The background of the slide is a photograph of a desert landscape. It features rolling sand dunes in the distance, a cluster of palm trees in the middle ground, and a sandy foreground with sparse, low-lying vegetation. The sky is a clear, bright blue.

The Computer is the Network: The Emergence of Programmable Network Elements

Randy H. Katz
Computer Science Division
Electrical Engineering and Computer Science Department
University of California, Berkeley
Berkeley, CA 94720-1776

Presentation Outline

- Technology Trends and Implications
- SAHARA Service Architecture
- OASIS Programmable Network Elements
- RADS Distributed Architecture
- Summary and Conclusions

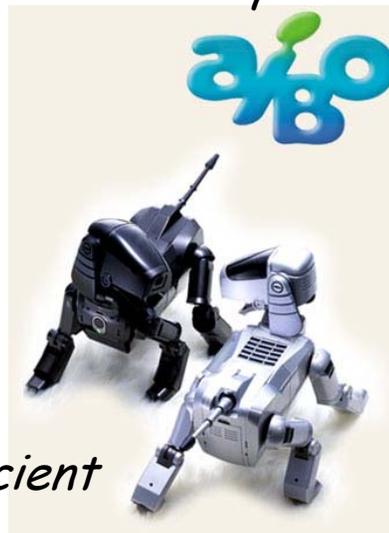
Presentation Outline

- Technology Trends and Implications
- SAHARA Service Architecture
- OASIS Programmable Network Elements
- RADS Distributed Architecture
- Summary and Conclusions

Shape of Things Today: Diverse Appliances and Devices



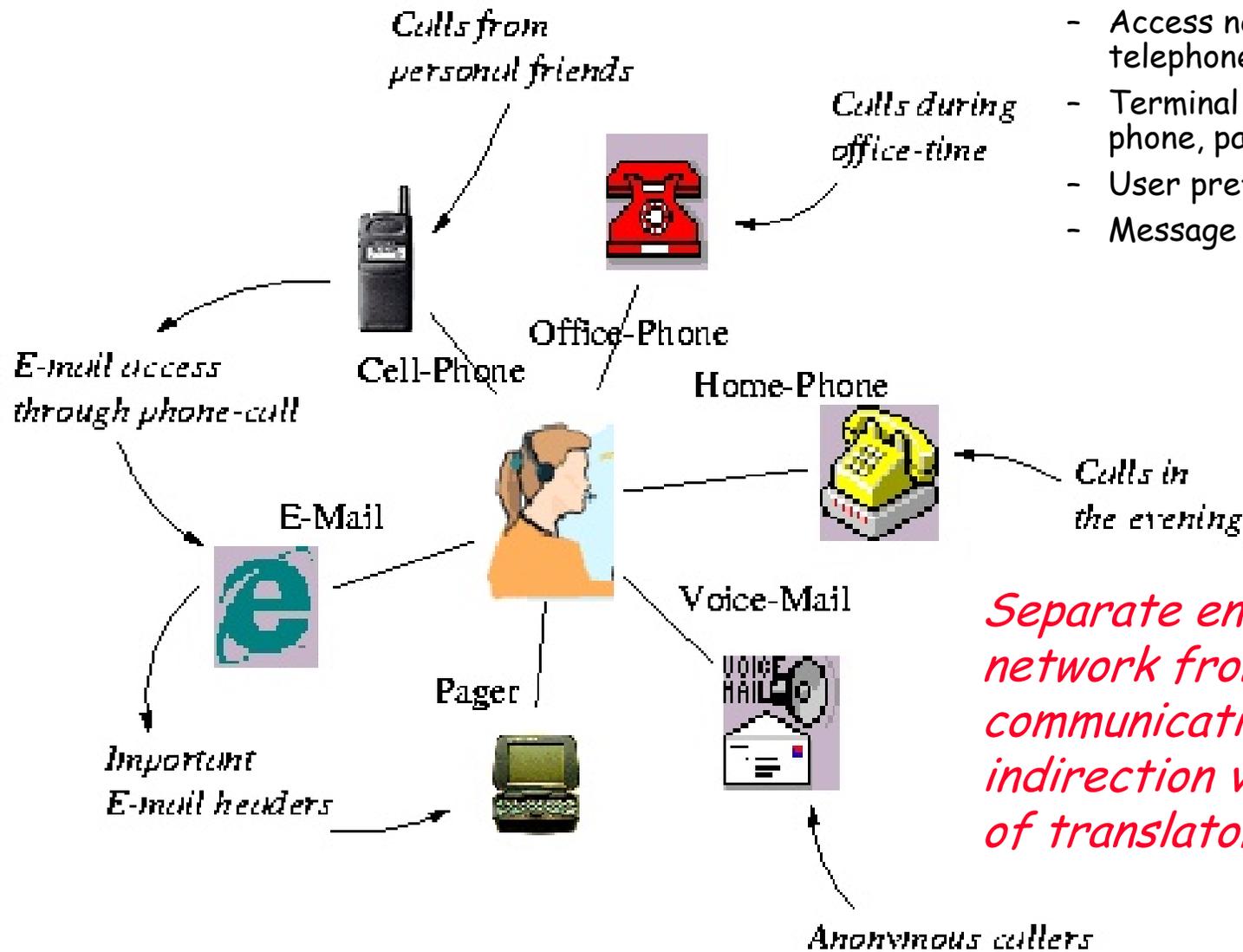
Game Consoles
Personal Digital Assistants
Digital VCRs
Communicators
Smart Telephones
E-Toys



*All will demand broadband
Internet connectivity*

... and 10BaseT won't be sufficient

Composed Applications: Universal In-box



- Message type (phone, email, fax)
- Access network (data, telephone, pager)
- Terminal device (computer, phone, pager, fax)
- User preferences & rules
- Message translation & storage

Separate end device and network from end-to-end communications service: indirection via composition of translators with access

Best Implementation Method: the Internet Programming Model

- Service composition across the network
 - Network-aware Distributed System architecture
- Bottlenecks near edge, not core
 - Deploy services close to where used
 - Service implementation topology-aware
- Enabled by:
 - Computing embedded in communications fabric: distributed, wide-area, topology-aware
 - Per session characterization, processing, prioritization, monitoring, management, billing

Presentation Outline

- Technology Trends and Implications
- **SAHARA Service Architecture**
- OASIS Programmable Network Elements
- RADS Distributed Architecture
- Summary and Conclusions

SAHARA

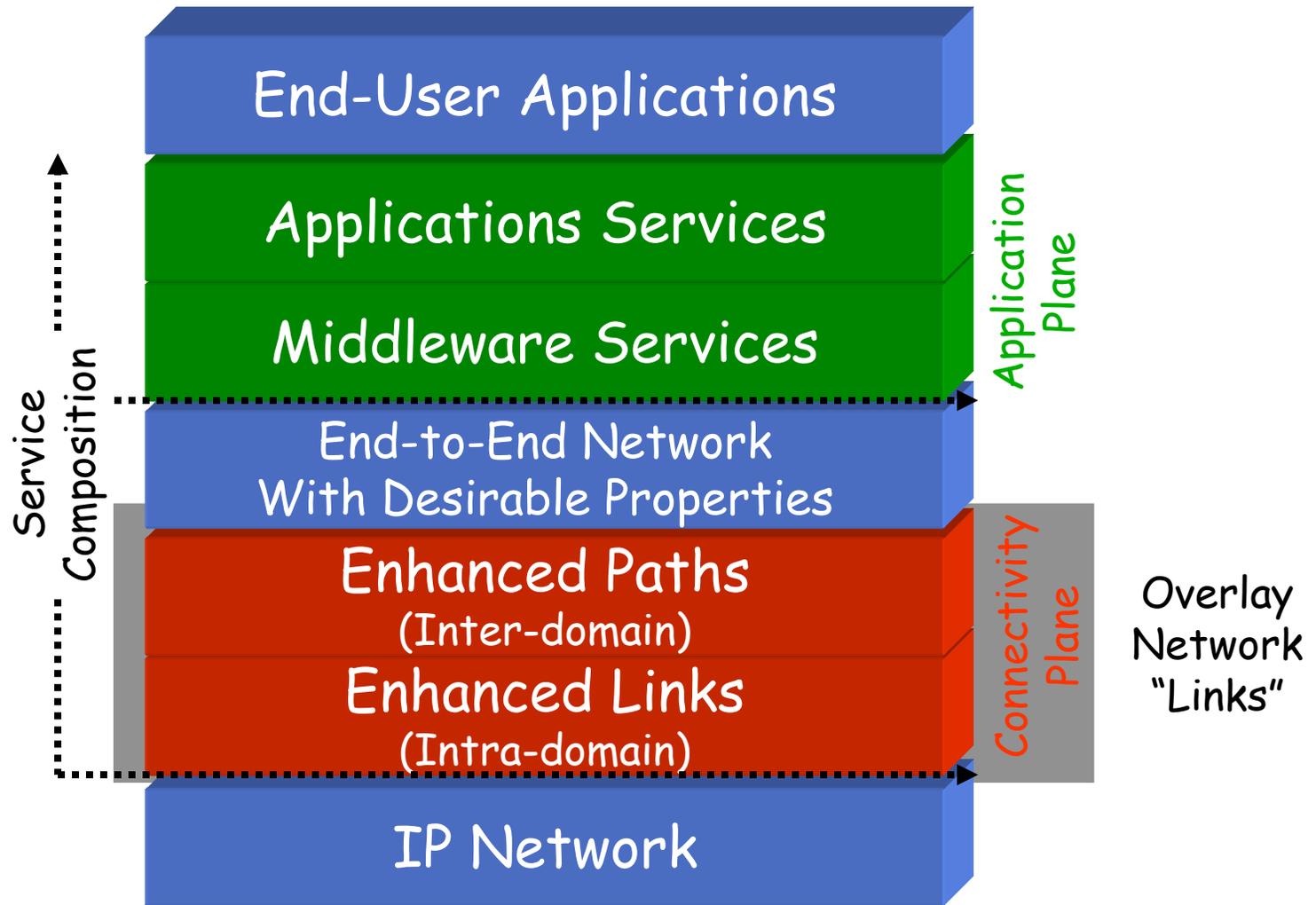
- **S**ervice
- **A**rchitecture for
- **H**eterogeneous
- **A**ccess,
- **R**esources, and
- **A**pplications



SAHARA Vision

- Achieving end-to-end services with desirable, predictable, enforceable properties spanning potentially distrusting service providers
- Via service composition and inter-operation across separate admin domains, supporting peering and brokering, and diverse business, value-exchange, access-control models
- Focus on interdomain routing, overlay networks, p2p algorithms
- Effective way to more rapidly extend and deploy enhanced wide-area network functionality

Layered Reference Model for Service Composition



Routing as a Composed Service

- Routing as a Reachability "Service"
 - Paths between composed service instances--"links" within an overlay network
 - Multi-provider environment, no centralized control
- Desirable Enhanced Properties
 - *Trust: verify believability of routing advertisements*
 - *Dependability: Identifying the sources of lost connectivity*
 - Reliability: detect service composition path failures quickly to enable fast recomposition to maintain E2E service
 - Agility: converge quickly in response to global changes to retain good reachability "performance"

Trusting the Infrastructure: BGP Route Verification

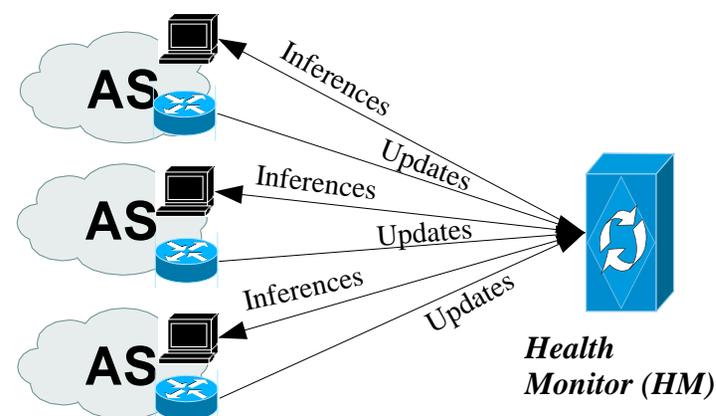
- BGP protocol is highly vulnerable!
 - Problem: BGP allows AS's to propagate invalid routes deviating from actual Internet topology
 - Significant performance and correctness implications
 - » Misconfigured routers can cause long outages
 - Drop packets ("blackholes")
 - Roughly 6% of misconfigurations cause reachability problems
 - » Malicious routers can cause even greater damage
 - Misroute or eavesdrop on traffic
 - Impersonate destinations
 - Collude with other nodes to make detection difficult

"Listen" and "Whisper"

- Detect inconsistencies in route advertisements to identify suspicious AS's to be avoided for routing purposes
 - *Listen*: "Passive" TCP-probing
 - » Modified nodes watch TCP traffic to detect reachability problems
 - » No mods to BGP, incrementally deployable
 - » But ineffective for malicious host detection: can't distinguish between genuine and malicious hosts
 - *Whisper*: Advertisements sent consistent with those received
 - » Route advertisement invalid if AS-PATH does not match its propagation path
 - » Use redundant net connectivity to verify route consistency

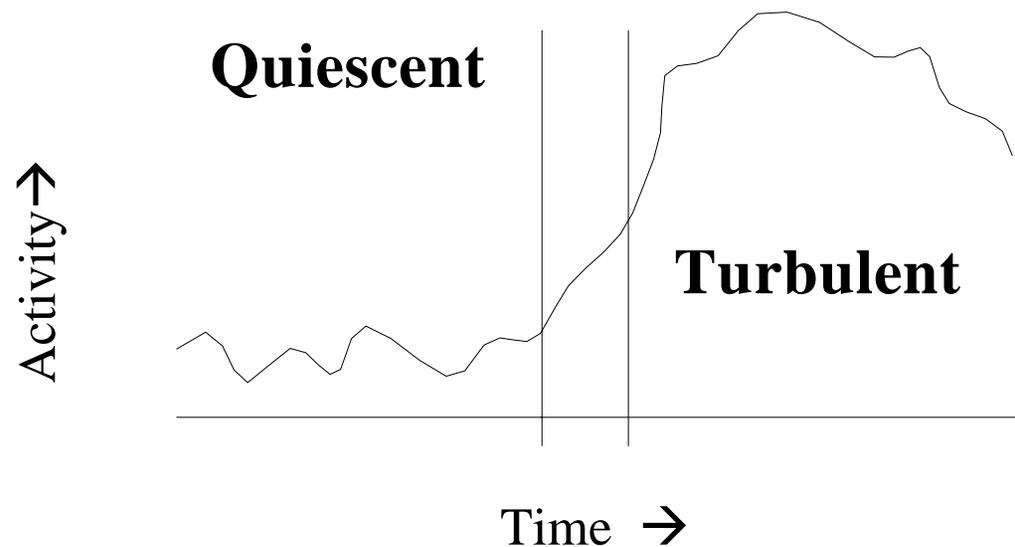
Depending on the Infrastructure: BGP Fault Localization

