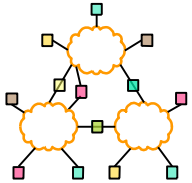


Computer Networks

(Graduate level)

Lecture 8: **SDN, Enabling technologies**

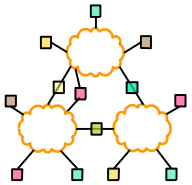
University of Tehran

Dept. of EE and Computer Engineering

By:

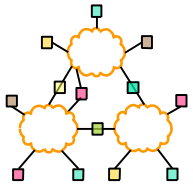
Dr. Nasser Yazdani

Outline



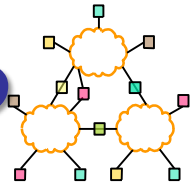
- Active Network
- History of separating control and data
- RCP
- Network virtualization
  - Active Network
  - RCP: Routing Control Platform

What is Active Network?



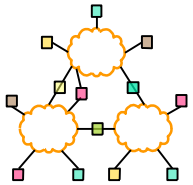
- Normally nodes (routers) passively forward data. What does it mean to make the nodes active?
  - Packets carry code
  - Next hop determined by executing code in a router (i.e. actively as opposed to passive table lookup)
- Sounds clever
  - Can provide customized service to packets
  - Similarities to "customizing" OS services like SPIN and Exokernel
- Gotchas
  - How to write such code?
  - Who can write such code?
  - How to make sure injected code does not break the network?

Why Active Networking?



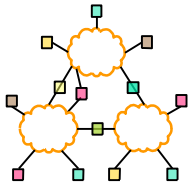
- Motivating problems
  - Passive handling of packets, nothing cab be done!
  - It's too slow/hard to develop and deploy new services on the network (network ossification)
  - Third-party interest in value-added, fine-grain control to dynamically meet the needs of particular applications/network conditions
  - Researcher's desire to experiment at scale
  - Unified control over middleboxes (firewalls, proxies, transcoders)
- Remarkably similar to those of SDN!

Some Acronyms



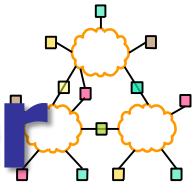
- ANTS
  - Active Node Transfer System
- IETF
  - Internet Engineering Task Force
- MD5
  - Message Digest version 5 (RFC 1321)
    - A one way hash function that generates a 128-bit cipher from arbitrary data
- PIM
  - Protocol Independent Multicast
- PIP
  - Private IP – precursor to IPv6
- MTU
  - Maximum Transmission Unit

Active Networks



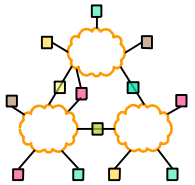
- Introduced by Tennenhouse and Wetherall (1996)
- Vision
  - Architecture for incremental innovation of network services
  - Changing core services (e.g. from IPv4 to IPv6) not easy
  - Allows innovation (by adding services) without full replacement of core services
- Expanded in Wetherall's PhD thesis with example services (1999)
- Worked on by several groups
  - MIT, UPenn, CMU, GT (Calvert and Zegura)
- Impact
  - Difficult to say…CISCO is router king…"don't want you messing around with my routers" attitude

Takeaways of this paper



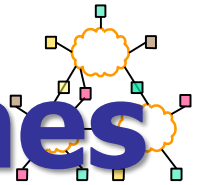
- Benchmarking original vision against reality
- Specific contributions
  - A programming model and implementation
  - ANTS toolkit
  - Feasibility of router programming with ANTS

Leading up to Active Networks



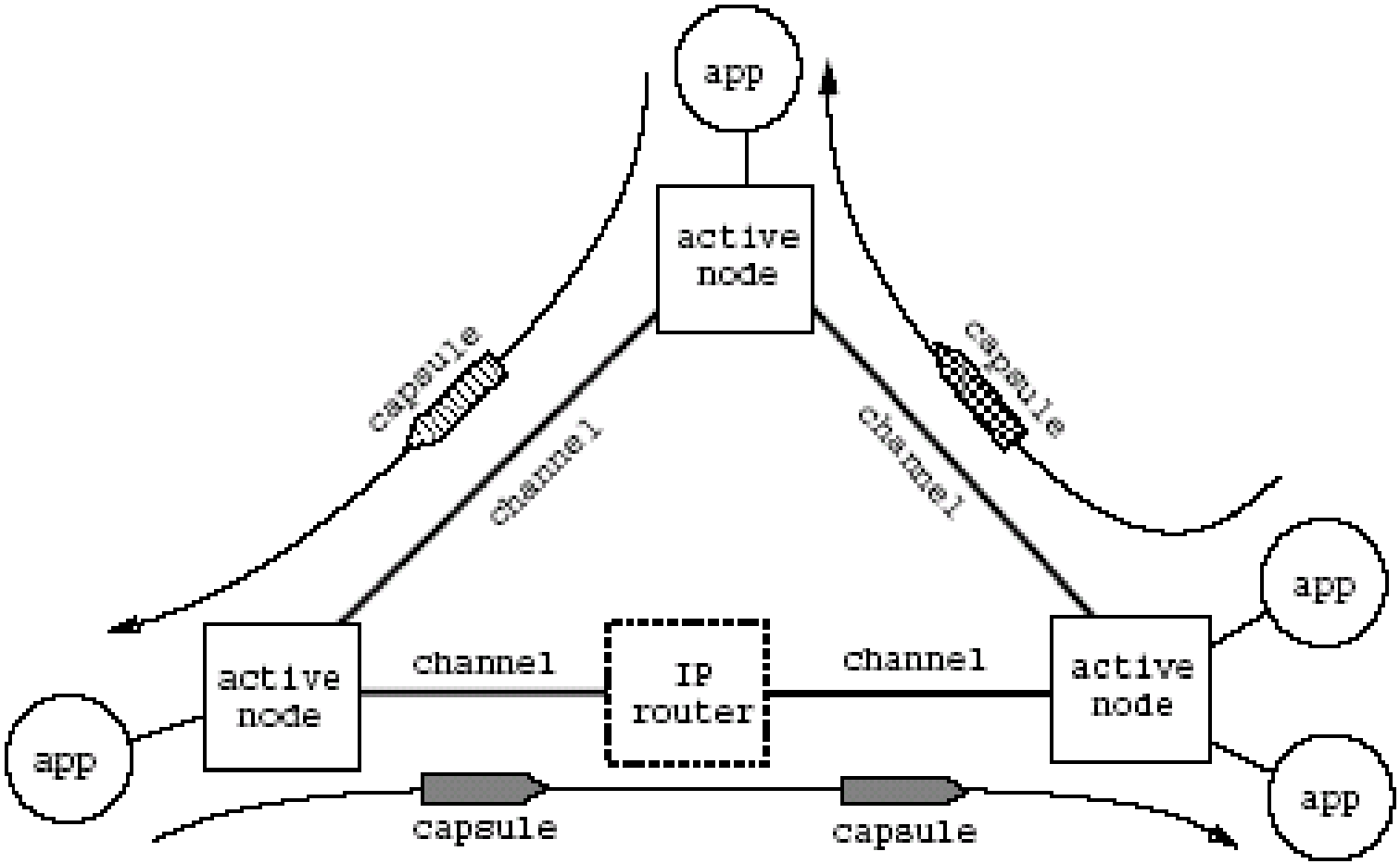
- Extensible individual routers for network admins to program (e.g. firewall code)
- Programmability across multiple nodes for control tasks (e.g. traffic measurement, route discovery)
- Limitations
  - Who can program
  - Where can such programs run
- Active networks
  - Remove both these restrictions

