Supporting Enterprise Applications with Attached Network Processors

Karsten Schwan, Ada Gavrilovska,
Greg Eisenhauer

GT Network Processors Group

- www.cercs.gatech.edu/projects/npg
- Compilers
  - Santosh Pande (CoC), ...
- Intrusion Detection
  - Wenke Lee (CoC), David Schimmel (ECE), ...
- Application-level Services
  - Karsten Schwan, Ada Gavrilovska, ...
  - focus on high-performance scientific and enterprise applications

Motivation

- Large-scale high-performance distributed application need dynamic and customizable services
  - better resource utilization, quality of service, runtime operating condition, changes in application needs...
  - extend core functionality and enable customizations that deal with more than just network-level information
  - need ability to dynamically deploy application-specific processing actions on data and modify data path through distributed system

Service Requirements

- Some actions required are similar to network-centric services, but access payload
  - content based routing, filtering, replication, data transcoding, notification...
  - should be able to implement them efficiently on networking devices
- Other services are resource intensive
  - resources available at standard hosts needed to support them – software, memory, computational resources...
  - floating point arithmetic, matrix manipulation, DB access...
- Some service need to be executed as early as possible
  - intrusion detection, filtering, ill-formed messages...

Current Approaches

- overlays are built to enable customized services and data delivery
  - customizations occur at user- (or kernel-) level
  - cost of network stack traversal, computational and memory/ I/O loads at hosts

- active networking approach to customize core communication services
  - too restrictive for the general case
  - mostly suitable for network-centric services

- device-level research
  - specific domain: storage, web services, custom devices

Our Approach

- Use network processors (NPs) to enhance standard hosts, as Attached Network Processors (ANPs)
- Core functionality – delivery of application-level messages to destination in distributed system - on ANP
- Map other services or service components across host-ANP boundaries
- create host-ANP platforms and use joint resources, offload hosts, and benefit from NP's specialized hardware for certain functionality

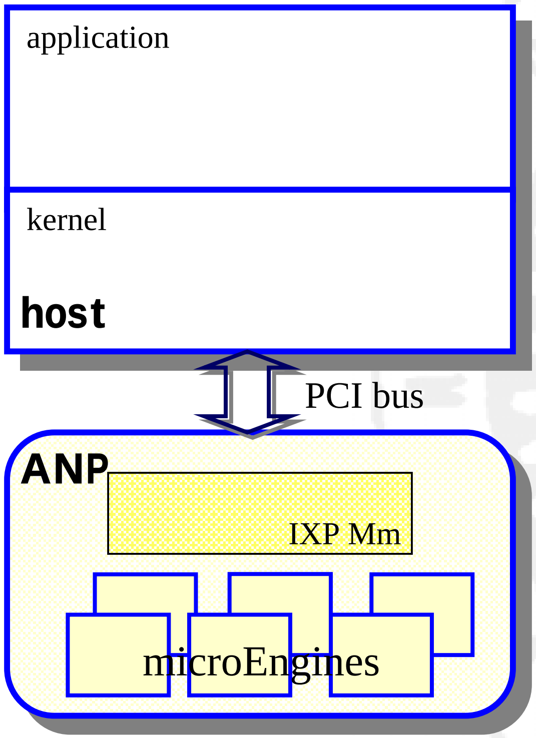
why programmable NP?

- optimized hardware with built in support for networking functionality, efficient data movement, multiple parallel processing context, headroom available...

- NP's programmability demonstrated to be useful
  - software routing, differentiated services, network monitoring, intrusion detection

- specialized hardware has been used to enhance host's capabilities
  - graphics cards, crypto units, NICs, FPGAs, I2O devices...