- Interdomain routing suffers from many problems
 - Instability, slow convergence, misconfigurations
- Poor visibility into dynamics
 - What is the spectrum of causes of route changes?
 - What are the primary causes of instability?
 - How does BGP respond to a routing change?
- BGP Health Monitoring System
 - Collect routes from routers
 - Infer properties of network elements
 - Redistribute information
 - Achieves greater visibility

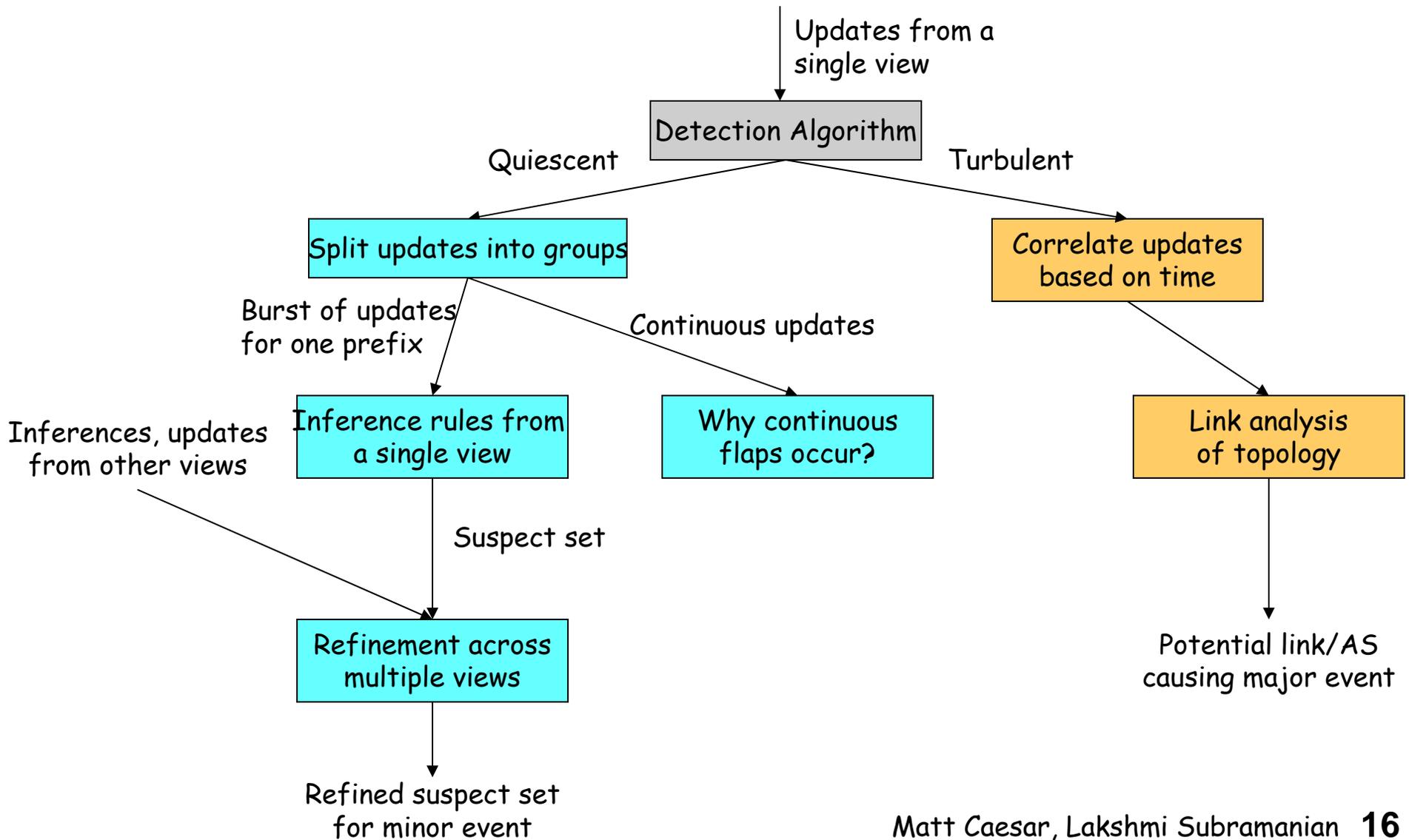


Fault Inference

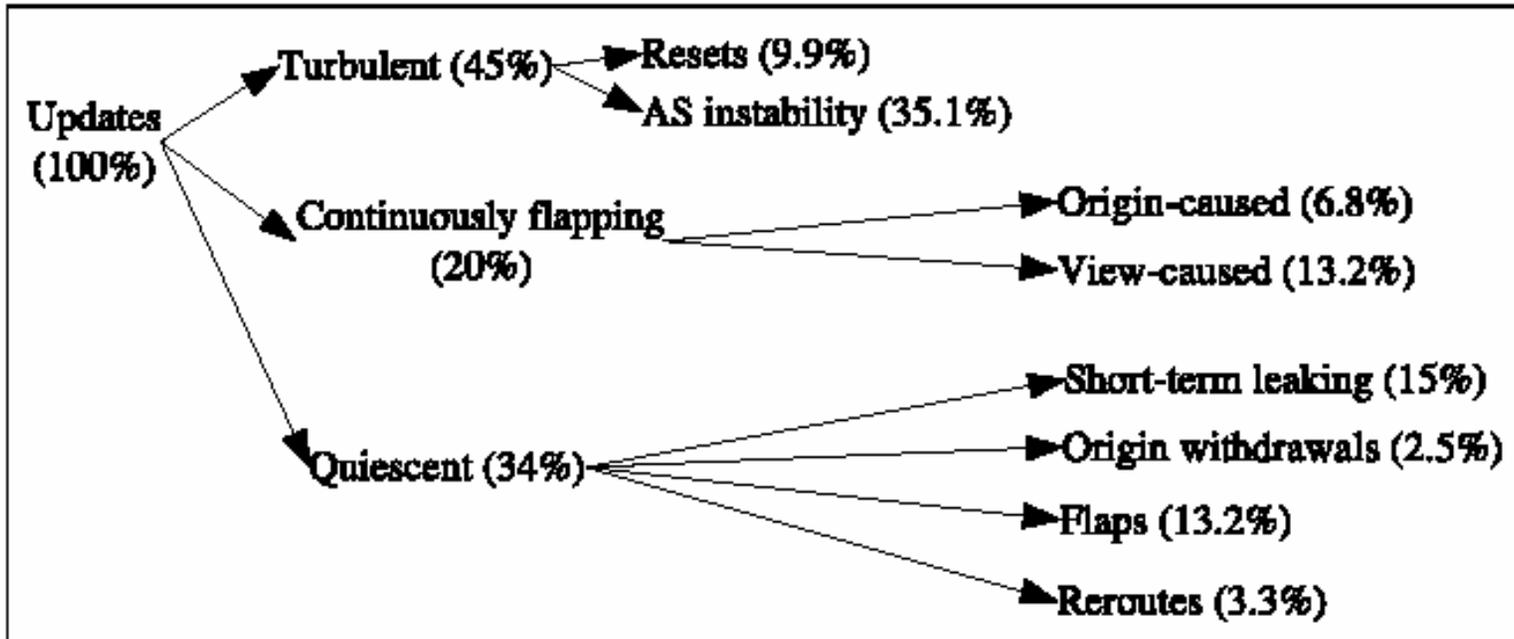
- Given route updates observed at multiple vantage points, determine the **suspect set** = {**suspect cause set**, **suspect location set**} of routing events that trigger each update



Inference Approach



Inference Results



- 70% of updates can be pinpointed to a single inter-AS link (pair of AS's)
 - More precise inference for more major events
- Few AS's, links causing majority of updates

Presentation Outline

- Technology Trends and Implications
- SAHARA Service Architecture
- **OASIS Programmable Network Elements**
- RADS Distributed Architecture
- Summary and Conclusions

Overlays and
Active
Services for
Inter-networked
Storage

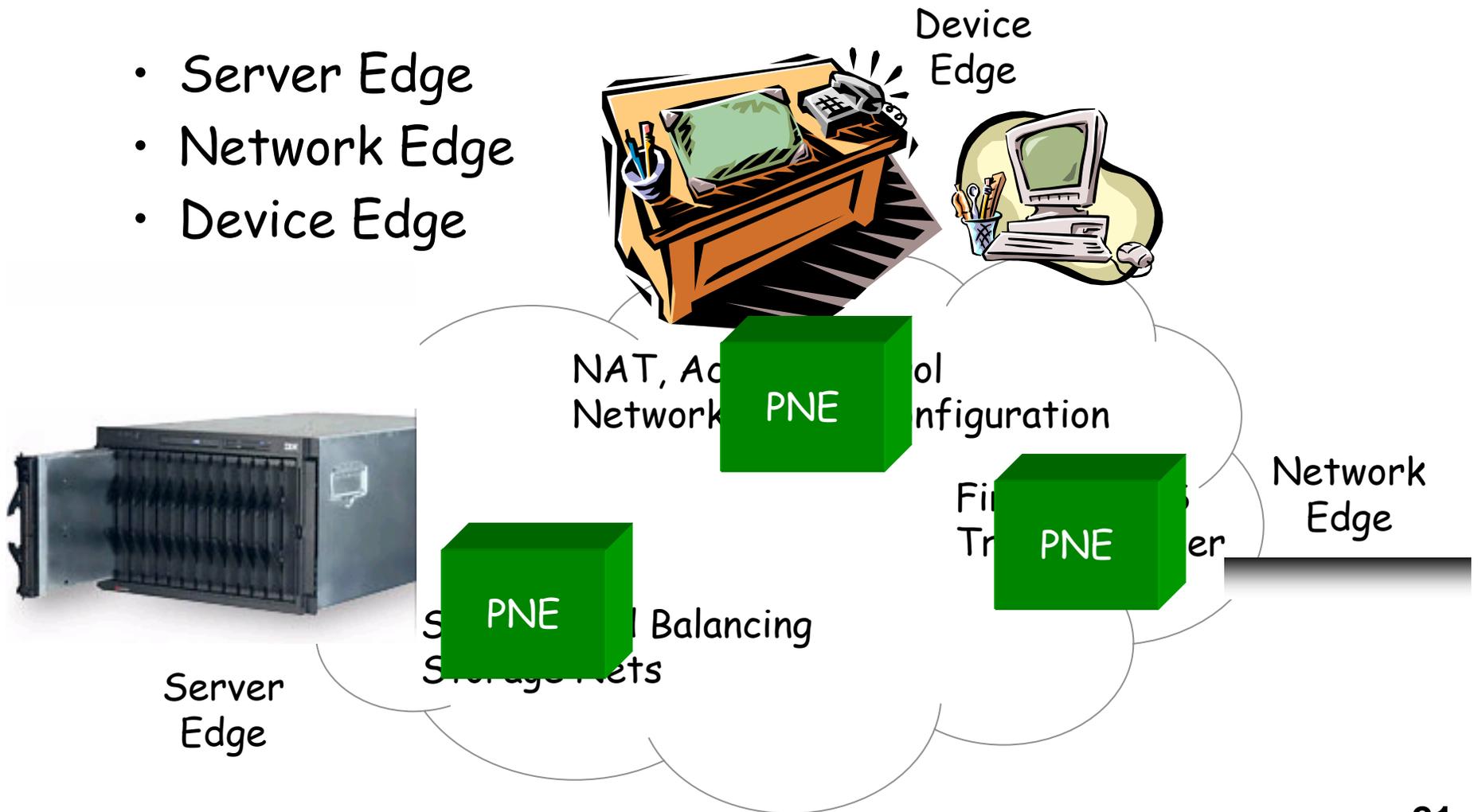


"The Computer is the Network"

- Emergence of Programmable Network Elements
 - First Gen Network Appliances, Directors
 - Storage Virtualizers, Intrusion Detectors, Traffic Shapers, Server Load Balancers, MIE accountants
 - Next Gen: Third Party Programmable beyond rules
- Generalized PNE programming and control model
 - Generalized "virtual machine" model for this class of devices
 - Retargetable for different underlying implementations
- Apps of Interest
 - Network Services: L7 switching, firewalls, intrusion and infected machine detection, storage virtualization, network monitoring and management, etc.
 - Network storage, iSCSI support
 - Streaming media transcoding/adaptation
 - Billing, accounting, stream customization for Mobile Network Edge

Adaptive Edge Networks

- Server Edge
- Network Edge
- Device Edge



Proliferation of Network Appliances



Packeteer PacketShaper
Traffic monitor and shaper



Network Appliance NetCache
Localized content delivery platform



F5 Networks BIG-IP LoadBalancer
Web server load balancer



Ingrian i225
SSL offload appliance



Cisco SN 5420
IP-SAN storage gateway



Nortel Alteon Switched Firewall
CheckPoint firewall and L7 switch



NetScreen 500
Firewall and VPN



Extreme Networks SummitPx1
L2-L7 application switch



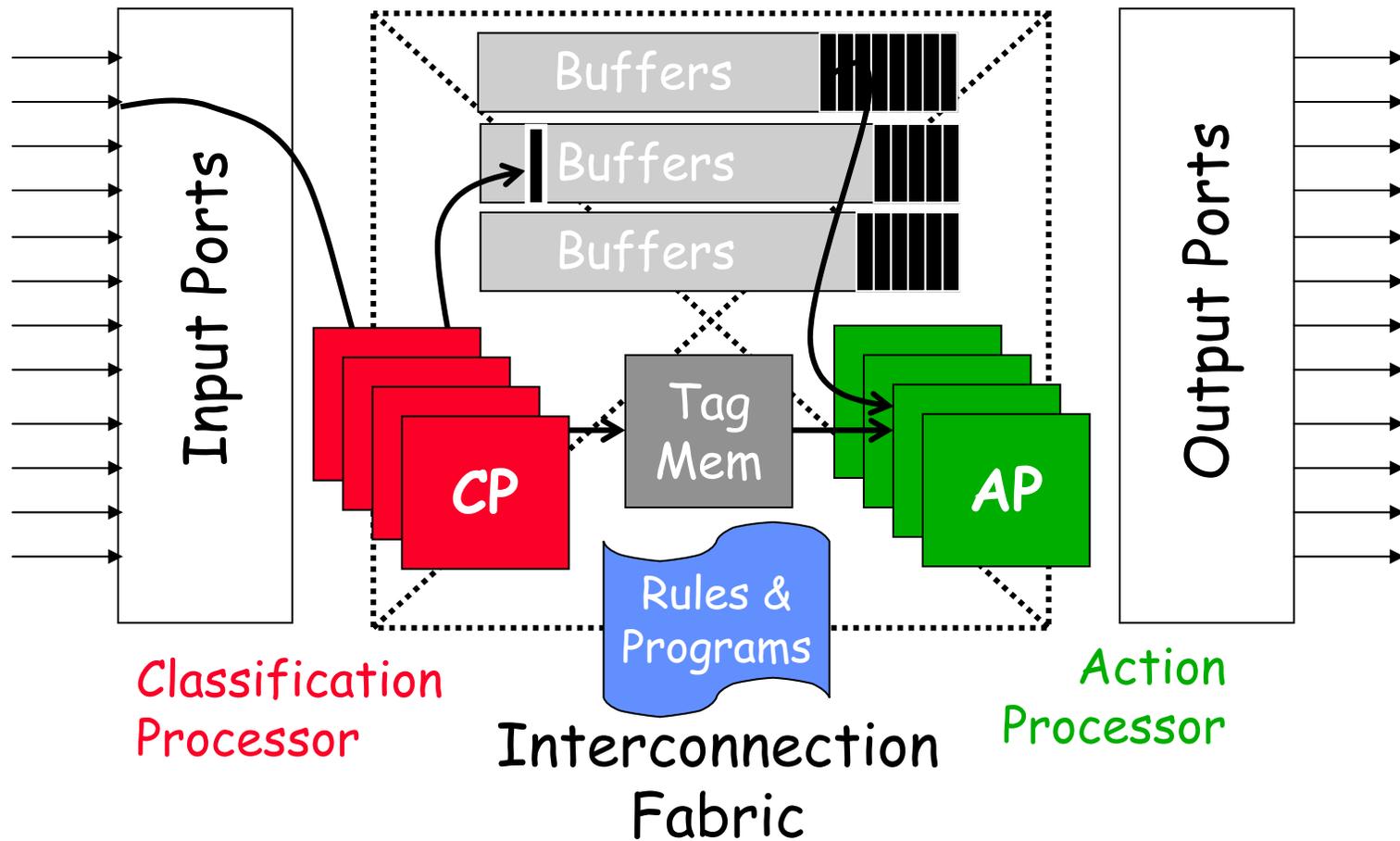
Cisco IDS 4250-XL
Intrusion detection system