Two Different Approaches

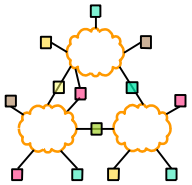


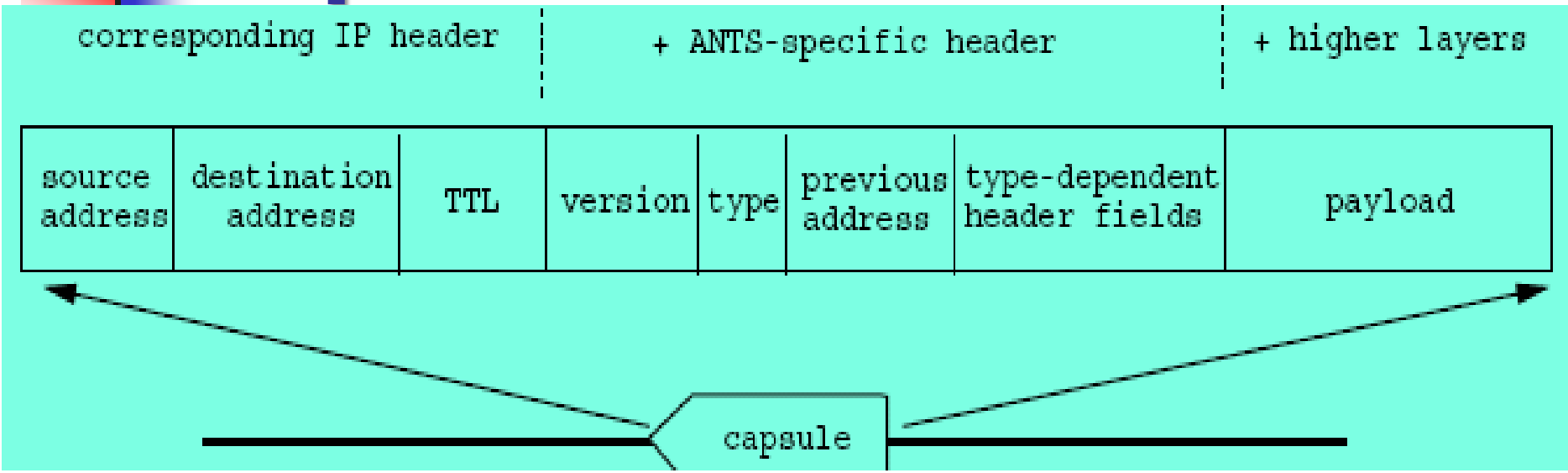
- Capsules ("integrated")
  - Every message is a program. Active nodes evaluate content carried in packets.
  - Code dispatched to execution environment
- Programmable Switches ("discrete")
  - Custom processing functions run on the routers
  - Packets are routed through programmable nodes
  - Program depends on the packet header

ANTS Active Network



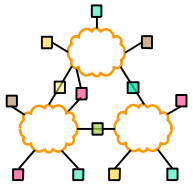
Capsule





- Originally extension of IP packet format
  - ANTS toolkit uses UDP datagrams in overlay with capsule as payload
- Capsules are typed
  - Type is an MD5 fingerprint
    - Identifies the code that should handle the capsule

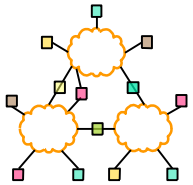
API for coding active nodes



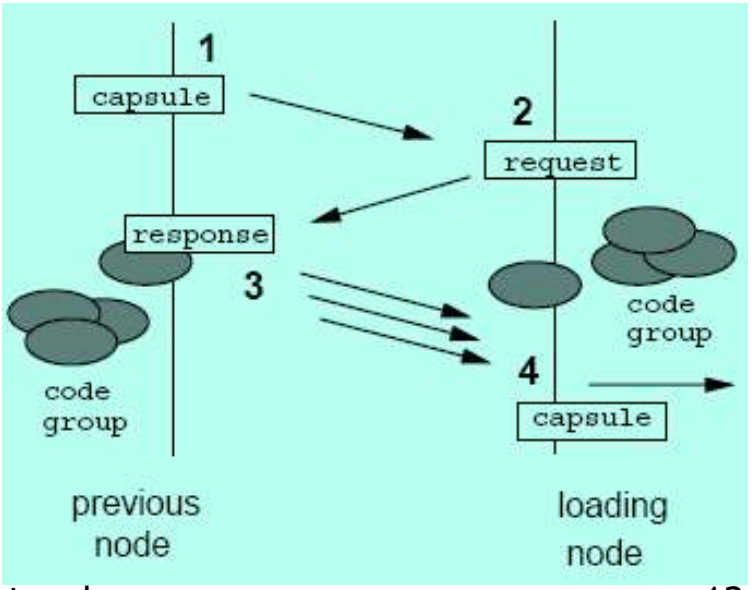
Method	Description
`int getAddress()`	Get local node address
`ChannelObject getChannel()`	Get receive channel
`Extension findExtension(String ext)`	Locate extended services
`long time()`	Get local time
`Object put(Object key, Object val, int age)`	Put object in soft-store
`Object get(Object key)`	Get object from soft-store
`Object remove(Object key)`	Remove object from soft-store
`void routeForNode(Capsule c, int n)`	Send capsule towards node
`void deliverToApp(Capsule c, int a)`	Deliver capsule to local application
`void log(String msg)`	Log a debugging message

