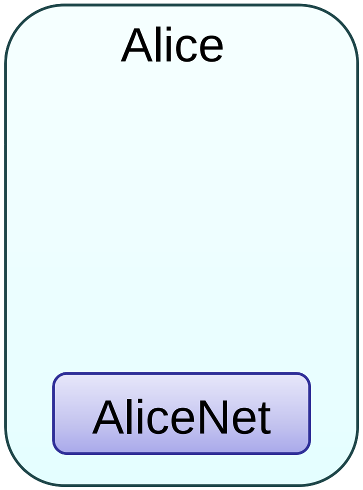
OverlayID	Overlay Address
alice_net	10.0.0.2

Table `netAddress`

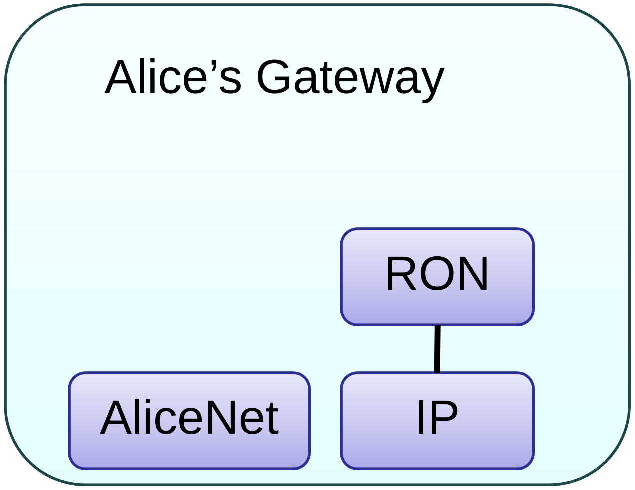
OverlayID	Overlay Address
alice_net	10.0.0.1
0	12.34.56.78