In-the-Network Processing: the Computer IS THE Network

OASIS Vision

- Common programming/control environment for diverse network elements to realize full power of "inside the network" services and applications
- Via VM architecture for PNEs, retargeted for diverse appliance-specific architectures
- Focus on stream extraction, intrusion detection, network monitoring, iSCSI processing and performance enhancement
- Open framework for multi-platform appliances, enabling third party service development

Generic PNE Architecture



RouterVM

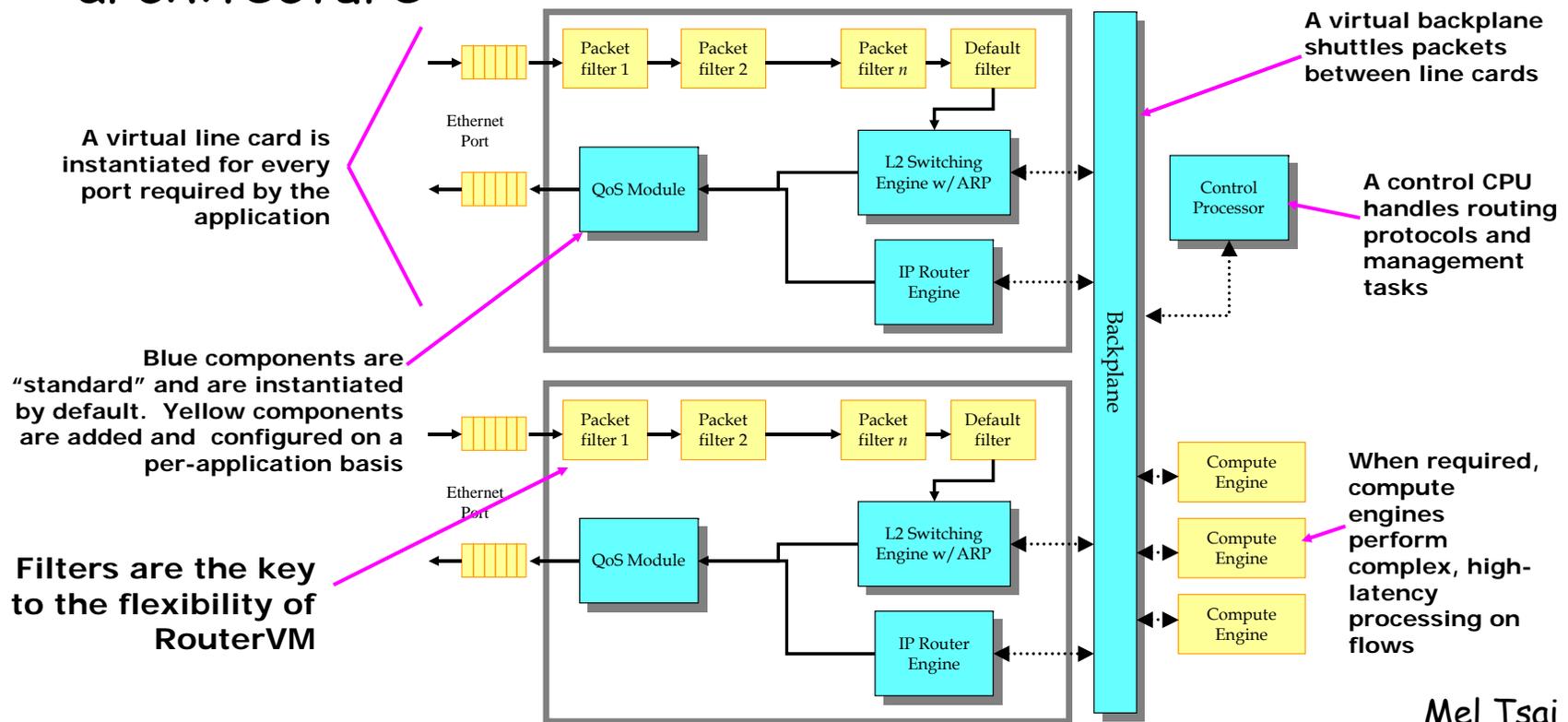
- High-level environment for writing multiple, coexisting network apps targeted for a single programmable router
- Develop powerful apps quickly, without new code
- New and existing apps written by network admins, not app engineers
- Management of apps and standard routing functions intuitive; environment maximizes flexibility and minimize configuration errors through detection and learning
- Network apps hardware-independent, while still effectively utilizing unique hardware of target device

RouterVM

- Flexible, high-level environment for developing and testing network applications
- Virtualized: consistent view of underlying hardware resources of any target
 - Apps portable across diverse architectures
 - Apps easily simulated before deployment
- Apps, policies, and standard routing functions managed thru a CLI
 - RouterVM implements *generalized packet filter*, allows new apps, policies to be implemented/configured thru CLI
 - Many types of new apps implemented without new code

Virtual Machine Architecture

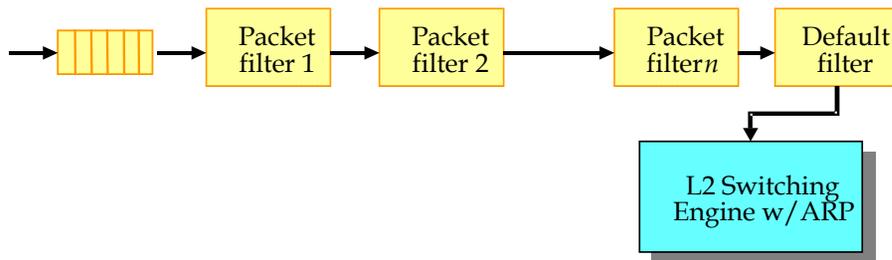
- Virtualized components representative of a "common" router implementation
- VM structure does not depend on particular hw architecture



Generalized Packet Filters

- Key to approach's flexibility

- Extends concept of "filters" normally found in routers
- Small number of GPFs used as building blocks for large number of apps
 - » DB of GPFs precludes writing of new code
- Supports flexible classification, computation, and actions
- GPFs executed in numeric order



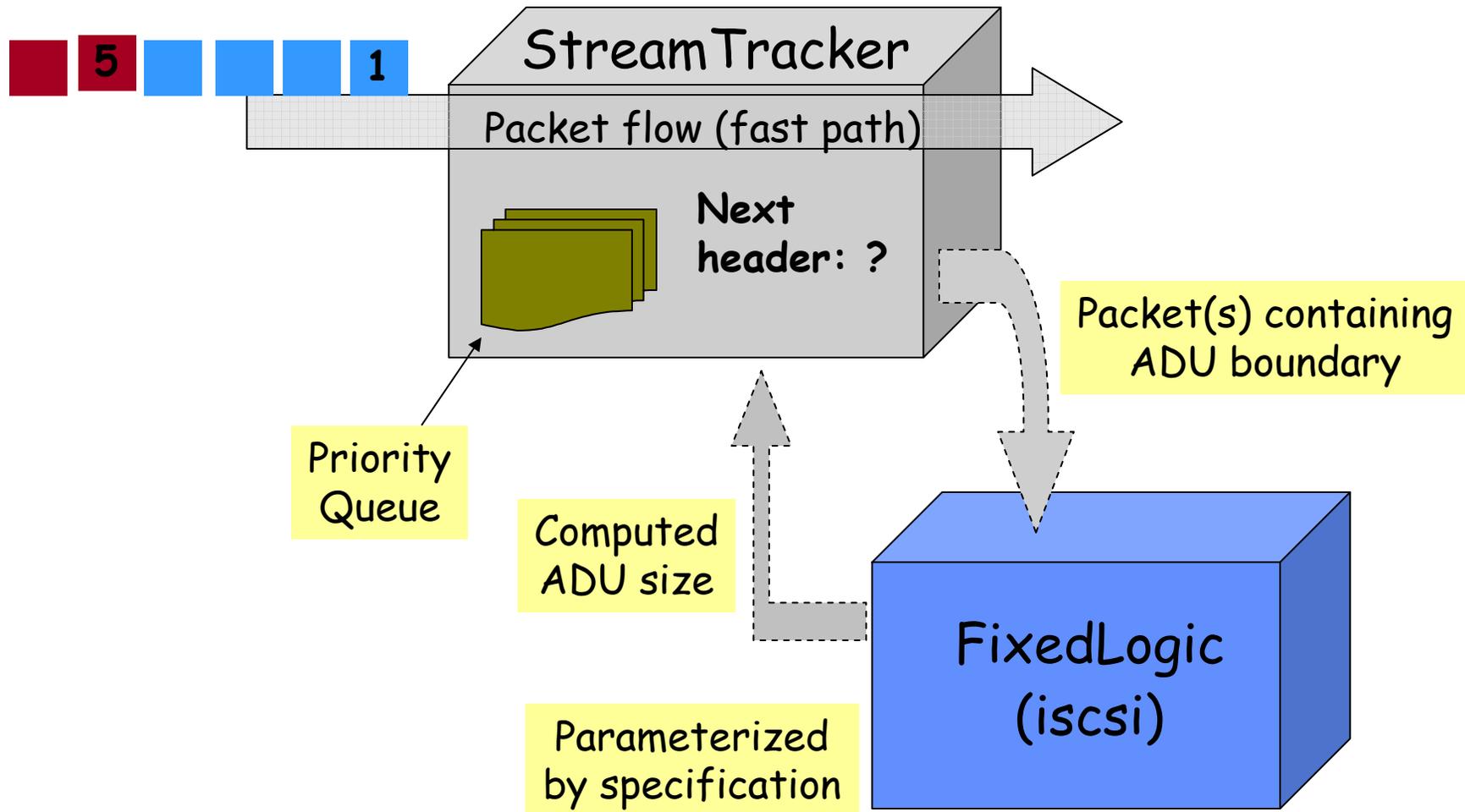
- Examples:

- Traffic shaping and monitoring
- L7 traffic detection (Kazaa, HTTP, AIM, POP3, etc.)
- QoS and packet scheduling
- NAT
- Intrusion detection
- Protocol conversion (e.g. IPv6)
- Content caching
- Load balancing
- Router/server health monitoring
- Storage
- Fibre Channel to IP
- iSCSI
- XML preprocessing
- TCP offload (TOE)
- Mobile host management, 802.11
- Encryption/compression. VPNs
- Multicast, Overlays, DHTs

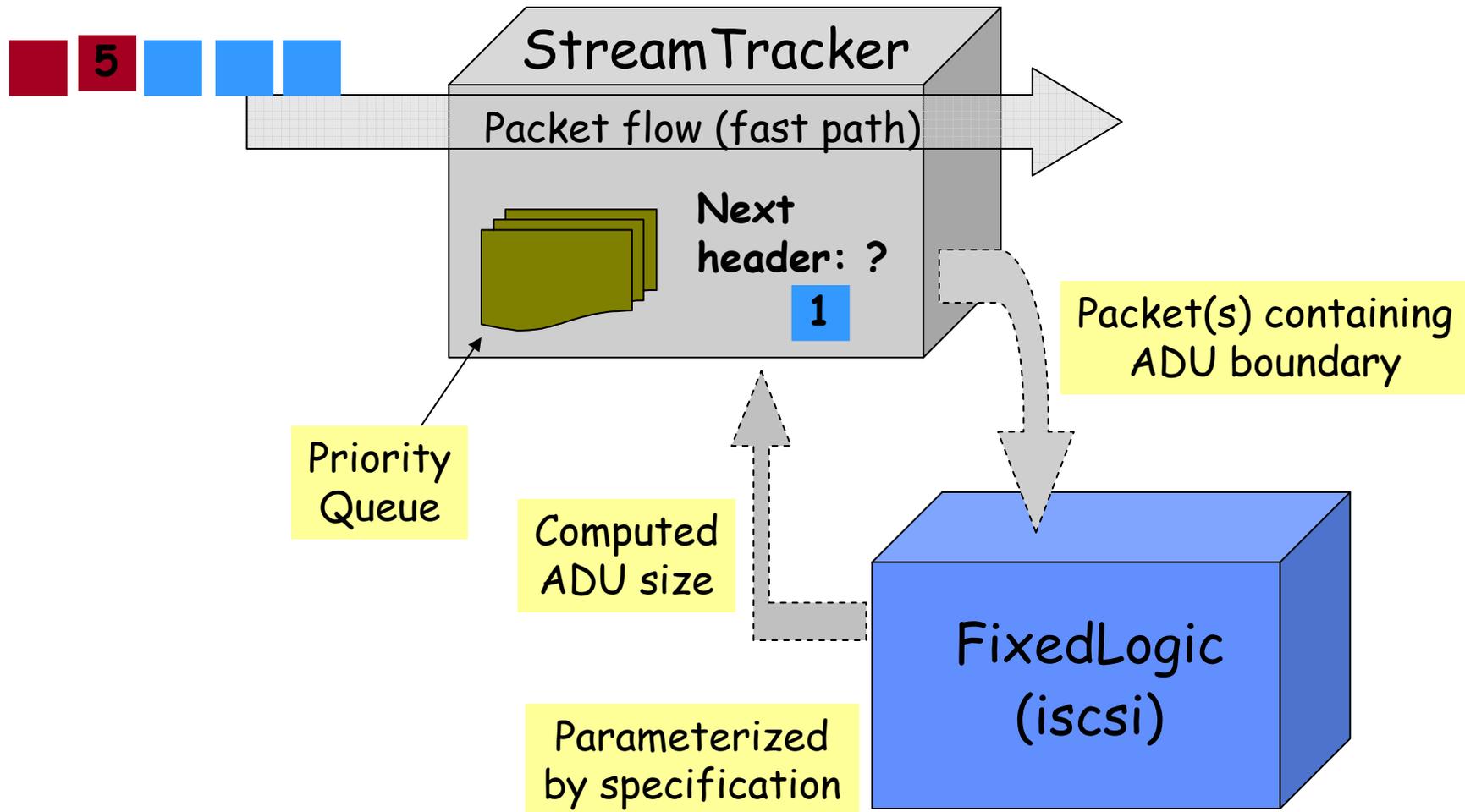
Stream Processing

- Streams prevalent in the edge network
 - Storage: iSCSI; storage virtualization, load-balancing, security, caching, in-network optimizations
 - Web/P2P: HTTP--pipelining, load-balancing
P2P--HTTP used as transport
 - Measurement/Monitoring: tracking state of observed protocols over time
- Need general mechanism for stream processing (L7) integrated with PNE design and architecture
 - Extension to PNEs to specify streams
 - General mechanism for executing spec at high speeds
- Provides clear separation between packet transport and protocol structure