- Three categories
  - Query node
  - Manipulate soft-store
  - Route capsule

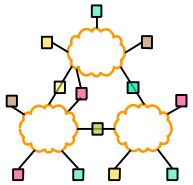
Implementation



- Capsule code passed by reference, why?
  - Type field serves as the fingerprint of the capsule
  - Active node "demand loads" capsule code
    - Either in soft store
    - Or from previous address (and cached for subsequent use)
    - Or drop packet (higher levels will take care of retransmission)
- ANTS implemented in Java (nothing fundamental here…just convenience)

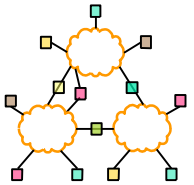


Capsule feasibility



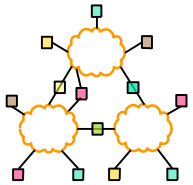
- Capsules represent in-network processing
- Why is this a problem?
  - Routing now happens in *software*
  - Plus there is overhead for processing
- Software based routing not feasible always
  - Commercial hardware routers
    - Forward 4 M packets/sec (70 byte packets) => nGbps throughput
  - PC-based routers
    - 70K packets/sec => cannot keep up the forwarding rate
- ANTS architecture
  - Partially active nodes (for example at the edges); core routers simply forward (i.e. not active)
- Code spoofing
  - MD5 fingerprint ensures no spoofing (locally verified)

ANTS Forwarding performance

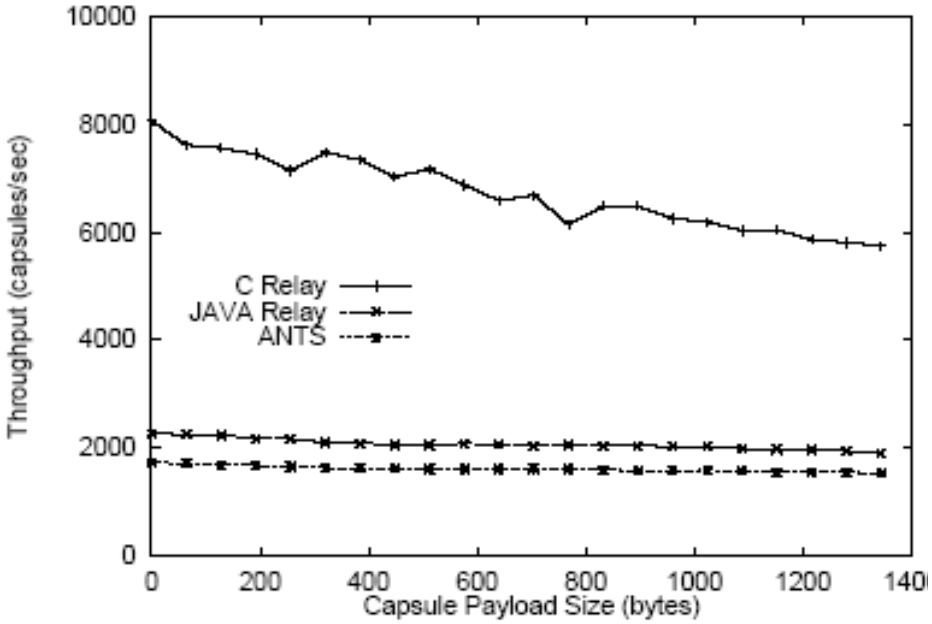


- Three reasons to feel good
  - Adequate for access links up to T1 (1.5 Mbps)
  - Comparable to a Java based relay
  - No data dependent loss of performance (Java, C, and ANTS perf lines same slope in Figures 4 and 5)
- ANTS performance can be much better
  - With C implementation
  - In kernel implementation

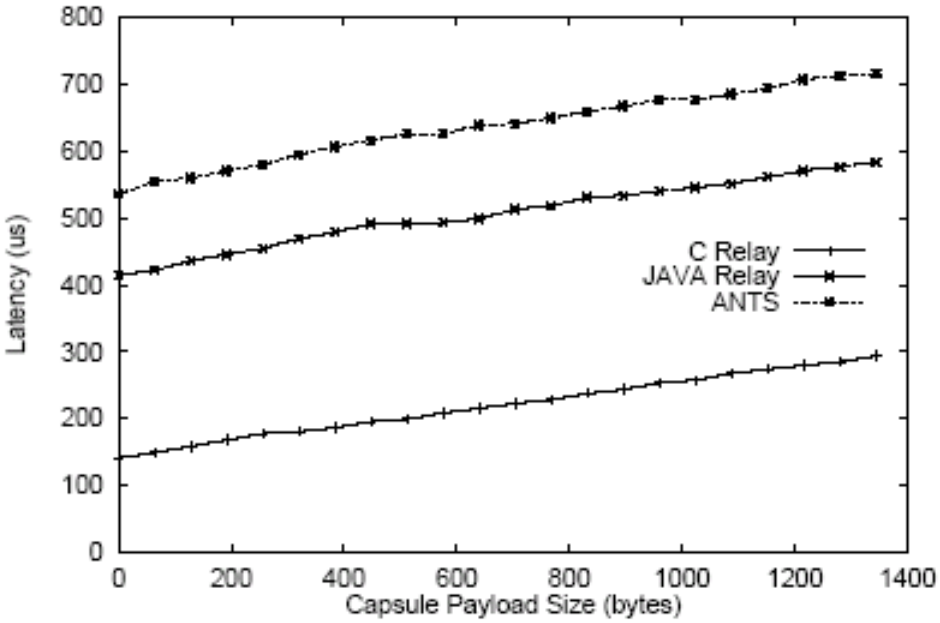
Performance



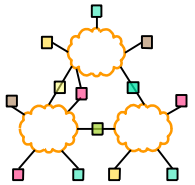
Thruput



Latency



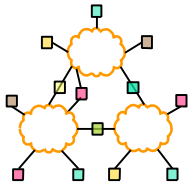
Profile of Capsule Processing



- Xmit/recv (1 and 9) can be eliminated by in-kernel implementation
- Enc/Dec (4 and 7) due to Java
- Leaves 30% overhead compared to IP