Table `netAddress`

Create RON



OverlayID	Overlay Address
alice_net	10.0.0.2

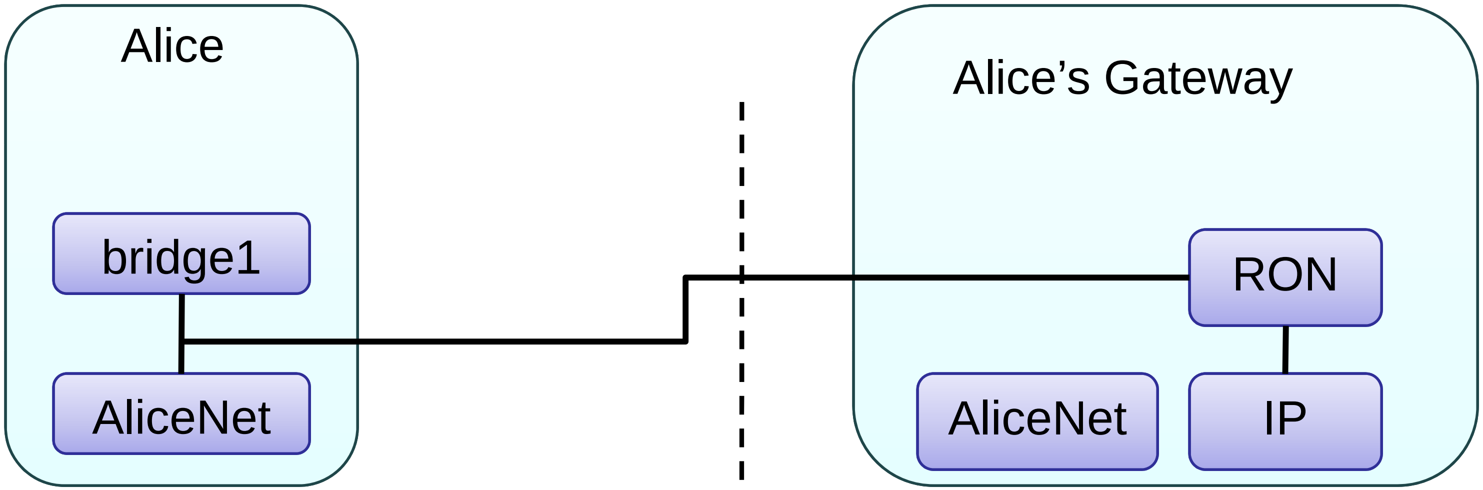


OverlayID	Overlay Address
alice_net	10.0.0.1
0	12.34.56.78
ron	ron::12.34.56.78

RON over IP:

```
c1 underlay(ron,A):-
       netAddress(0,A).
```

Create bridge1



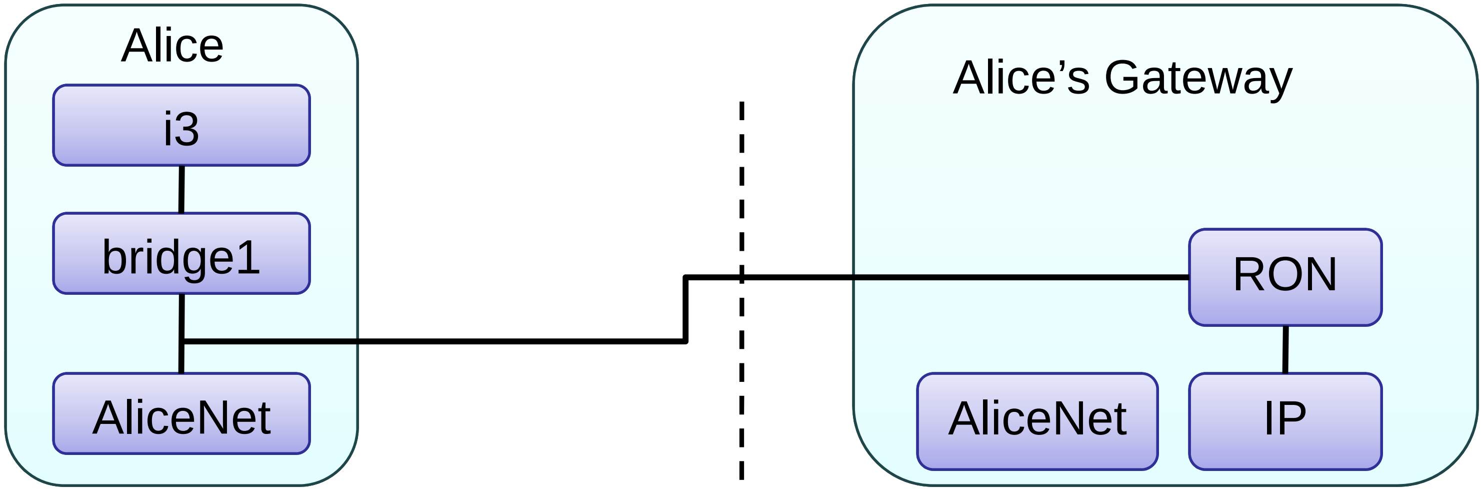
OverlayID	Overlay Address
alice_net	10.0.0.2
bridge1	sr::[ron::12.34.56.78, 10.0.0.2]

OverlayID	Overlay Address
alice_net	10.0.0.1
0	12.34.56.78
ron	ron::12.34.56.78

RON over IP:

```
c1 underlay(ron,A):-
        netAddress(0,A).
```

Create i3



OverlayID	Overlay Address
alice_net	10.0.0.2
bridge1	sr::[ron::12.34.56.78, 10.0.0.2]
i3	i3::alice_id