Host-ANP pairs

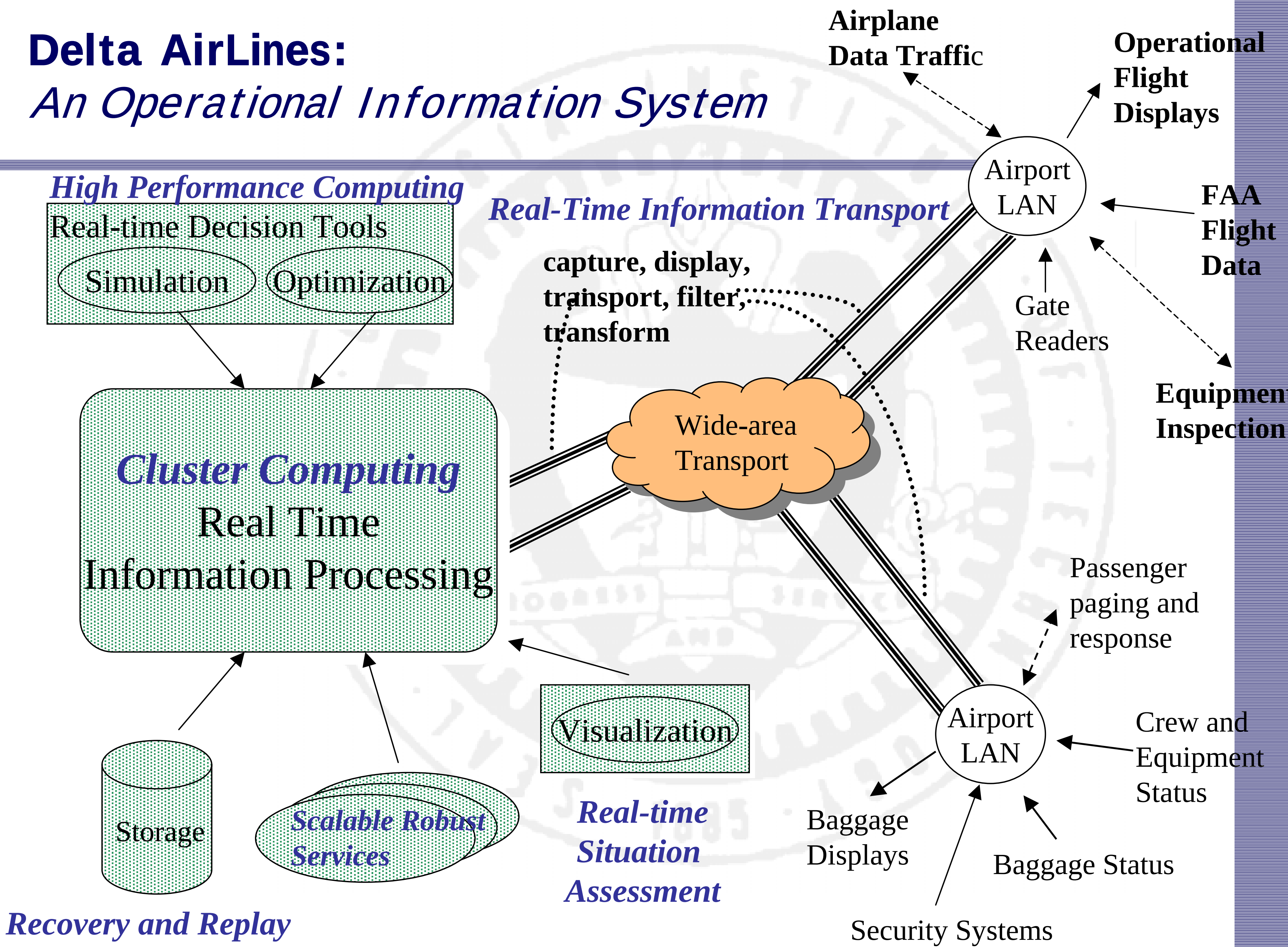


- standard hosts and NPs attached via the PCI interface
- *Receive* and *Transmit* stage on ANP execute core functionality:
  - compose application level data, move data along its data path
- additional functionality is implemented via *handlers* accessing data on ANP and/or host

Sample Applications

- scientific collaborations
  - SmartPointer
- event notification systems
  - stock ticker updates
- delivery of dynamic web content
  - continuous queries
- operational information systems
  - Delta AirLines, WorldSpan

Delta AirLines:
An Operational Information System



Services which can benefit from IXPs

- Adaptive mirroring
  - enterprise cluster enhanced with IXP NPs
- Ill-formed messages
  - IXPs on ingress path into the enterprise system
- Data customization
  - IXPs on egress to perform destination/client-based filtering/multicast…
- Interaction with external partners
  - format translation from internal representation, driven by legacy systems, to standards used by external partners
  - data translation to share with external partners only necessary info
- Business rules execution & pre-processing

Representing application-level actions

- Stream Handlers are lightweight, composable, parameterizable, computation units
- represent application-level processing that can be embedded into data fast path and executed on ANPs.
- can be composed to implement a rich set of application-level functionality
- executed on the fast path by the IXP's microengines
- operate on both packets' header and payload data