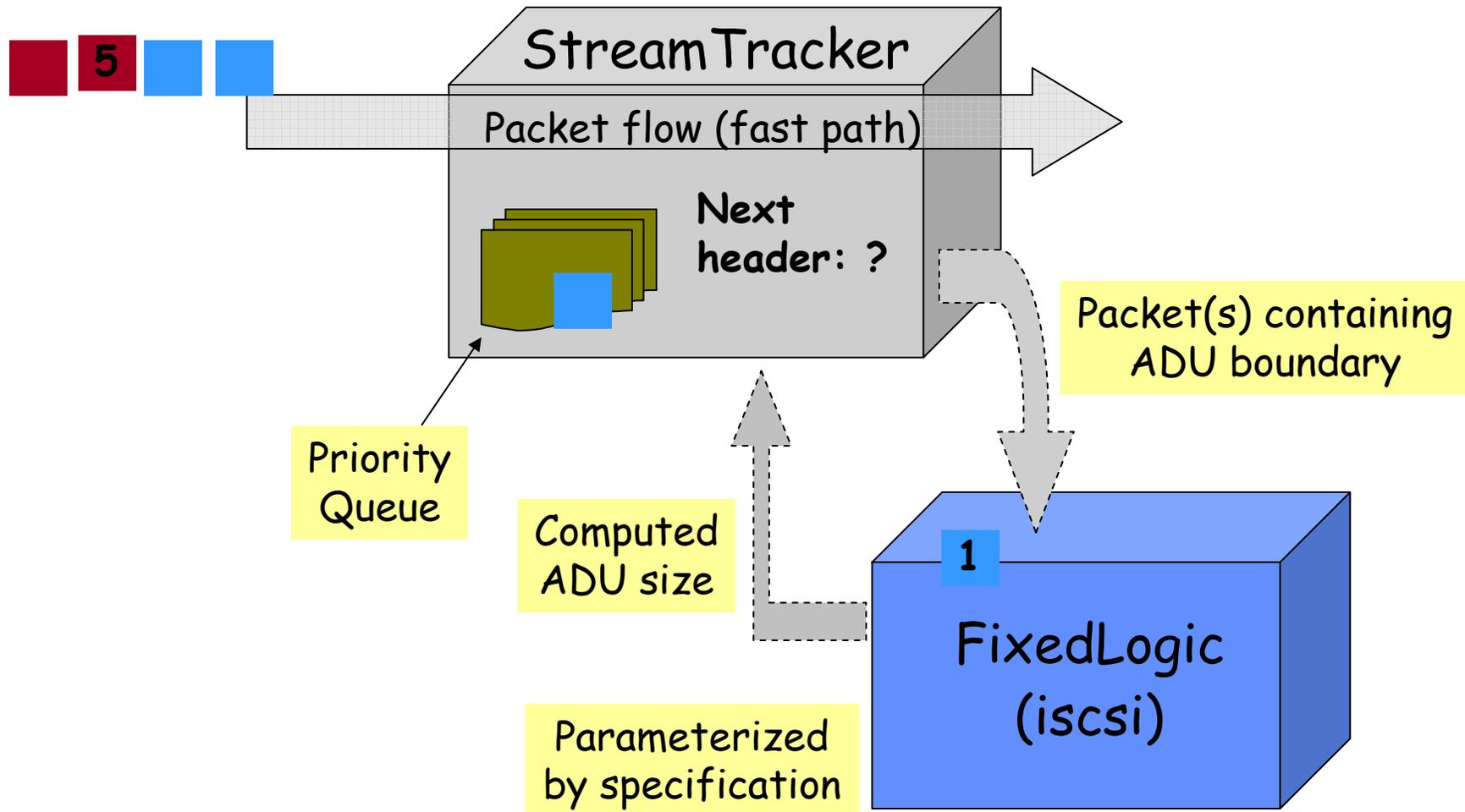
Mechanism for Tracking ADUs



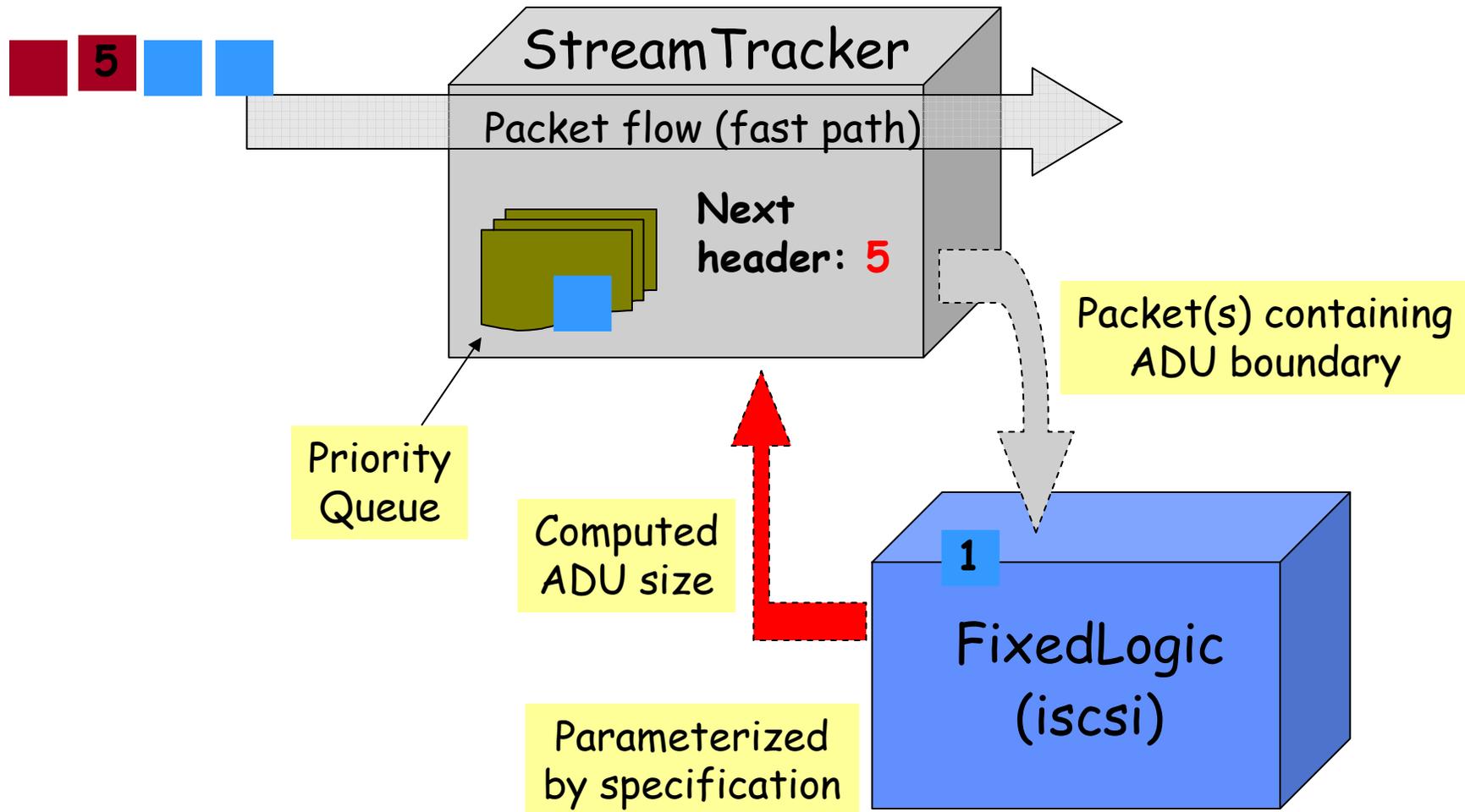
Mechanism for Tracking ADUs



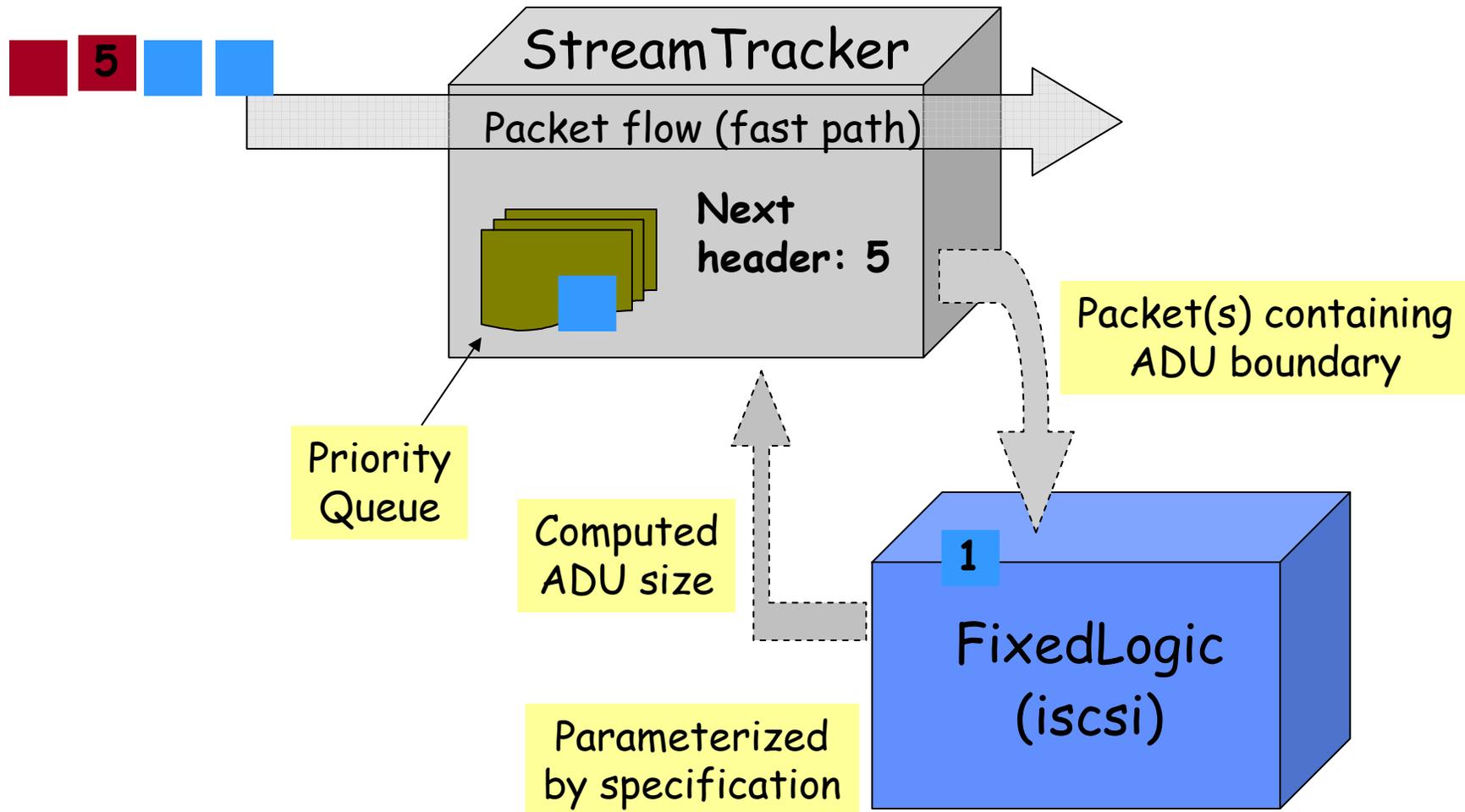
Mechanism for Tracking ADUs



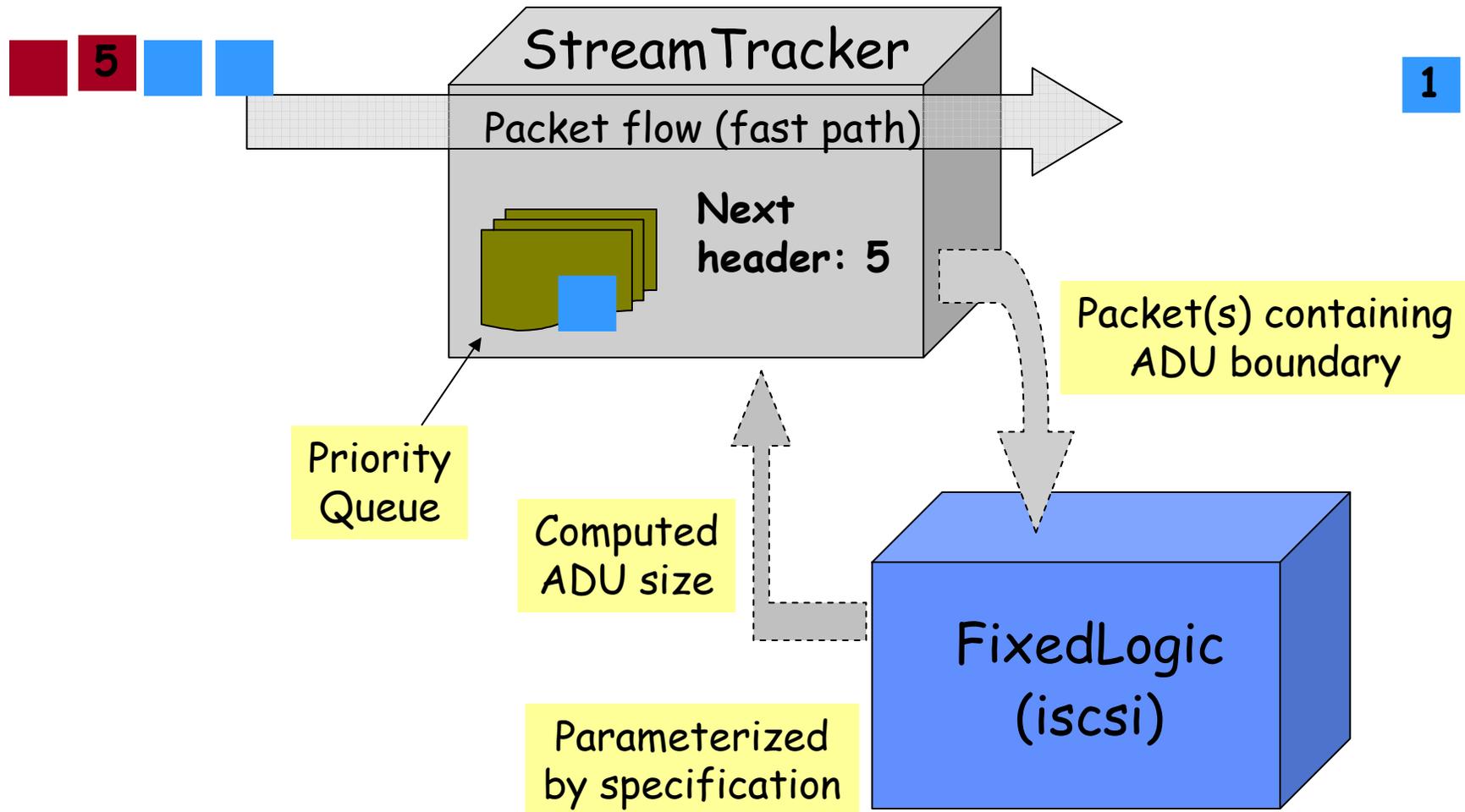
Mechanism for Tracking ADUs



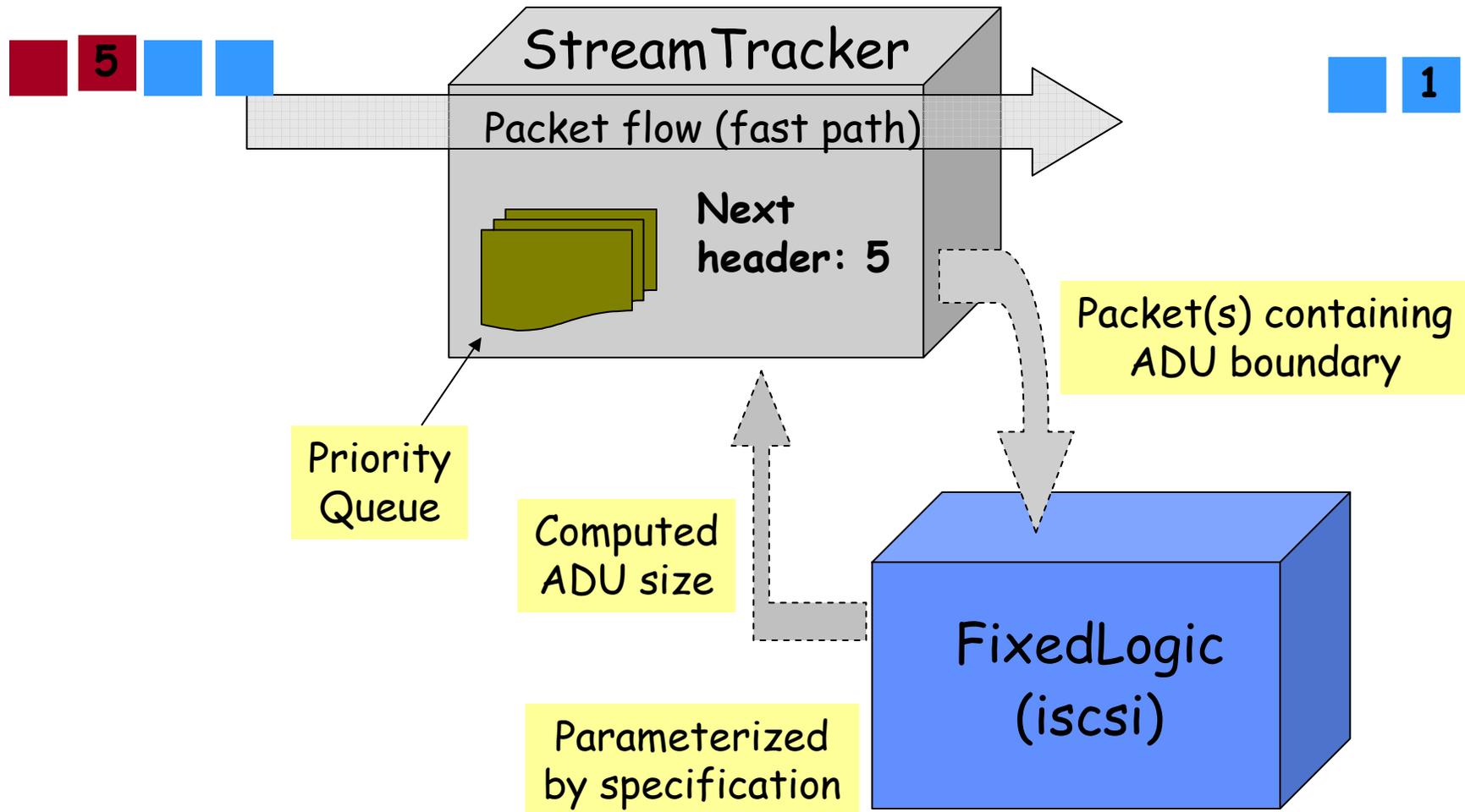
Mechanism for Tracking ADUs



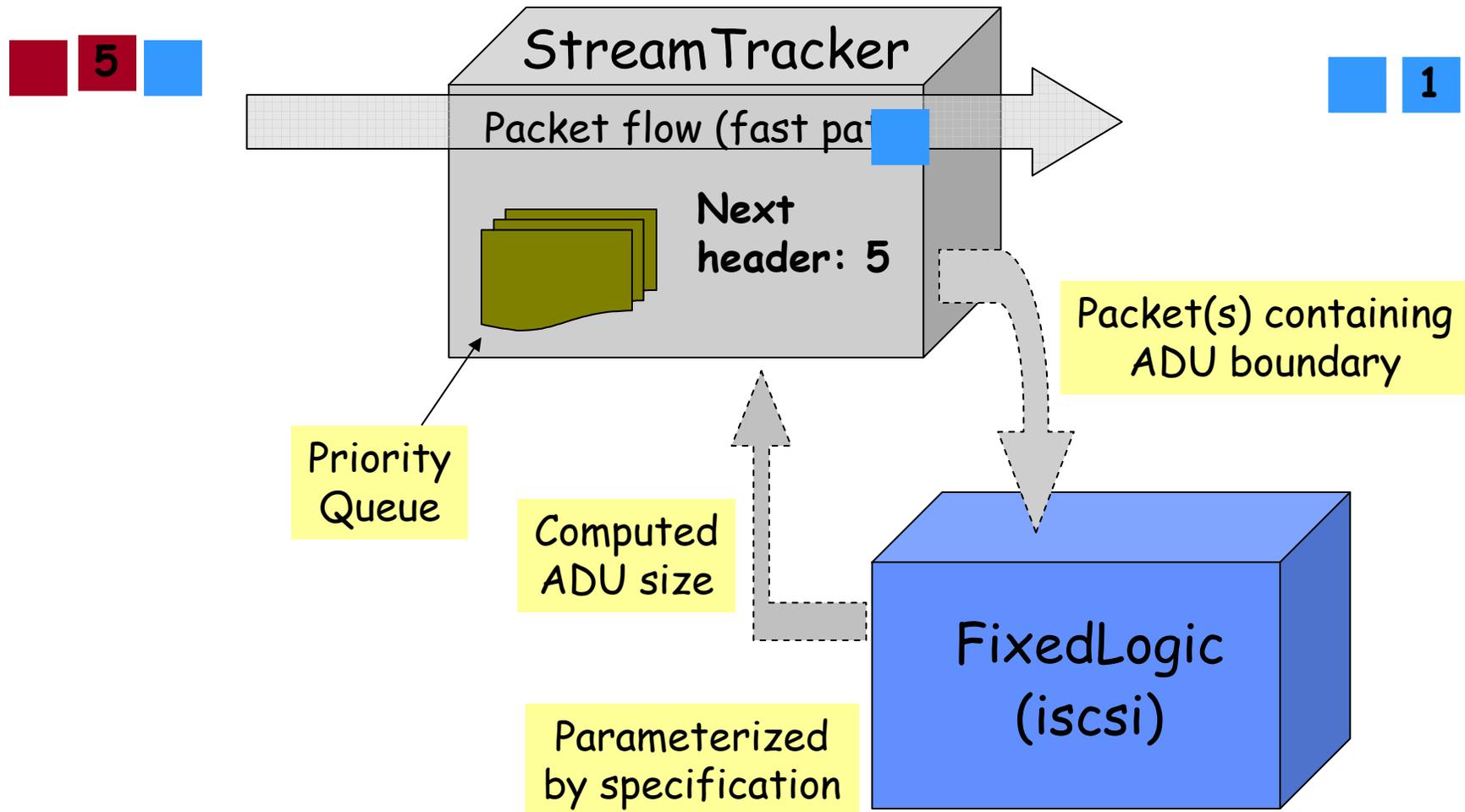
Mechanism for Tracking ADUs



Mechanism for Tracking ADUs

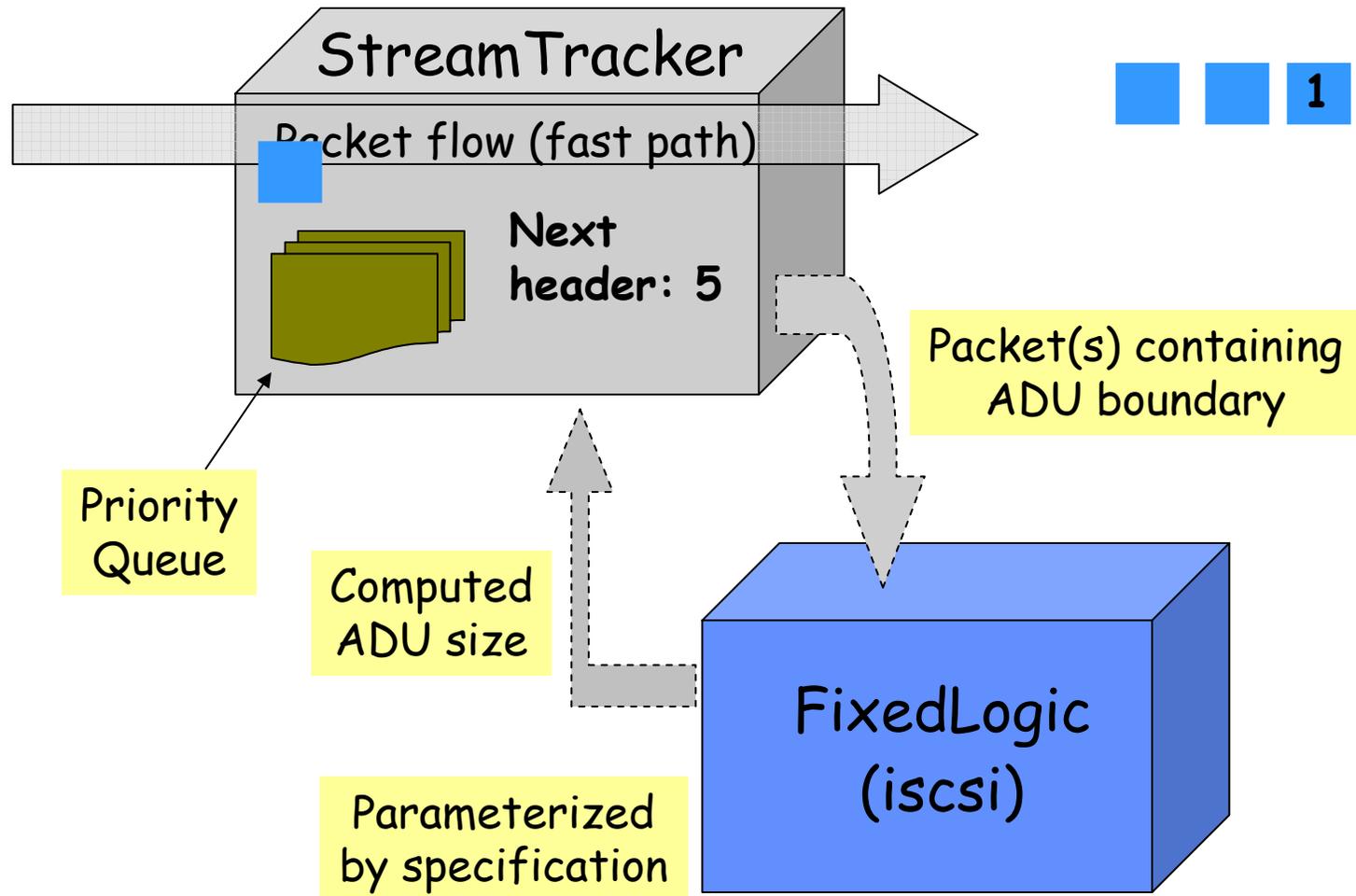


Mechanism for Tracking ADUs



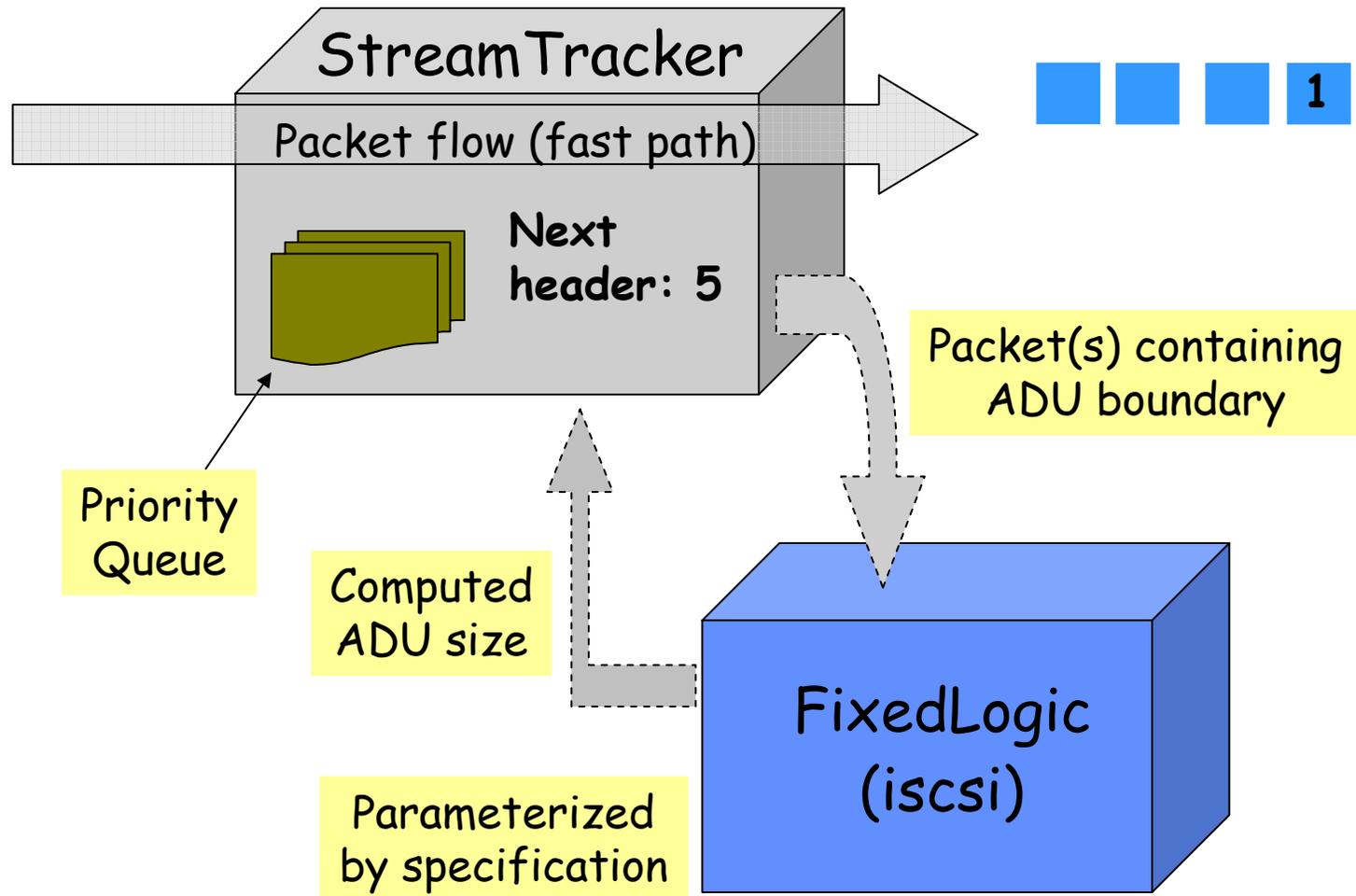
Mechanism for Tracking ADUs

■ 5

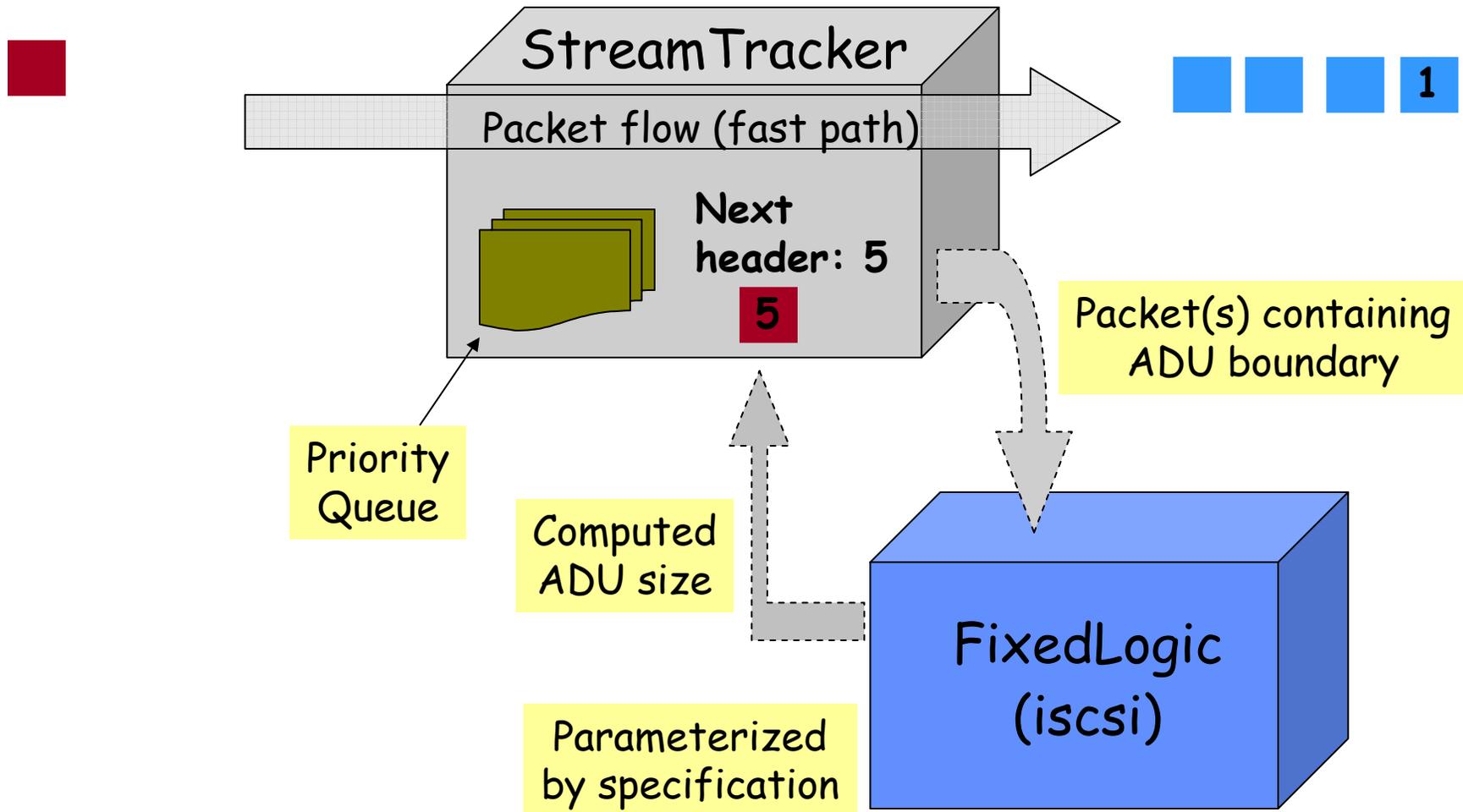


Mechanism for Tracking ADUs

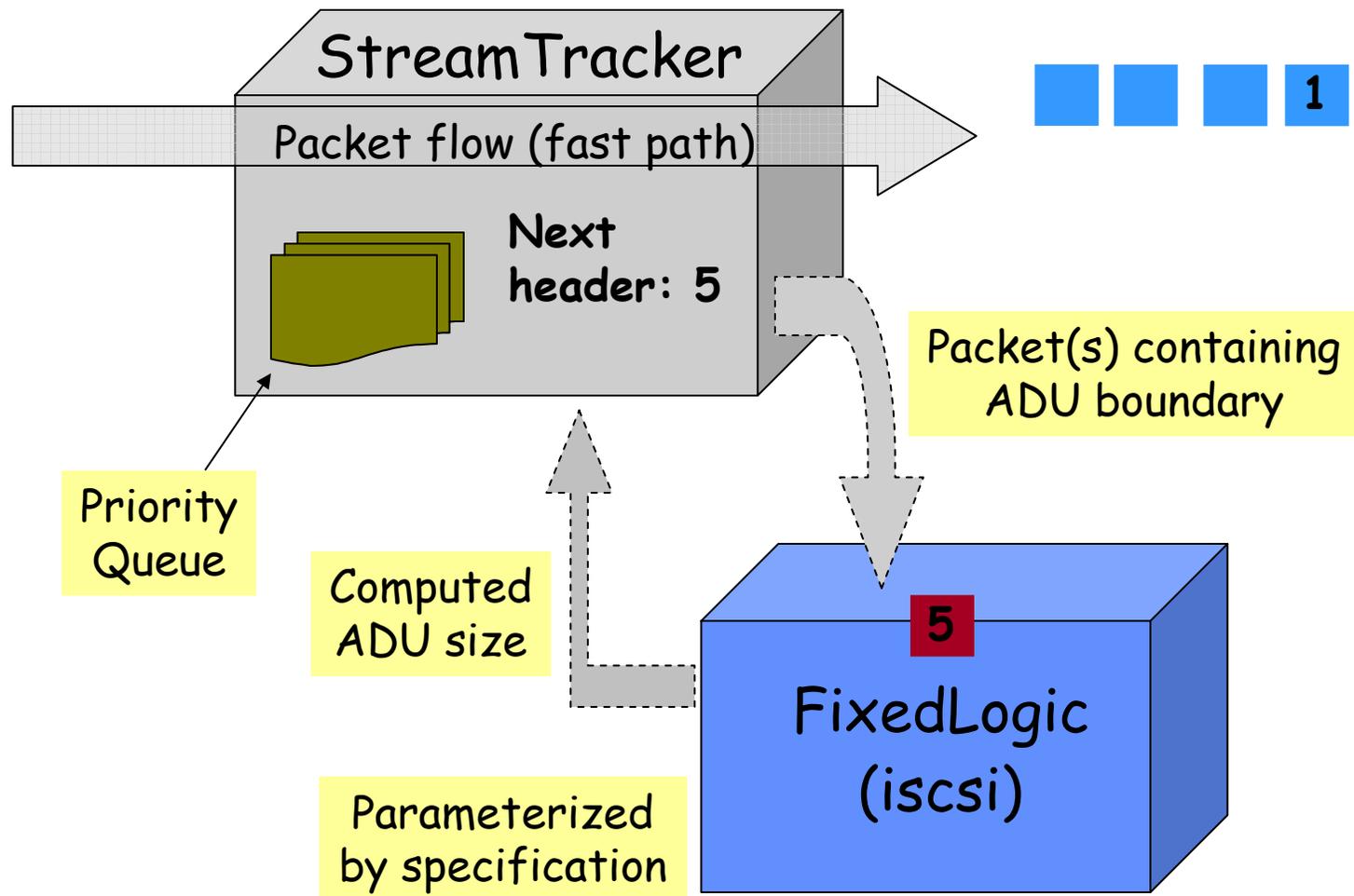
5



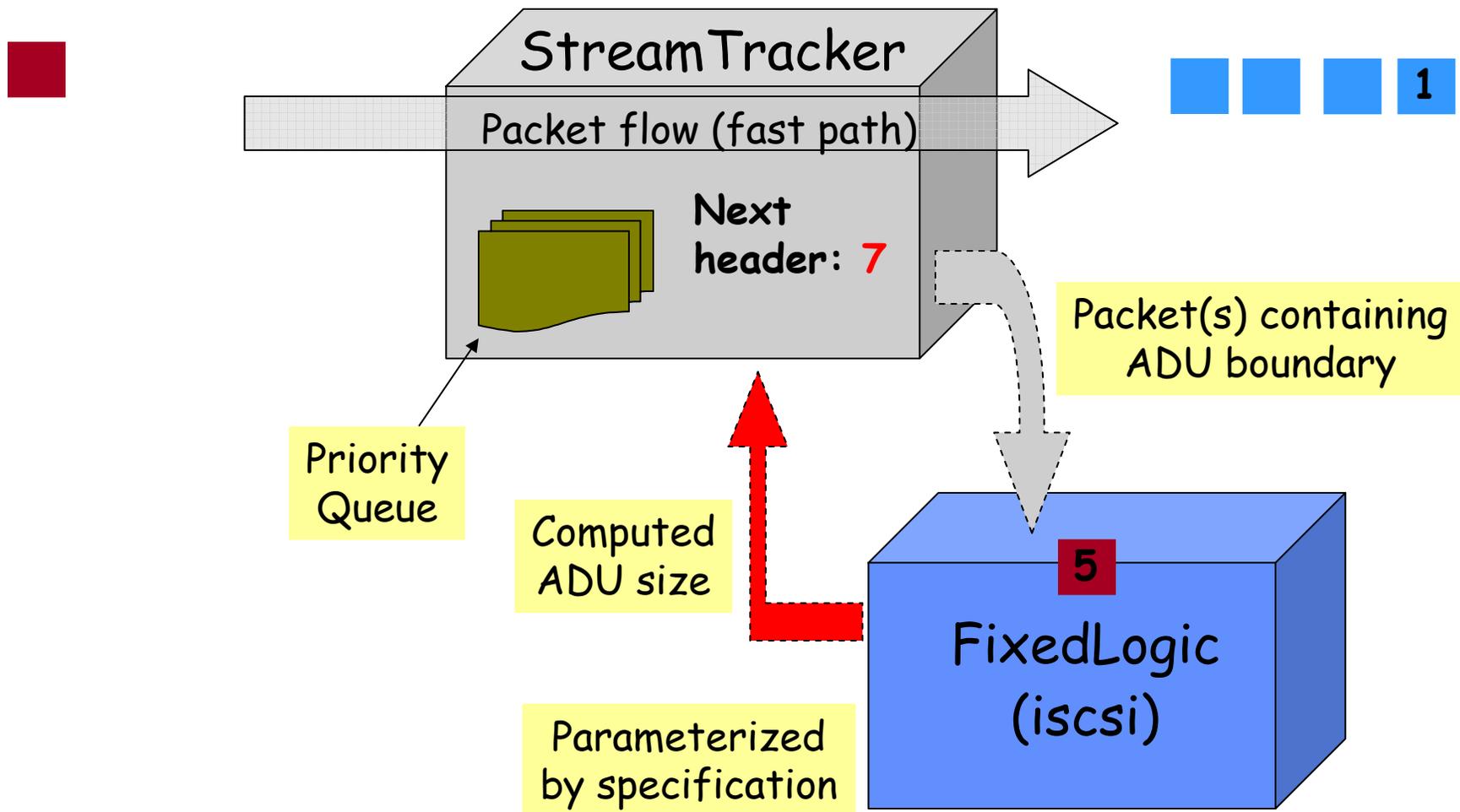
Mechanism for Tracking ADUs



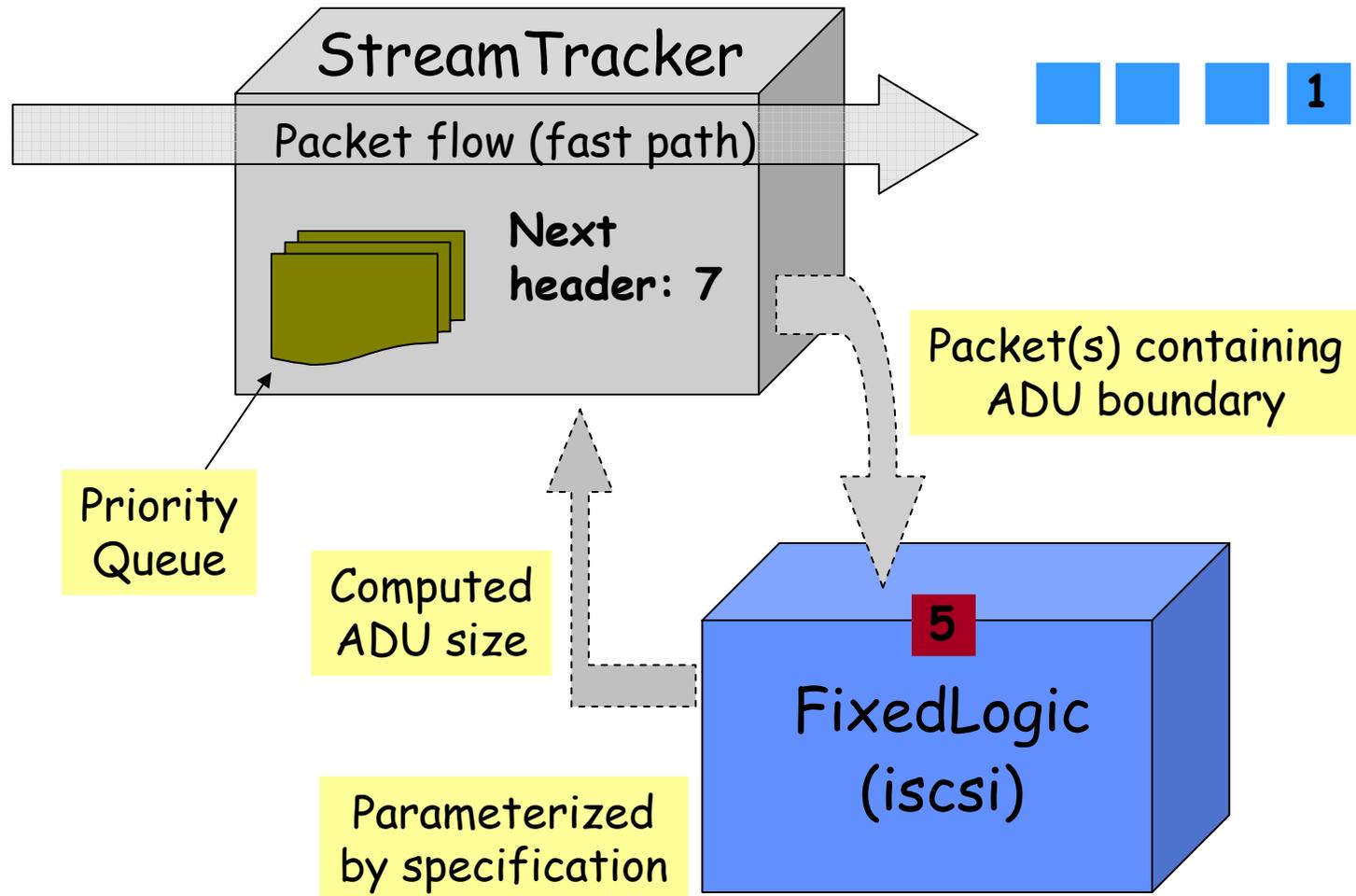
Mechanism for Tracking ADUs



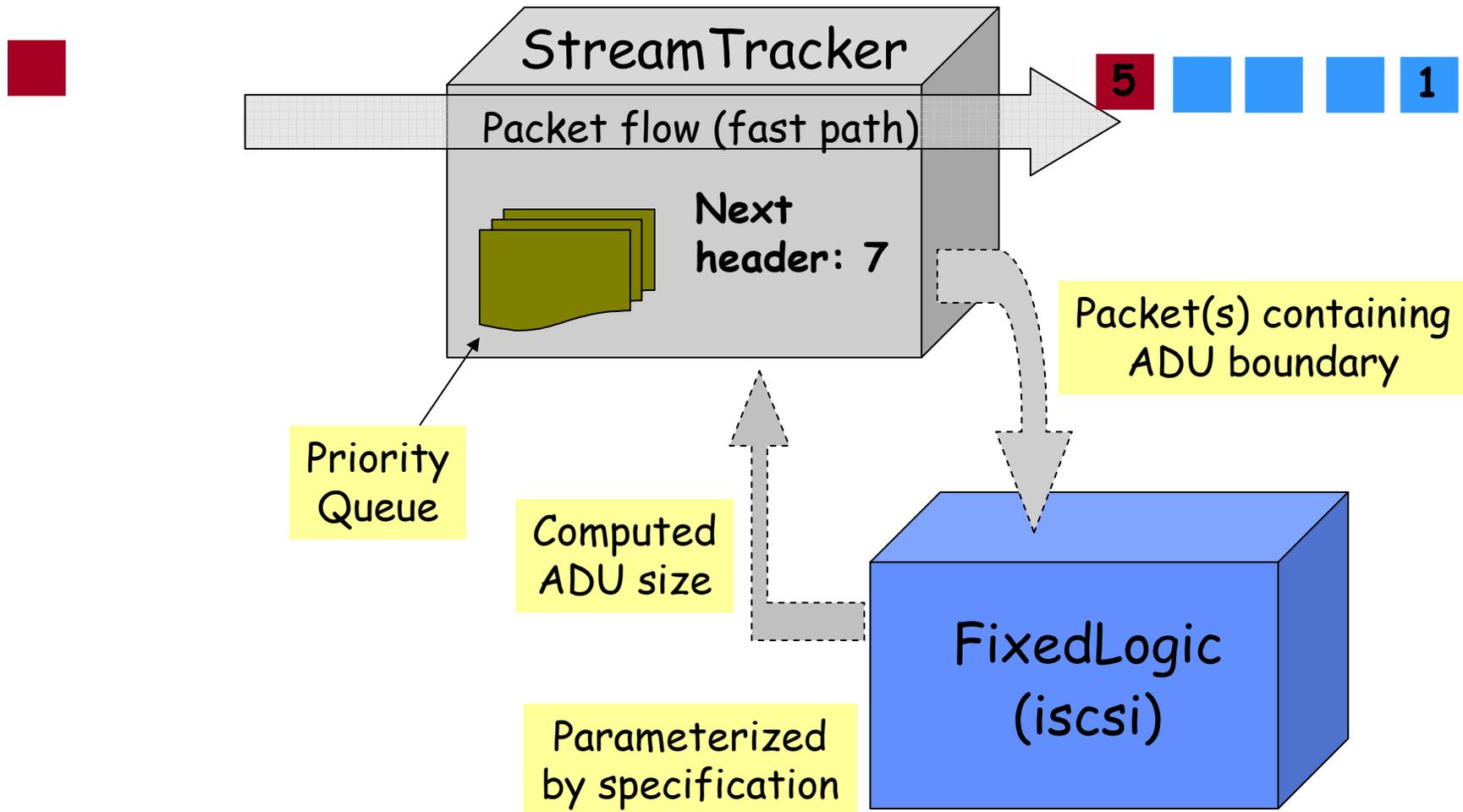
Mechanism for Tracking ADUs



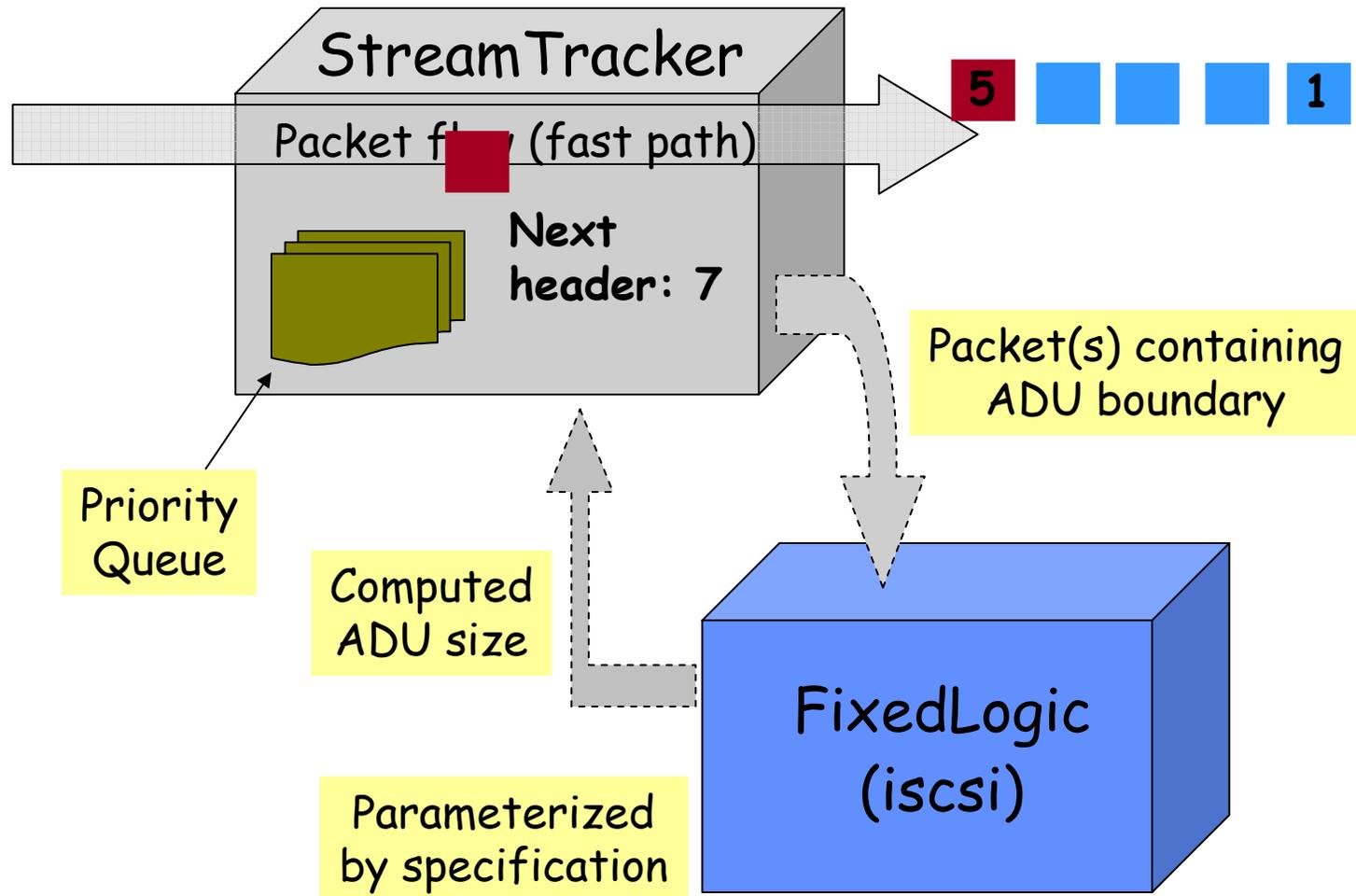
Mechanism for Tracking ADUs



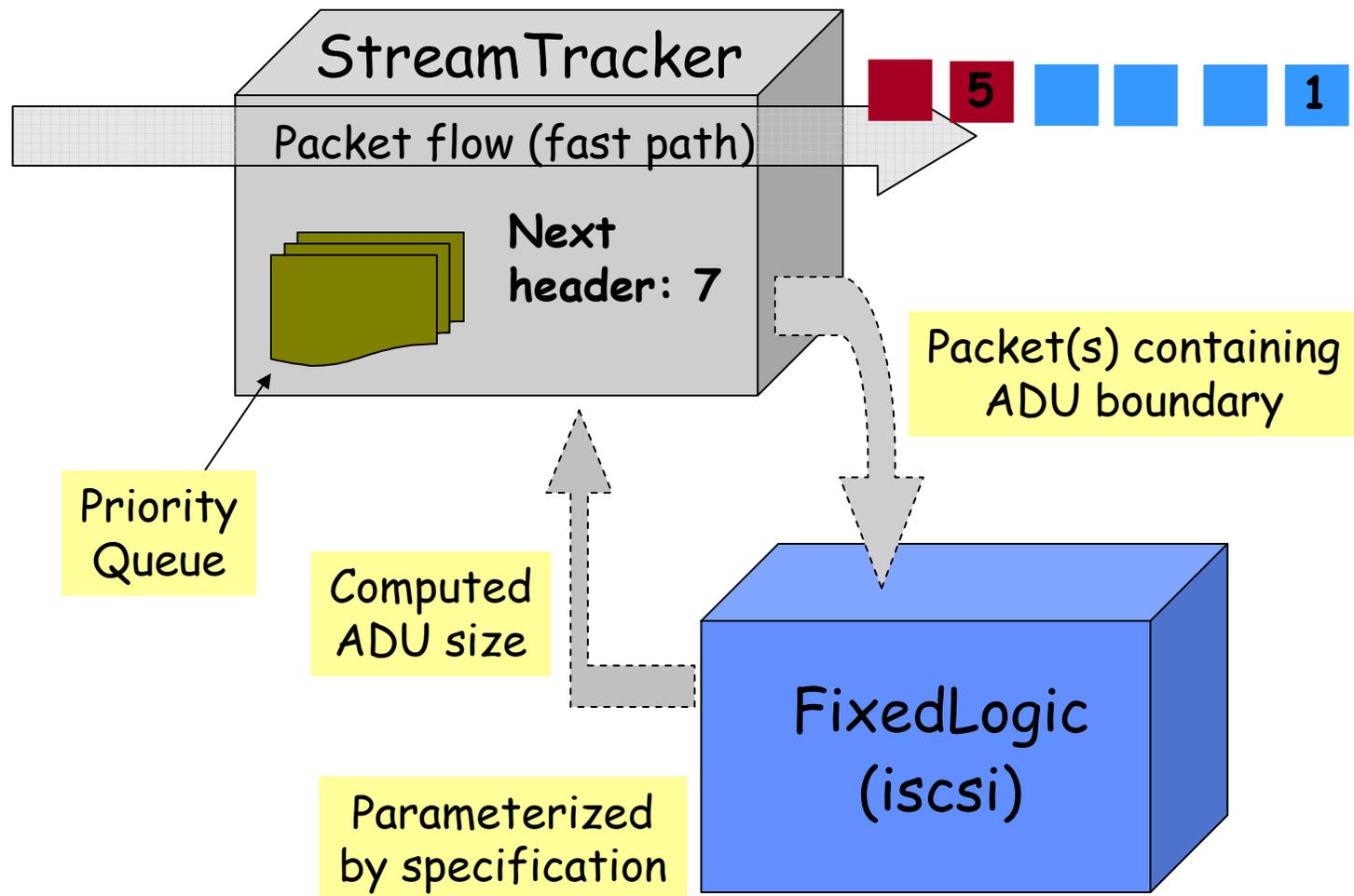