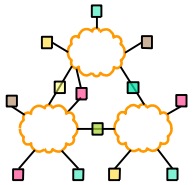
Operation	IP?	Time (μs)	(%)
1. Packet Receive	no	180	29
2. Header Processing	yes	30	5
3. Type Demultiplex	no	20	3
4. Capsule Decode	no	110	18
5. Capsule Evaluate	no	10	2
6. Route Lookup	yes	30	5
7. Capsule Encode	no	90	14
8. Header Processing	yes	40	7
9. Packet Transmit	no	80	13
Other	n/a	25	4
Total	n/a	615	100

Security and Safety



- Who can introduce new services? – two basic concerns
  - Protection – can my service damage yours (intentionally or not)
  - Resource management – can my service consume arbitrary resources

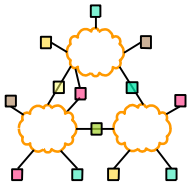
Protection threats



- Corrupt the ANTS runtime at a node?
  - Java sandboxing (other implementations possible (PCC, SFI)
- Code spoofing (incorrect or corrupted code)?
  - Capsule type is a MD5 fingerprint of the code – no way to spoof unless you can break MD5
- Corrupt soft-state at node?
  - Restricted node API
  - Soft-state also controlled by fingerprint
    - A class can only access its own soft-state
  - State sharing through hierarchy of fingerprint types

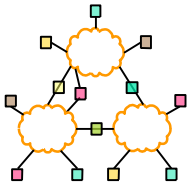
Summary: ANTS as good (or bad!) as Internet

Resource management threats



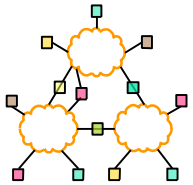
- Unconstrained resource consumption at a single node?
  - Programming model is restricted
    - ANTS node runtime enforces resource bounds
- Unconstrained resource consumption at multiple nodes
  - Watchdog timers
  - Capsule TTL fields
- An application floods the network with capsules, starving other apps?
  - Punt (the Internet doesn't address this either)

Resource Management Threats



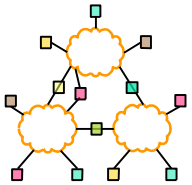
- A capsule consumes large amounts of resources across a subset of the network?
  - By creating other capsules and routing them
  - Internet is static; ANTS is not
  - Per capsule hop limit?
    - Can still inflict damage
    - Some behaviour is OK (multicast) – hop limit not related to topology
  - Punt (rely on certified code => limits 'any user can introduce new code' idea)

Applications can be built?



- Ones that are difficult to deploy in the Internet
  - not tied to administrative setup, which tends to mirror physical setup
  - services whose implementation is spread along the datapath
  - make use of network level info: topology, loss, load…
  - variants of routing, forwarding, flow setup, congestion handling
- Characteristics
  - Expressible – restricted API
  - Compact – limit of 16KB
  - Fast – node runtime is limited, no blocking
  - Incremental – not all nodes are active nodes
- Network layer services tend to be best for this, rather than application code
- In general There is no "killer app" for an "Extensible foo"!!

Example Services



- Multicast (PIM-style)
- Reliable multicast
- Explicit congestion notification
- PIP
- Anycasting

Early Days at Telephone

- Control and Data Together
  - In-band signaling
    - Data and control sent over the same channel
    - Certain frequencies (e.g., 2600 Hz) could reset phone trunk lines, route calls
- Resulting network was brittle, insecure, etc

Network Control Point

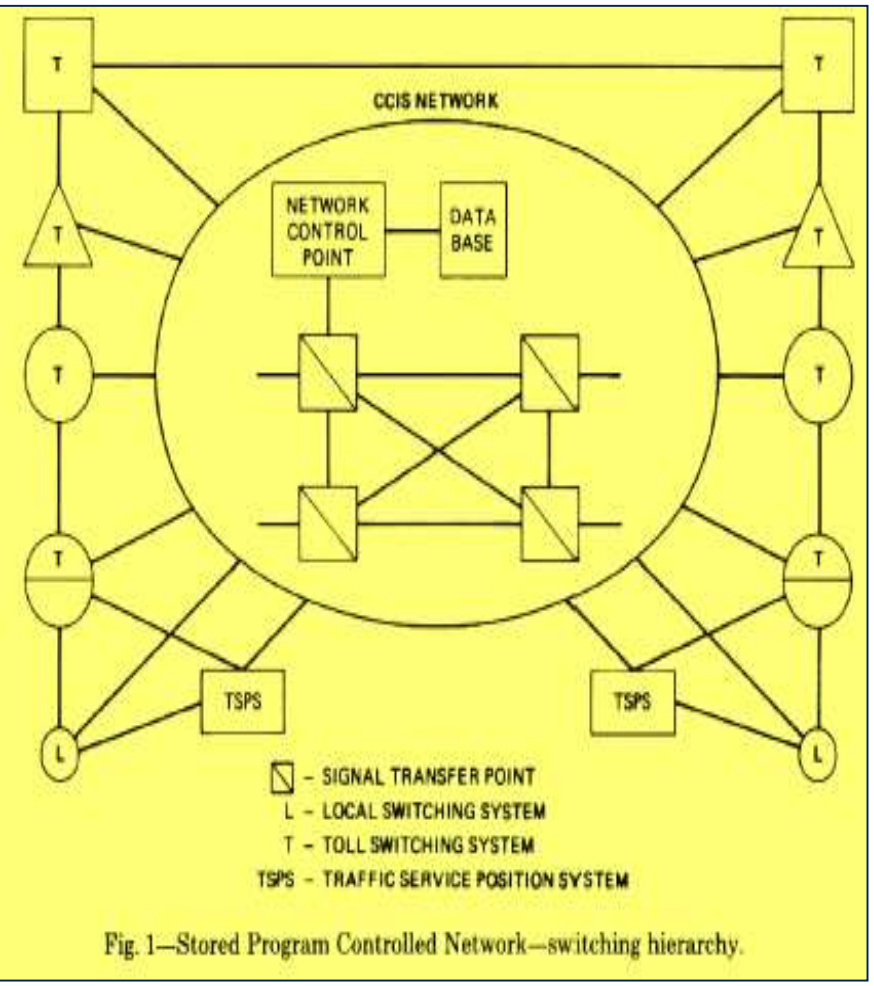
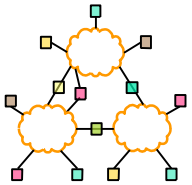


Fig. 1—Stored Program Controlled Network—switching hierarchy.