OverlayID	Overlay Address
alice_net	10.0.0.1
0	12.34.56.78
ron	ron::12.34.56.78

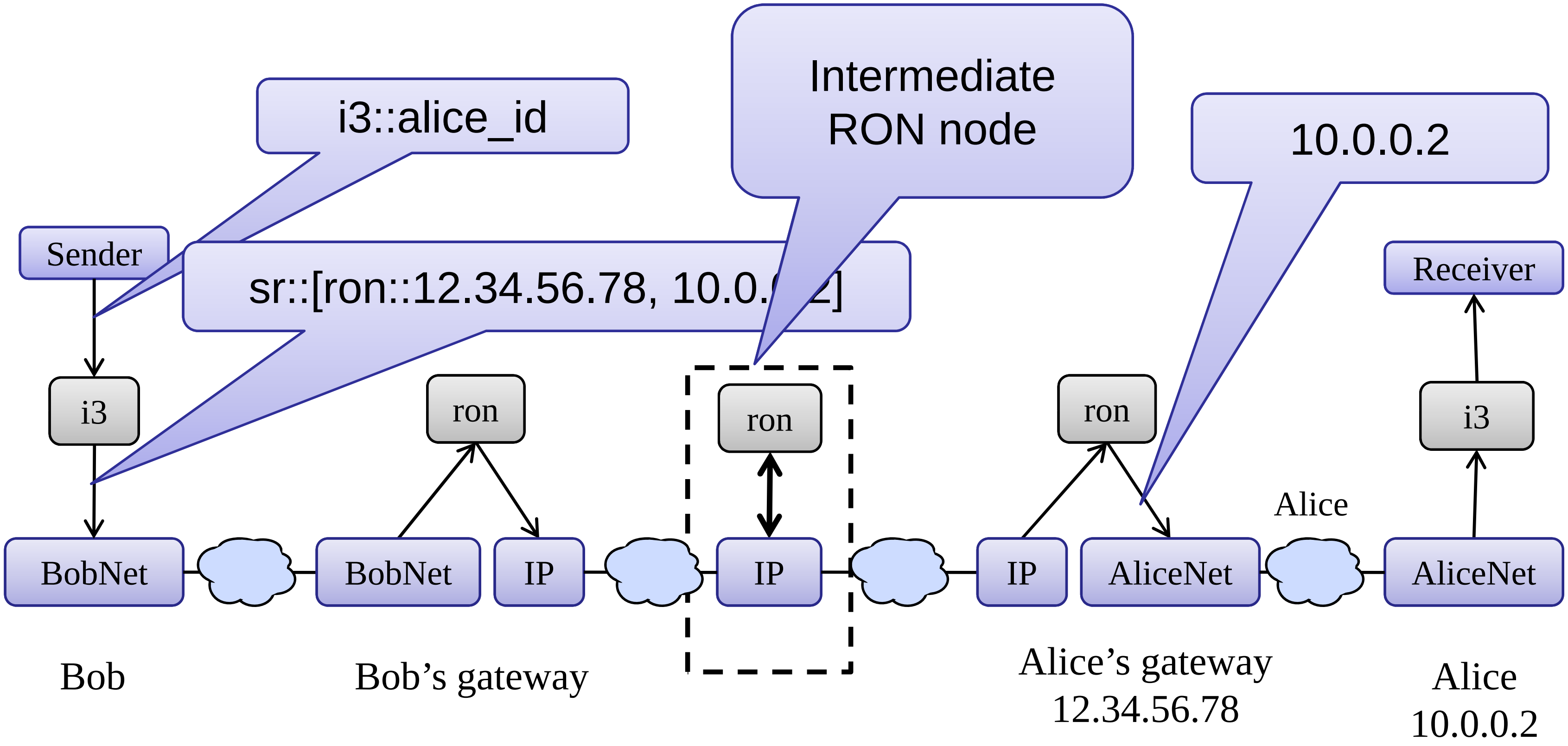
i3 over the bridged network:

```
c2 underlay(i3_oid, A):-
        netAddress(bridge1,A).
```