Mechanism for Tracking ADUs



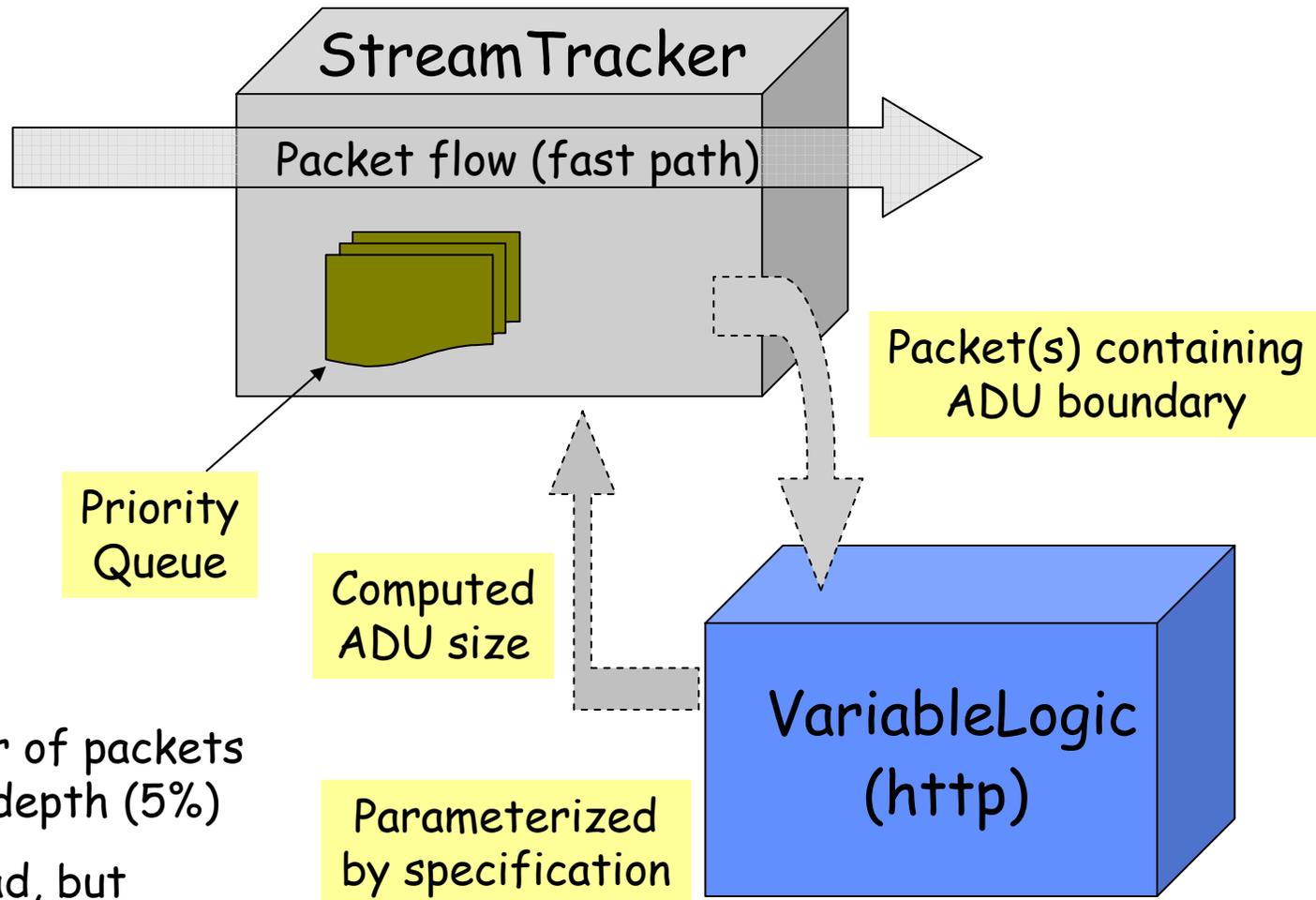
Mechanism for Tracking ADUs



Mechanism for Tracking ADUs



Mechanism for Tracking ADUs



Small number of packets
examined in depth (5%)

High overhead, but
infrequent (30%)

OASIS Testbed



- Current PNE Testbed
 - Emerging architecture: PNEs plus PCs on high speed interconnect in a single rack
 - Level 7 Switches
 - » Filtering limited to HTTP
 - » Back door processing as in stream extraction experiments
 - » Next generation significantly more third party programmable
 - Enterprise Class Routers
 - Fast blade PCs running Linux
- Intended target not limited to this hardware

Presentation Outline

- Technology Trends and Implications
- SAHARA Service Architecture
- OASIS Programmable Network Elements
- **RADS Distributed Architecture**
- Summary and Conclusions

Reliable Adaptive Distributed Systems

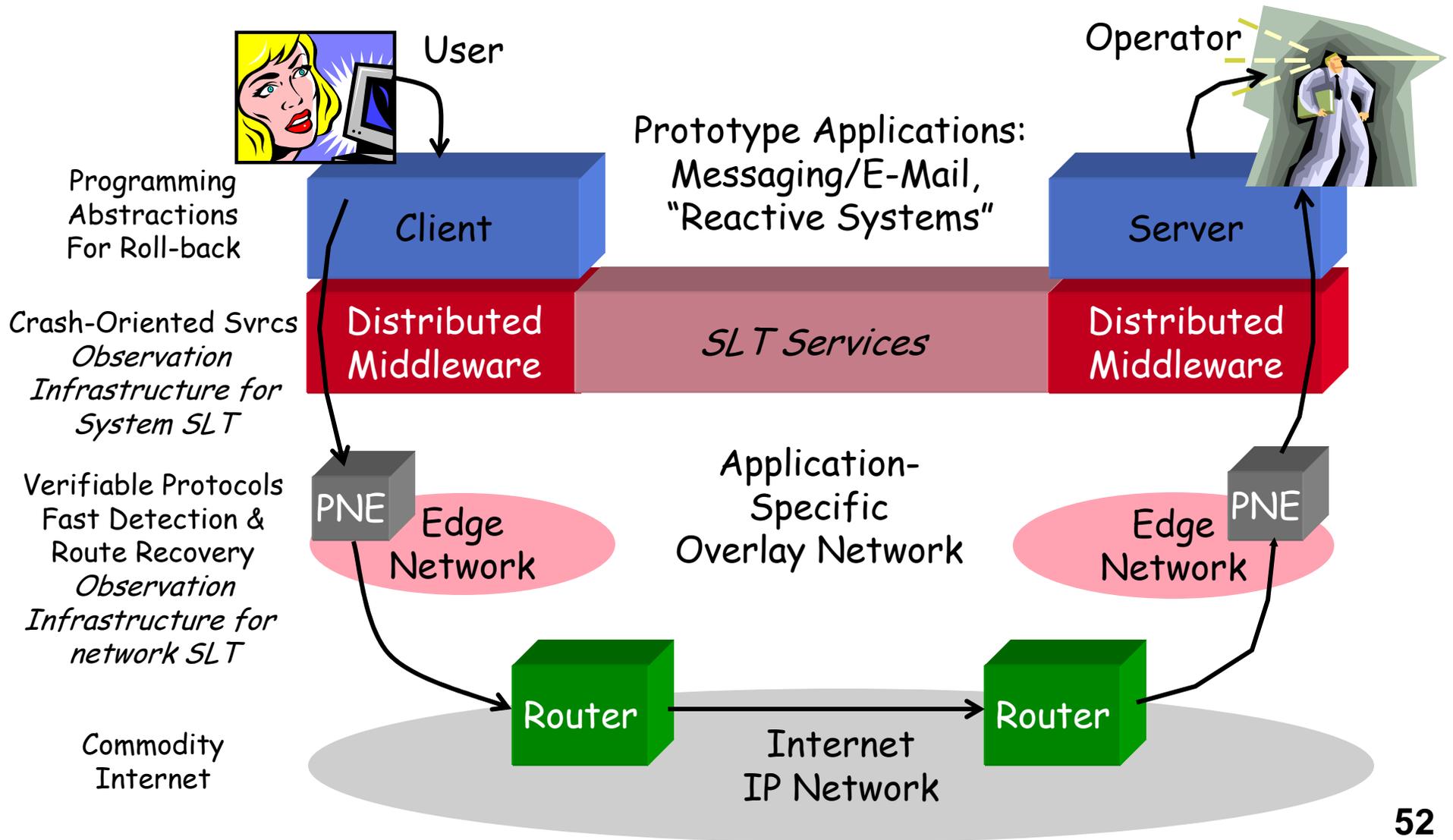
Dramatically improve the trustworthiness of networked systems

- **Observe**: design observation points throughout system