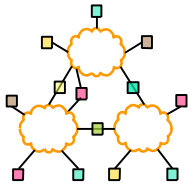
- Introduced in 1981
- Signaling at NCP
- Benefits
  - Services on demand
  - Rapid introduction of new services

Benefits of NCP



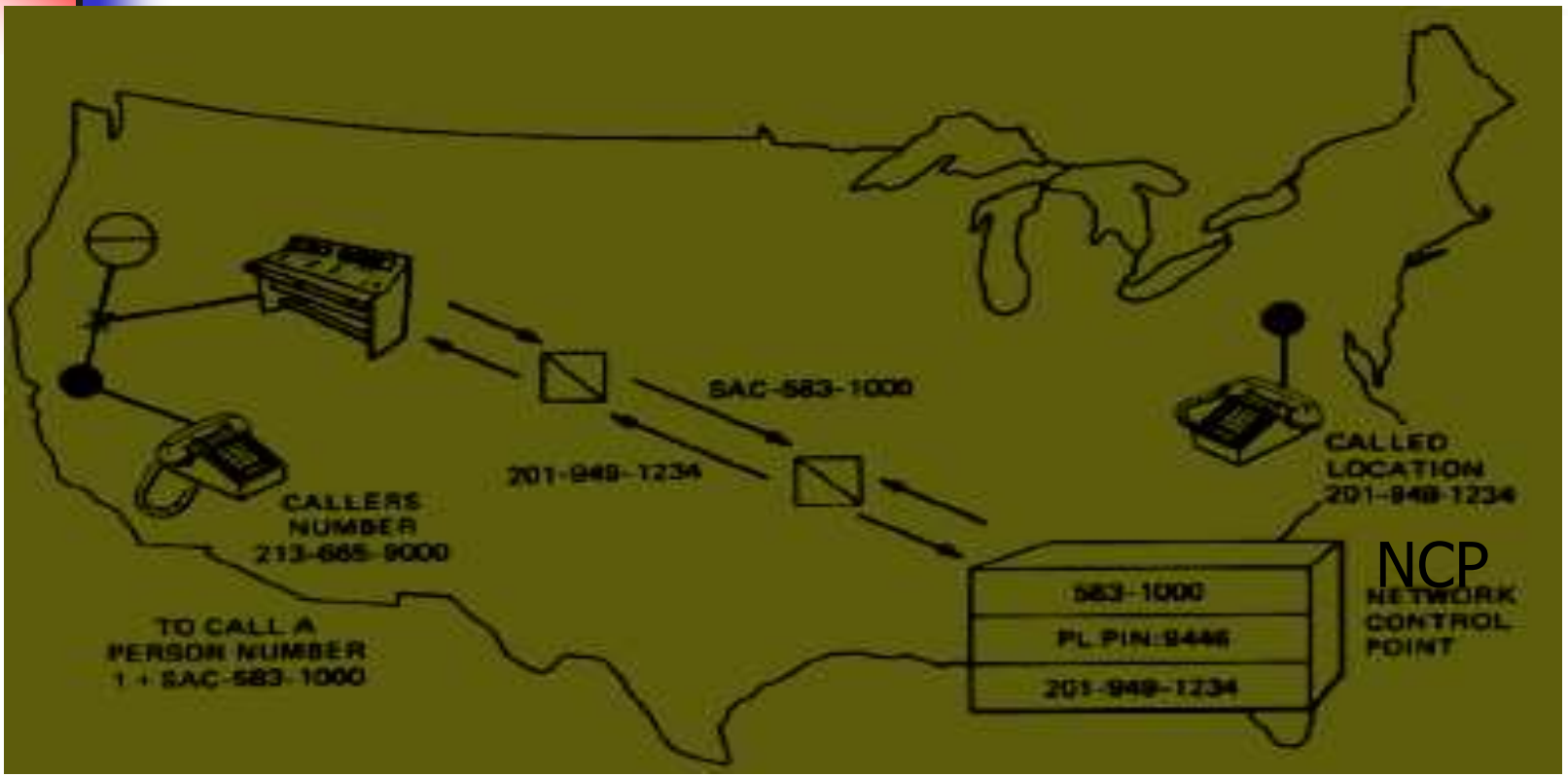
- Elimination of in-band signaling reduces expenditures
  - Shorter circuit holding time
  - Ability to determine busy/idle status before requesting a circuit
- Rapid introduction of new services
  - "In the area of new services that can be supported…possibilities are limited only by imagination.

Apps Basic Primitives



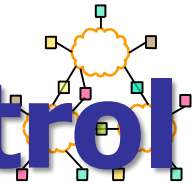
- Collect N digits
- Send a message to the NCP
- Make a billing record

Person Locator



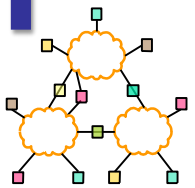
- User registers location with NCP database
- NCP routes call to the current location/ number
- NCPs currently used to route 800 calls

Benefits of Central Control



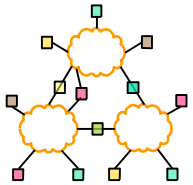
- Network-wide vantage point
  - Can directly observe (rather than infer) network wide behavior
- Independent evolution of infrastructure, data, and services
  - Services and resource allocation decisions can be made based on customer data, network load, etc.