RON over IP:

```
c1 underlay(ron,A):-
        netAddress(0,A).
```

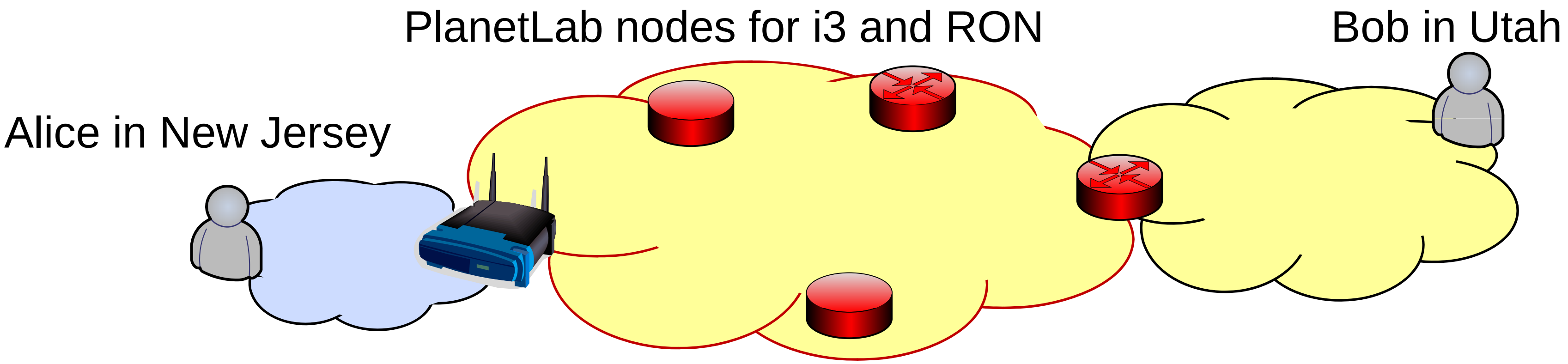
Composed Network

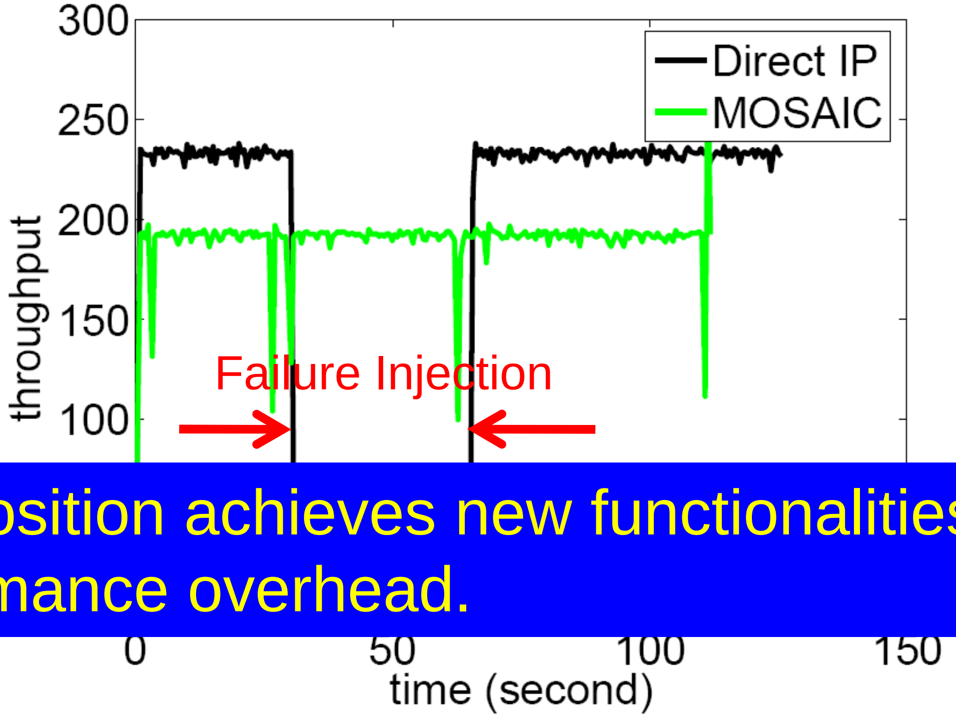


Implementation

- Compilation to distributed dataflows
  - Similar to Click modular router
  - Additional support for relational operators, encapsulation/de-encapsulation, and legacy application support via *tun* device
- Evaluation:
  - Performance benchmarks in a LAN
  - Wide-area composition evaluations on PlanetLab

