Extensible Network Security Services on Software Programmable Router OS

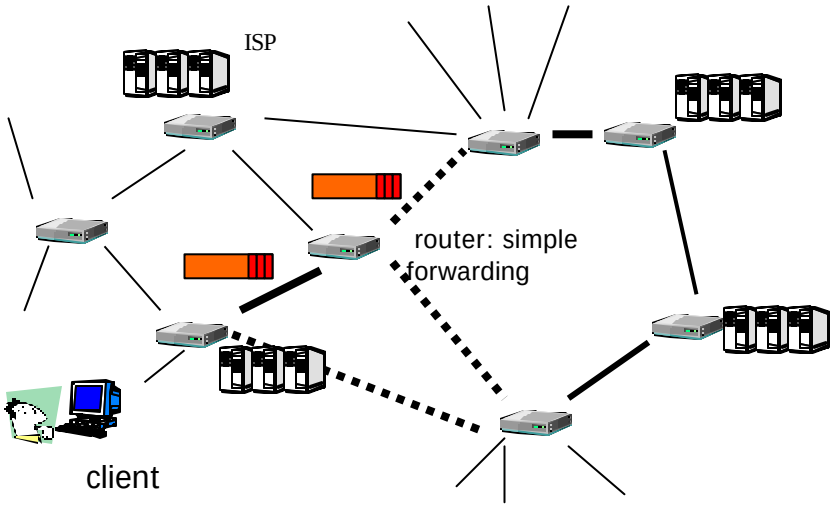
David Yau, Prem Gopalan, Seung Chul Han, Feng Liang

System Software and Architecture Lab
Department of Computer Sciences
Purdue University

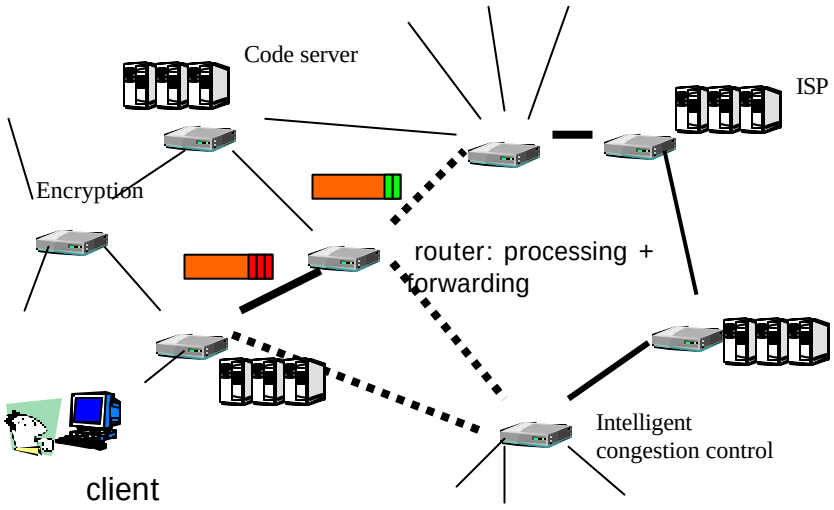
Motivations

- Value-added services router technologies
- Need for secure network infrastructure
  - attacks are easy (DDoS, address spoofing, network intrusions, ...)
  - need to detect/trace/counter attacks in flexible ways
    - protect innocent users, prosecute criminals, etc

Existing Networks



Value-added Services Networks



Project Goals

- OS prototype (CROSS) for value-added services routers
- Security services to be dynamically injected into routing infrastructure

Router OS Challenges

- Heterogeneous users
  - needs, priorities, purchased shares
- Untrusted programs
  - greedy, buggy, malicious, ...
- Diverse resources
  - space-shared, time-shared
- Diverse resource *bindings*
  - multi-processes, multi-threads, multiplexed threads

The CROSS Approach

- Virtualized router resources
  - virtual machines
- Orthogonal fine-grained allocations
  - Resource Allocation objects
- Flexible/scalable packet classification
  - resource binding, per-flow processing
- Efficiency, modularity, configurability