Routing Control platform (RCP)



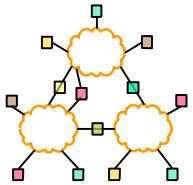
- What's wrong with Internet routing?
  - Full-mesh iBGP doesn't scale
  - Route-reflectors help by introducing hierarchy
    - but introduce configuration complexity, protocol oscillations/loops
  - Hard to manage
    - Many highly configurable mechanisms
    - Difficult to model effects of configuration changes
    - Hard to debug
  - Hard to evolve and provide new services, improve upon protocols

Problems with BGP



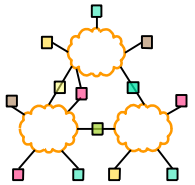
- BGP is broken
  - It converges slowly, sometimes not at all
  - It causes routing loops
  - It's misconfigured frequently
  - Traffic engineering is hard
- Fixing BGP is hard
  - Incremental fixes: Makes BGP even more complicated
  - New architectures and inter-domain protocols: Deployment is almost impossible

BGP Problems

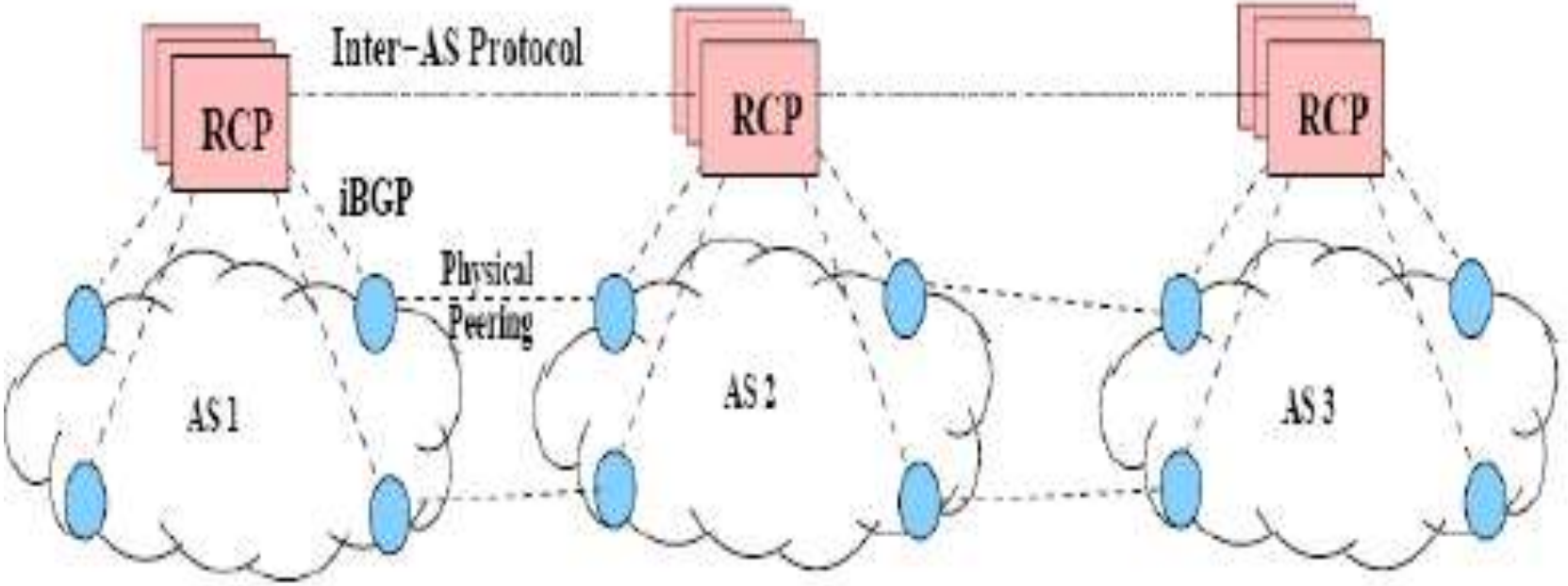


- AS is the logical entity for inter-domain
  - BGP state, logic are decomposed across routers
  - No router has complete BGP state
  - Each router makes routing decision based on partial and incomplete view
- BGP interacts in odd ways with other protocols
  - Most notably with the IGP (Interior Gateway Protocol) running inside an AS

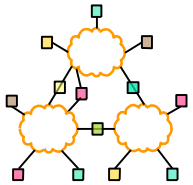
Possible solution



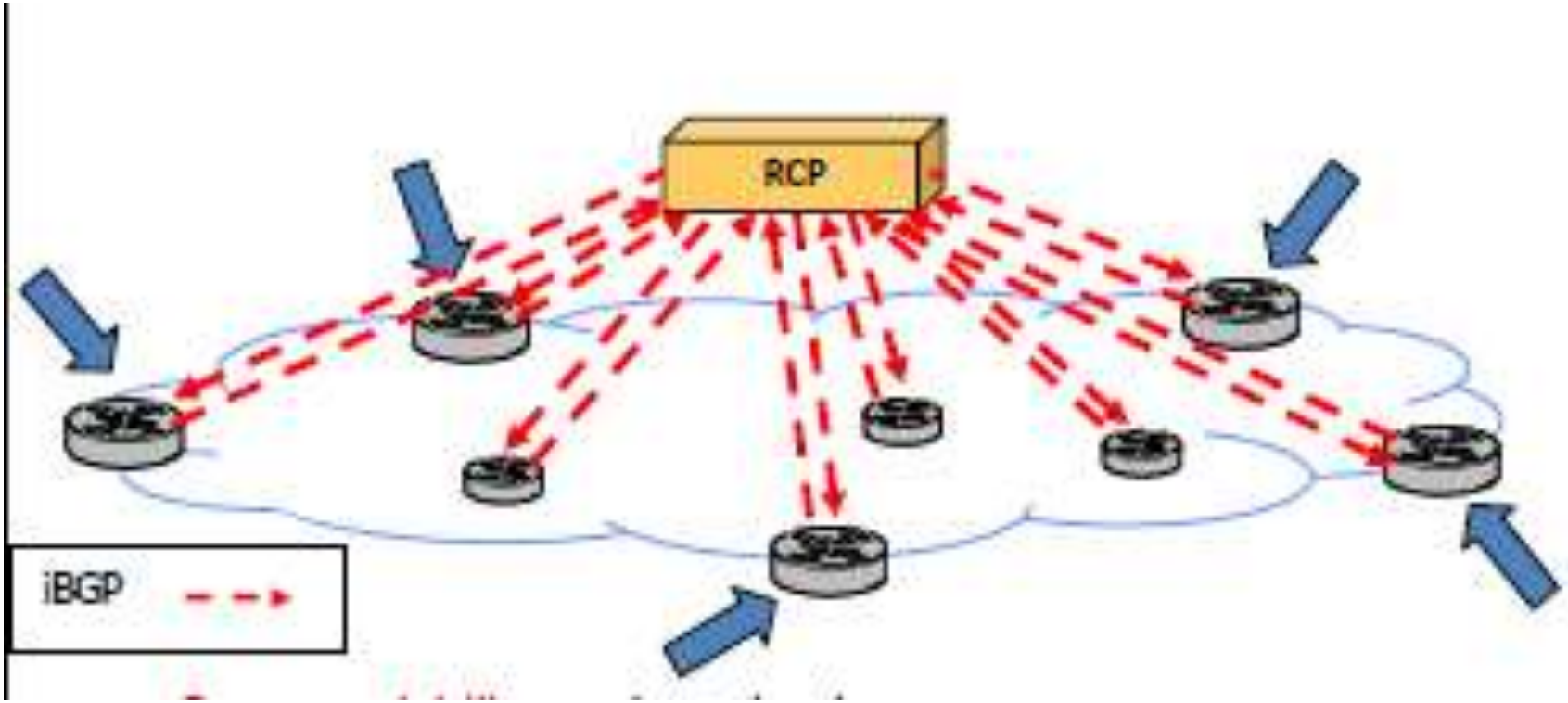
- Route Computation logic should have "full view" of the network state
- • Control Routing Protocol Interactions