Accessing application-level data

- Assembling application-level data
  - RUDP-like, efficient protocol for reassembly and fragmentation of application-level data in IXPs
  - next generation IXP NPs – support for standard protocols
- Interpreting application-level data
  - rely on data format descriptors to interpret and correctly access data
  - XML and internal data representation

Handling formats on the NP

- PBIO – provides interoperability in heterogeneous environments
  - used for internal data representation
  - PBIO-to-XML transcoding at enterprise edges
- permits application evolution/upgrades
  - involves execution of well-defined rules to determine versions, etc.
- format/handler cache & registration
  - controlled through general purpose hosts
  - core?
- middelware-level actions
  - e.g., channel `derivation' in publish-subscribe

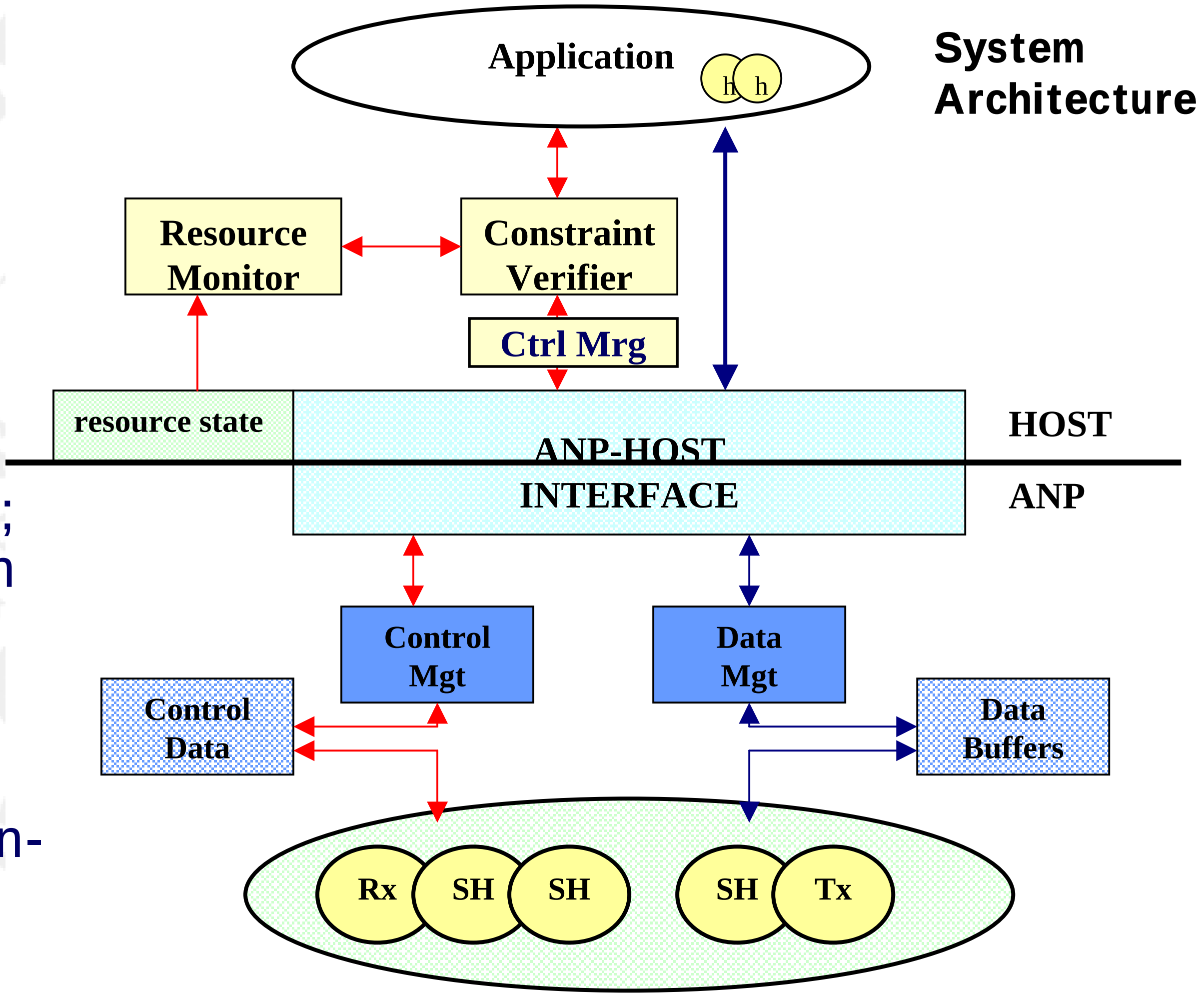
SPLITS

Software architecture for Programmable LIghtweighT Stream handling

- enables joint use of hosts and their ANPs
- deployment of stream handlers onto ANP
- permit application to dynamically reconfigure
  - paths through host-ANP nodes (contexts traversed)
  - services implemented along these paths (handlers invoked)