Experiment 1: Mobility + NAT + Reliability

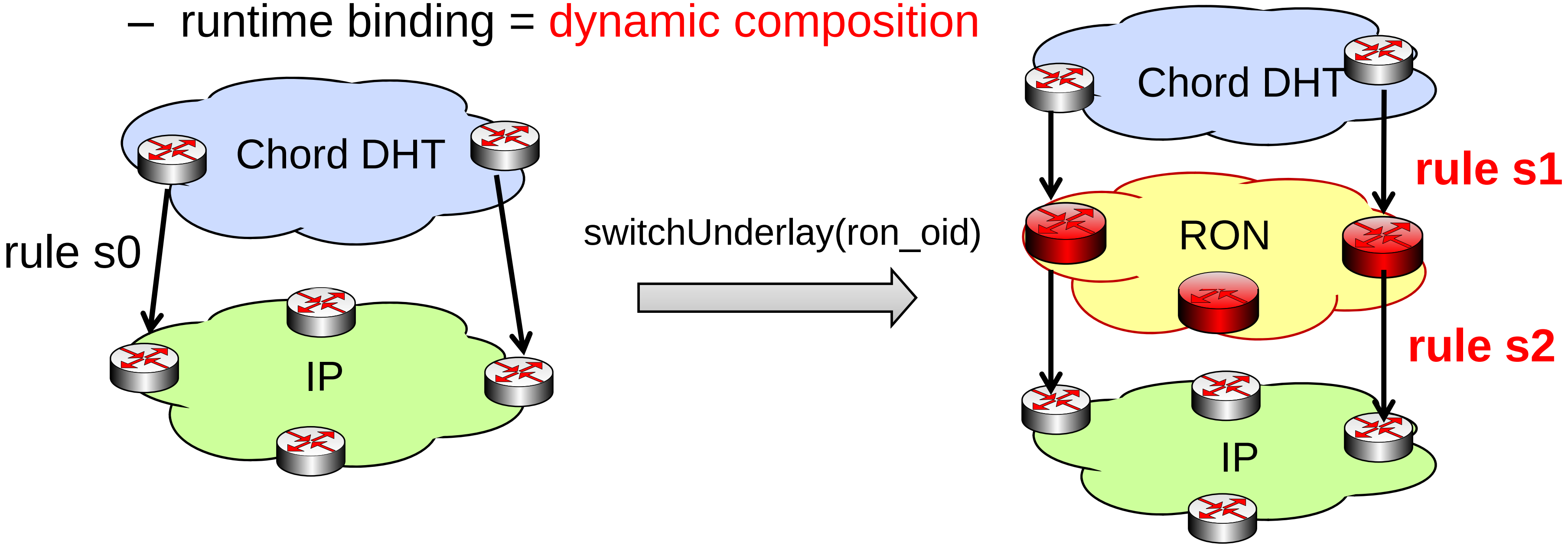




Composition achieves new functionalities at low performance overhead.