- **Analyze**: infer via statistical learning
 - Respond: detect anomalous behavior vs. baseline
 - Learn: use observations to modify responses to future observations
- **Act**:
 - Reactive: use control points in system for rapid recovery if detect something wrong
 - Proactive/protective: prophylactically act on system to prevent predicted impending failure

Brittle Distributed Systems

- Fragile, easily broken, poor dependability and security
 - E.g., Amazon: yearly revenue \$3.1B, downtime costs \$600,000/hr
- Design for rapid detection, diagnosis, recovery
 - Rapid application and server recovery, agile network rerouting, proactive protective actions ...
 - No distinction between “normal operation” and “recovery”
- Elements of our solution
 - Programming paradigms for robust recovery
 - Crash-only software design for rapid server recovery
 - Network protocols designed for observation to allow rapid detection of behavioral violations
 - Instrumentation and SLT for on-line analysis, anomaly detection, diagnosis of failure
- Adaptation benchmarks to measure progress
 - What you can't measure, you can't improve
 - Collect real failure data to drive benchmarks

Reliable Adaptive Distributed Systems



Presentation Outline

- Technology Trends and Implications
- SAHARA Service Architecture
- OASIS Programmable Network Elements
- RADS Distributed Architecture
- **Summary and Conclusions**

SAHARA

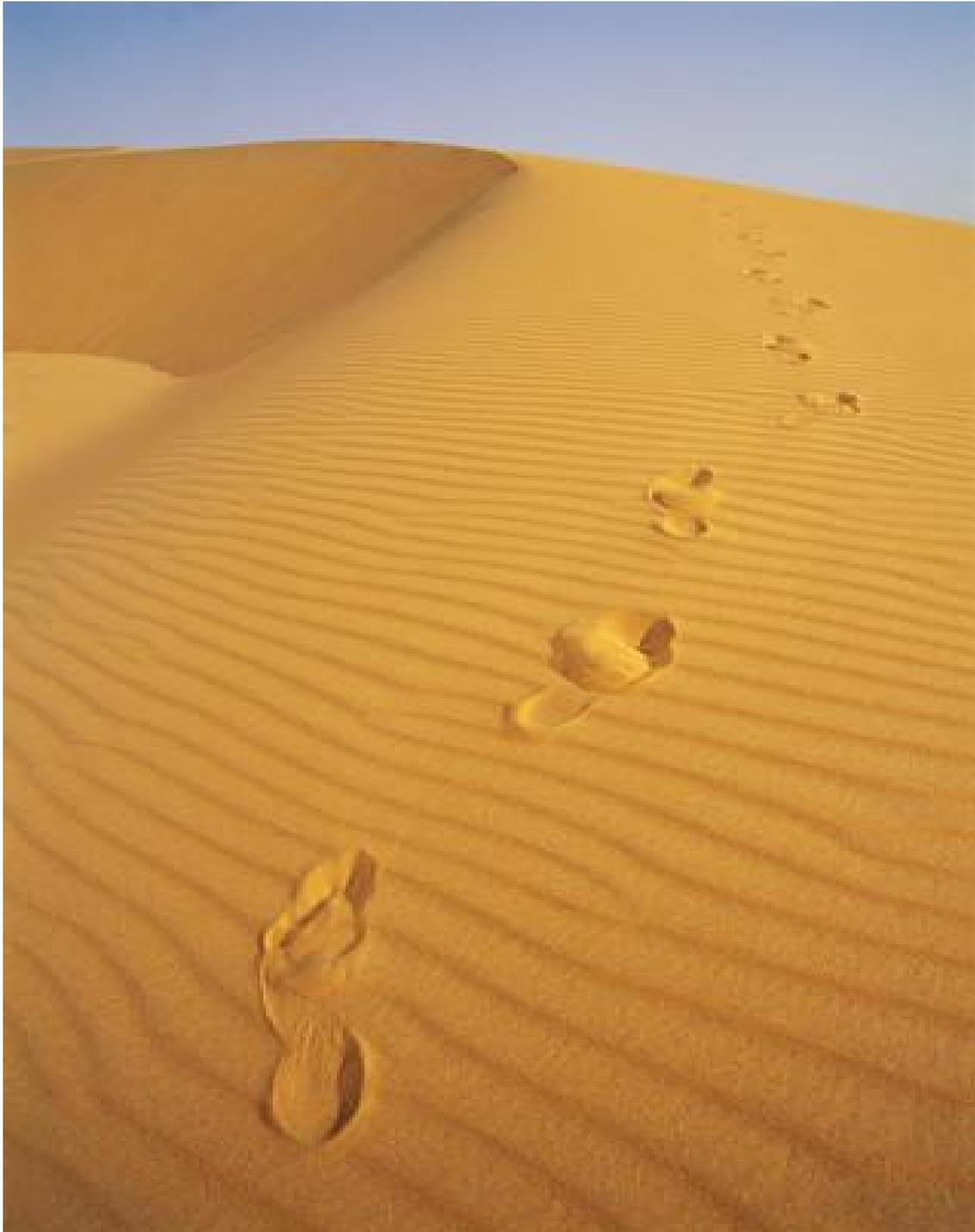
- Evolve Internet architecture to better support multi-network/multi-service provider model
 - Dynamic environment, many service providers & service instances
 - Achieve desirable properties across multiple, potentially distrusting (Internet) service providers
- Routing as a composed service
 - Trust: BGP Verification/Detection + Containment
 - Dependability: Localizing Routing Faults

OASIS

- "Active Networking" is real
 - Marriage of networking and processing in switched and routed infrastructures
 - Migration of services into PNEs is next leap in network service architecture
- Unifying framework lacking
 - Abstract programming model and supporting development tools
 - Focus on streaming video and storage
- Pervasive infrastructure for observation and action at the network level

RADS

- Network-based Enterprises
 - Bottleneck is reliability, not performance
 - Making distributed applications more dependable: reliability and security
 - Spanning Apps-, Middleware-, and Network-level
- Algorithm design and system implementation
 - Distributed architecture embedding SLT as a primitive building block
 - Embedding observational and inference means at strategic points
 - New on-line/real-time statistical inference/verification techniques
- Scientific Foundation for "Self-*" Systems
 - New design principles and tools for systems that continuously adjust their behavior in response to analysis of online observations
 - New metrics/benchmarks for evaluating self-adapting networked systems
 - Advances in SLT moving from off-line to on-line analysis



The Computer is the
Network:
The Emergence of
Programmable
Network Elements

Randy H. Katz

Thank You!