SPLITS Components

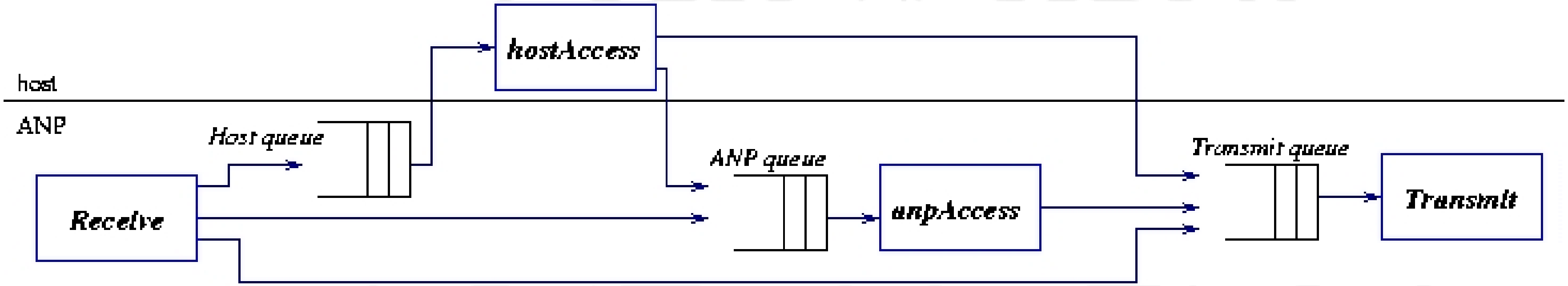
- ANP runtime
  - designated tasks for ANP contexts;
  - free microengines
    - 2 on ixp1200
    - 5 on ixp2400
- Control Mgt
  - interaction with host; runtime configuration
- Data Mgt
  - shared queues for controlled access to buffers of application-level messages



SPLITS Components

- Host-side components
  - maintain information on available handlers
  - API for application interaction with runtime
- Resource Monitor
  - monitor resources along established paths
- Constraint Verifier
  - determine validity of requests for path reconfiguration;
  - uses handler profiles
- Control Manager
  - issue control messages and execution of control protocols

Implementation details



- built on top of host-IXP PCI interface (Mackenzie et al.)
- dedicated ANP contexts for core functionality
  - Rx/Tx, data movement to/from host
  - default data path ANP-host-ANP
  - shared queues among stages
- well-defined activation points along path where stream handler can be invoked
  - runtime configuration in fast memory – checked at activation points
- reserved memory for handler state and parameters

Stream Handlers in SPLITS

- associated with all or subsets of data along data path
- provided by programmer, multiple representations suitable for different activation points
- have handler identifier, access to flow and system state, configuration parameters
- at activation points handler id determines the right offset in the Istore-resident jump table

Dynamic Reconfigurability

- Configure and deploy stream handlers compositions split across multiple execution engines, while still meeting underlying resources
- Dynamically select and deploy handlers and parameters to tune the service implementation to current application needs and network resources
- Enable deployment of new codes without service interruption, by reserving some of the IXP resources

Reconfiguration in SPLITS

- reconfigure both data path and processing applied to the path
  - handler selection
  - parameter passing
  - dynamic hot-swapping
- additional checks can be implemented efficiently and service interruption unnoticeable (28-30us)
- assumption: reconfiguration is not high-frequency

Constraint Verifier

- resource monitor
  - ‘headroom’ on each data path, number and type of memory accesses, instruction count
  - determine maximum amount of resources that can be utilized at a stage
- handler off-line profiling
  - compare with available resources at a stage and verify that handler does not violate that
- admission control
  - based on resource availability and handler requirements
  - based on application-specific data and handler inter-dependencies

Rules Engines

- perform event-action processing
  - in intrusion detection systems
    - e.g., firewalls…
  - in pub/sub middleware,
    - e.g., event notification systems delivering information `where it's needed, when it's needed'
- performing rule processing is complex, but experience with IXP1200 demonstrated that microengines can handle that.