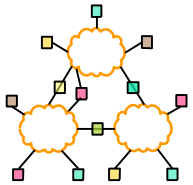
In an ISP

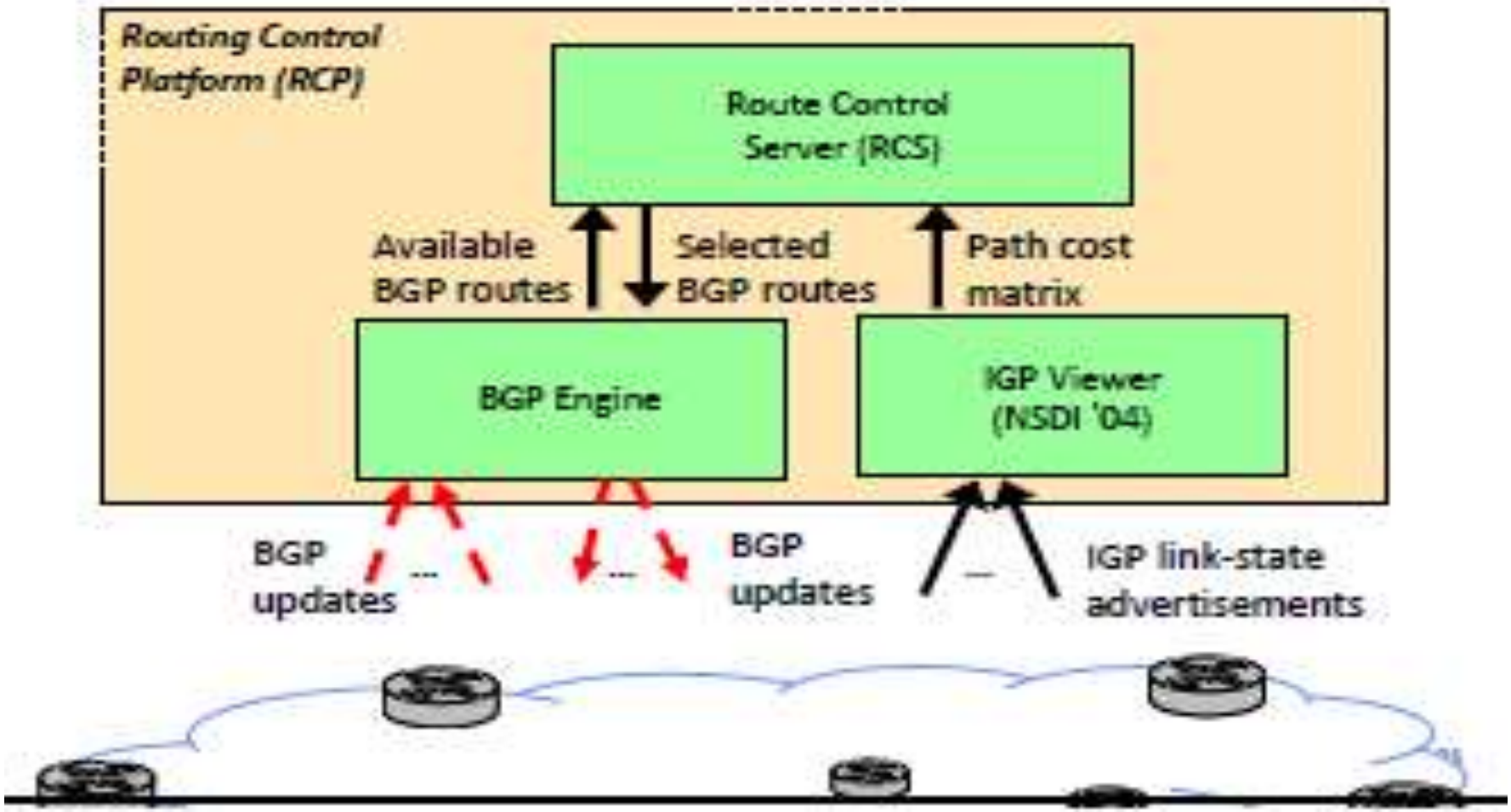


- Better scalability: reduces load on routers
- Easier management: configuration from a single point
- Easier evolvability: freedom from router software