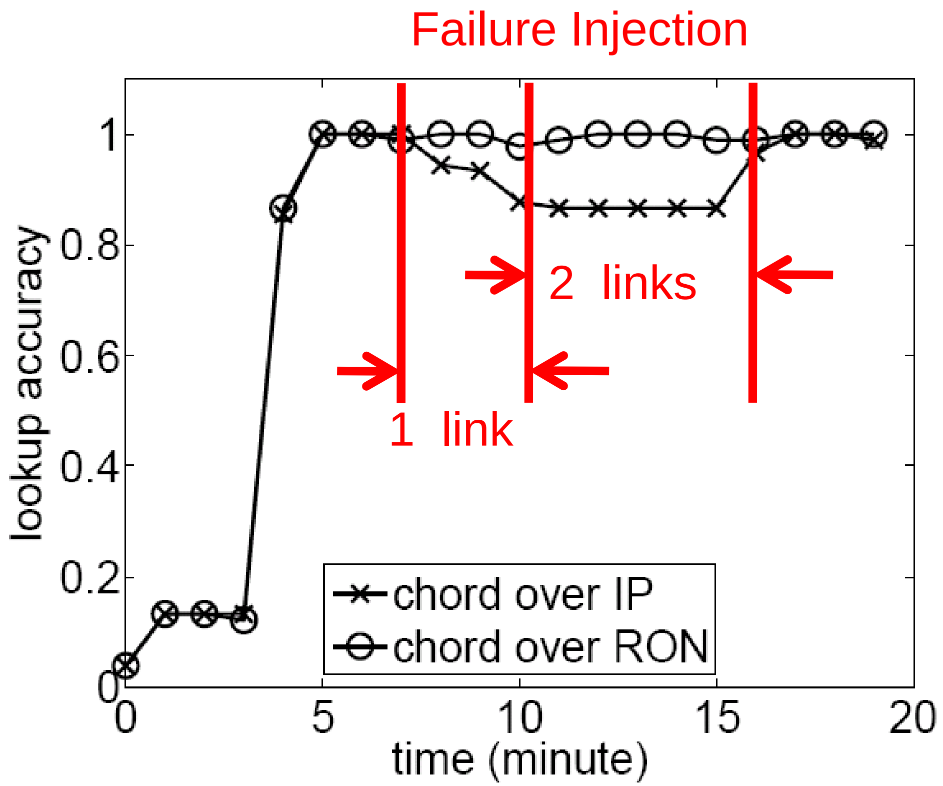
Dynamic Composition

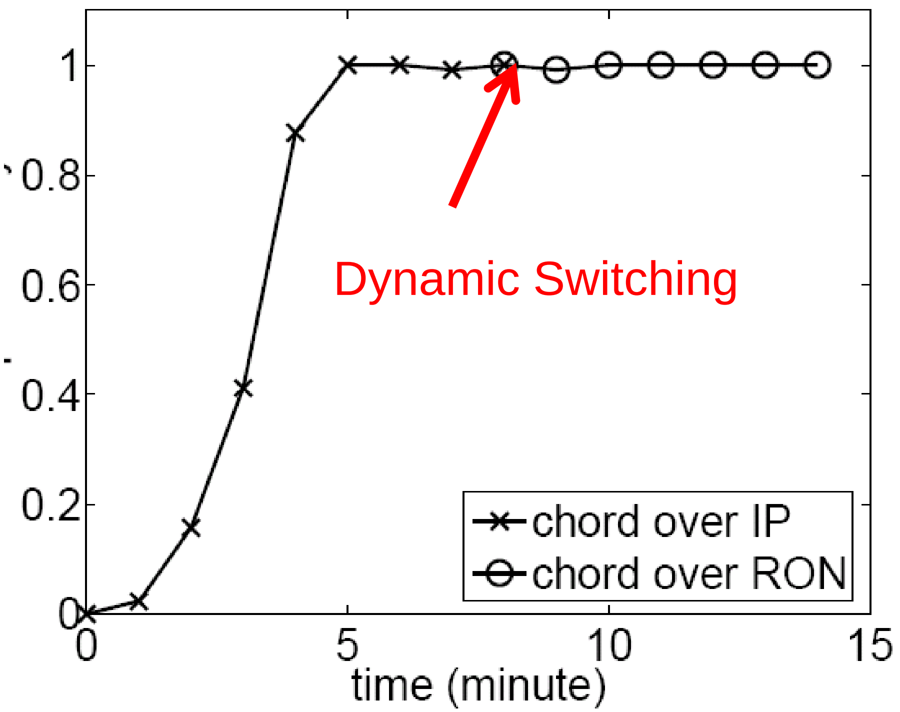
- Underlays are specified *logically* rather than hard-coded
- Bind (or rebind) underlying network address
  - runtime binding = dynamic composition



```
s0 underlay(chord_oid,A):- netAddress(0,A).
s1 underlay(chord_oid,A):- netAddress(OID,A),
                           switchUnderlay(chord_oid,OID).
s2 underlay(ron_oid,A):- netAddress(0,A).
```

Experiment 2: Dynamic Composition





Layering over RON improves DHT lookup accuracy, and can be composed dynamically.

Conclusions

- Contributions:
  - MOSAIC: A unified declarative platform for dynamic network composition
  - Leverages declarative networking techniques
  - Dynamic composition capabilities
  - Proof-of-concept deployment on PlanetLab
- Ongoing and future work:
  - Hybrid adaptable MANET Routing (SIGCOMM PRESTO ’08)
  - Declarative network verification (PADL ‘09)
    - Protocol reasoning, and in future, e2e composition properties

Thank you!

A mosaic is a larger pattern or picture constructed with small pieces of colored glass, stone, or other material.