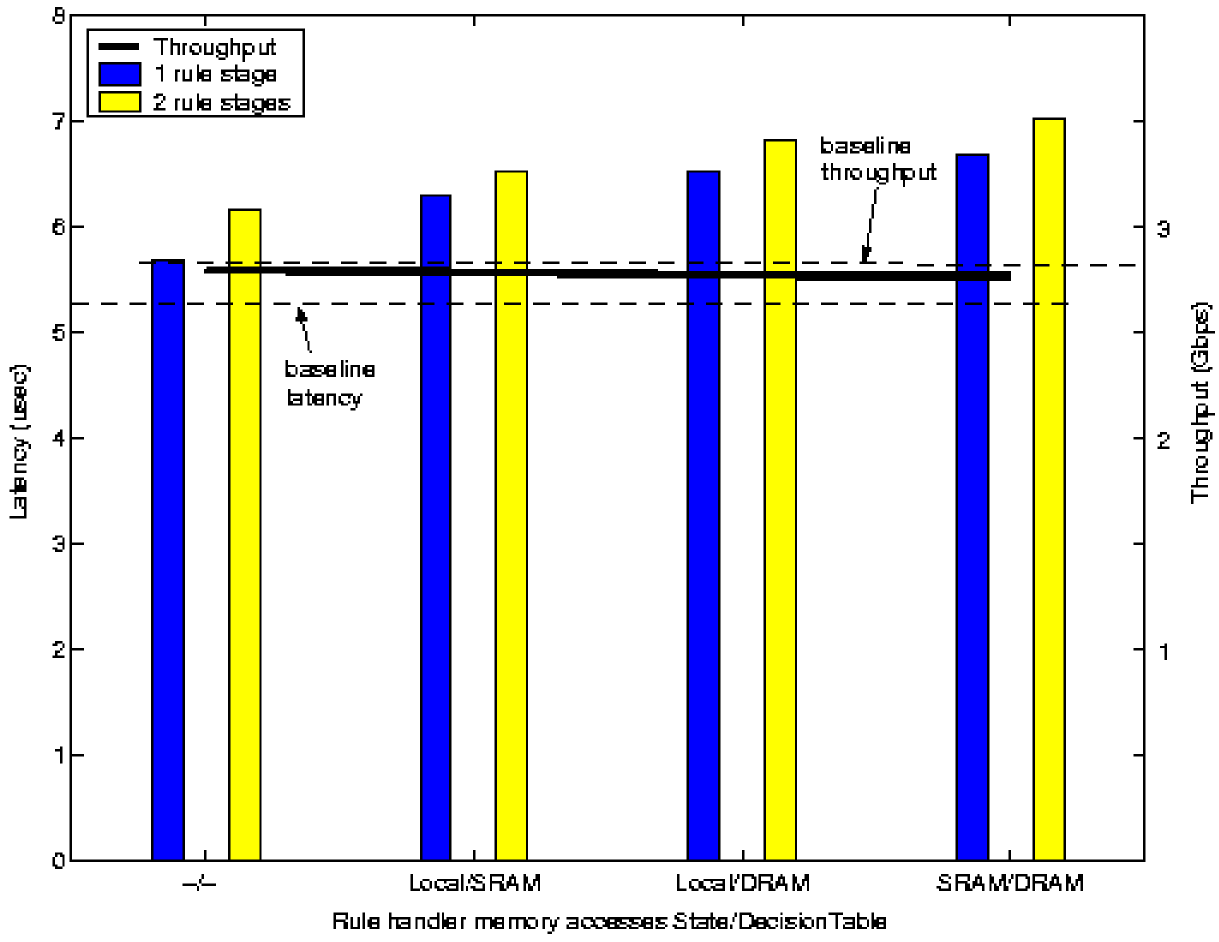
Rules Engines in Enterprise Applications

- in enterprise computing domain rules capture business logic
  - e.g., check-in policies, ticketing...
- tools for dynamic rule addition/removal
  - rapidly adapt to new customer requirements, business environments and regulatory changes
- require ability to efficiently assign event flows to corresponding actions
  - classification issue
- ability to automate ruleflow composition and state management