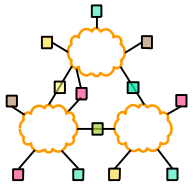


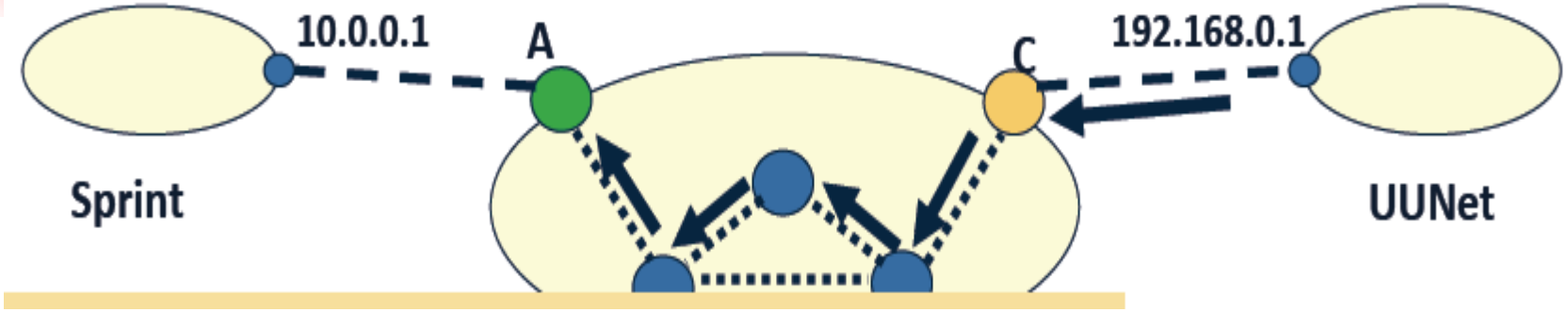
RCP architecture





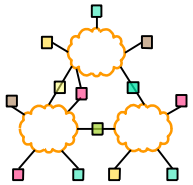
Decomposed Configuration State





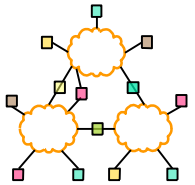
- Simple policy: "Don't advertise routes learned from UUNet to Sprint"
- Configuration is decomposed, so routes must carry state
  - **Assign routes "From UUNet" tag at router C**
- With RPC is easier, We have all information

More Flexible Routing



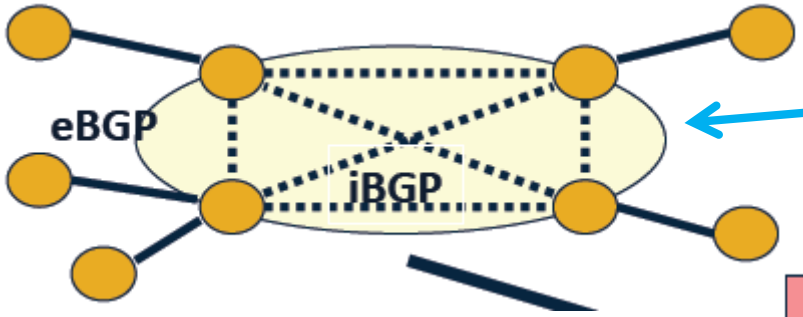
- Better network management
  - Diagnostics and trouble-shooting
  - Routing co-located with other information (e.g. traffic)
  - Ability to reason about an AS as a single entity
- Protocol Improvements
  - Attaching prices to routes
  - Inter-AS negotiation of exit points
  - Overlay routing informed by IP-layer information

In summary



- RCP embodies two principles for inter-domain routing
  - Treat an AS as a single logical entity
    - Compute consistent routes using complete AS-wide view
  - Control routing protocol interactions
- Benefits
  - Simpler, more expressive configuration
  - Intrinsic robustness: no loops, faster convergence
  - · Enable new applications and innovations
    - Opportunity for new traffic engineering applications

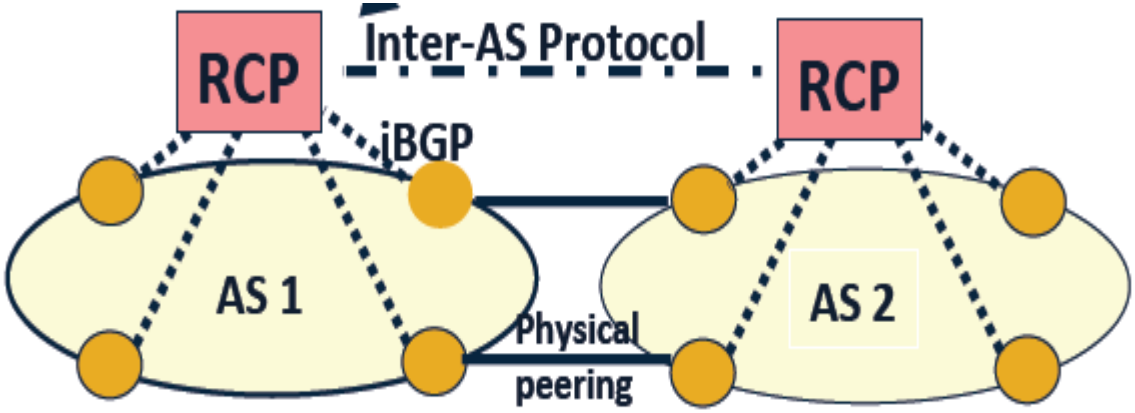
Deployment



Current state

Three phases:

1. iBGP
2. eBGP
3. final

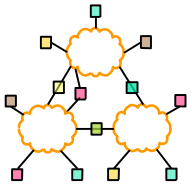


- Two issues
  - Backward compatibility
  - Deployment incentives

Challenges and contributions

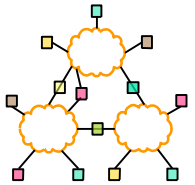
- Reliability
  - Problem: single point of failure
  - Solution : replication of RCP components
- Consistency
  - Problem: inconsistent decisions by replicas
  - **Solution** : guaranteed consistency without inter-replica protocol
- Scalability
  - Problem: large cpu/memory usage
  - **Solution** : Support large ISP in one computer

Scalability solution



- Eliminate redundancy
  - Store only a single copy of each BGP route
- Accelerate lookup
  - Quickly find routers whose routes changed
- Avoid recomputation
  - Compute routes once for groups of routers
  - Don’t recompute if relative ranking of egress routers unchanged

Next-lecture: SDN, Data Plane



- Assigned reading
  - Click: a modular software router
  - RouteBricks: Exploiting Parallelism To Scale Software Routers
  - Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN
  - SwitchBlade: A Platform for Rapid Deployment of Network Protocols on Programmable Hardware