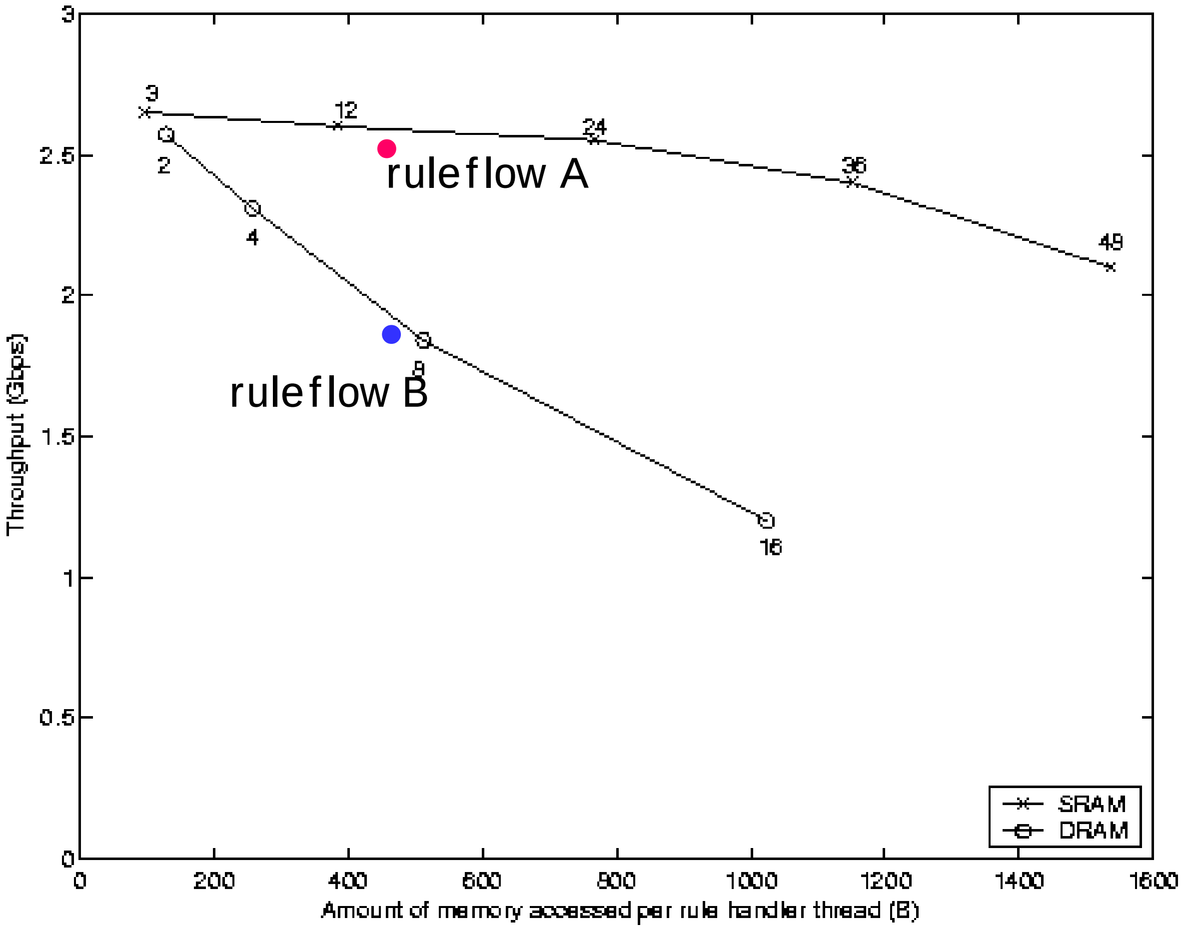
IXP-based Rules Engine

- rule handlers == stream handlers
- use of binary format descriptors in classification
- build configuration mechanisms on top of SPLITS model
- special consideration on state organization and placement
  - application-dependent
- understand constrains which need to be satisfied by rules to minimize perturbation as a result of new rule deployment

Rule execution is feasible on IXP2400



Ruleflows should be formed with consideration of state access requirements



Conclustion

- Network Processors can enhance the processing capabilities of standard systems and deliver significant improvements for application-level services
- Several classes of services can be targeted in enterprise computing domain
- Additional functionality required

  flexible classification, profiling tools and models, faster host-NP interconnect…

Thank you.