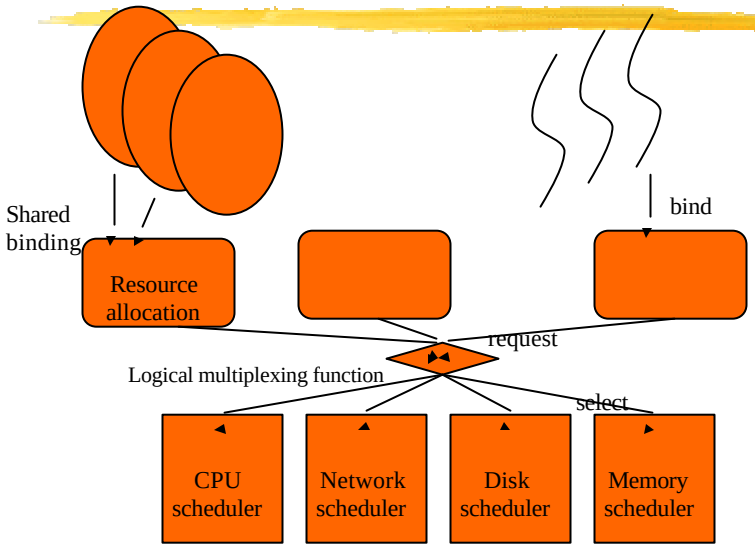
Resource Virtualization

- Hierarchical scheduling
  - virtual machines with different APIs
  - user allocations on demand
- Target resource types
  - CPU time
  - network bandwidth
  - memory pool capacity (virtual memory)
  - disk bandwidth

Resource Abstraction

- Kernel *Resource Allocation* objects
- Independent/orthogonal objects
  - relative to resource consumers
- Flexible bindings to resource consumers
  - shared binding
  - dynamic binding (with run-time information)
  - configurable parameters

Resource Allocations



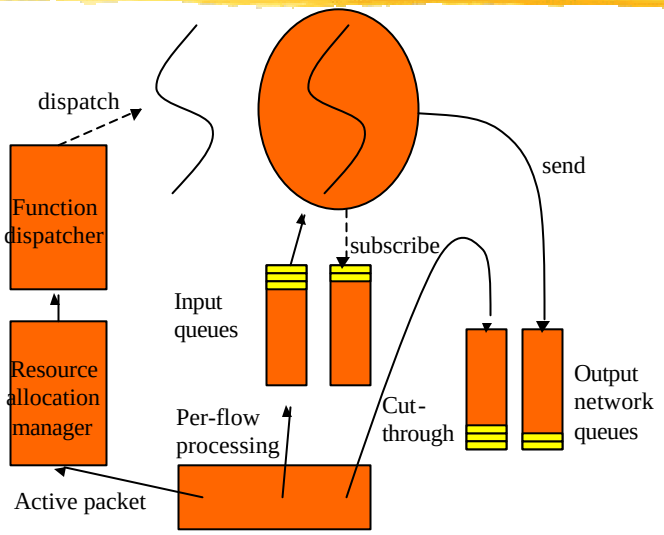
Packet Forwarding

- Three possibilities
  - active program dispatch
    - trusted (kernel thread), untrusted (user process)
  - Per-flow processing
    - subscribed by dispatched router programs
    - security processing, application-level routing
  - Cut-through fast path
    - minimal delay

Packet forwarding decision

- Based on packet header information
- Packet classification
  - scalable to many dimensions
  - scalable to many classification rules
  - flexible
    - support multiple and least-cost matches

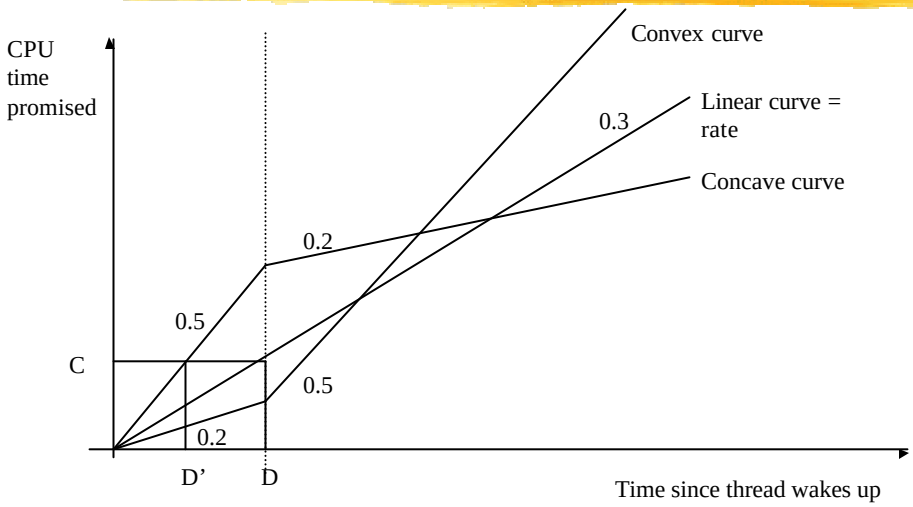
Cross Forwarding Paths



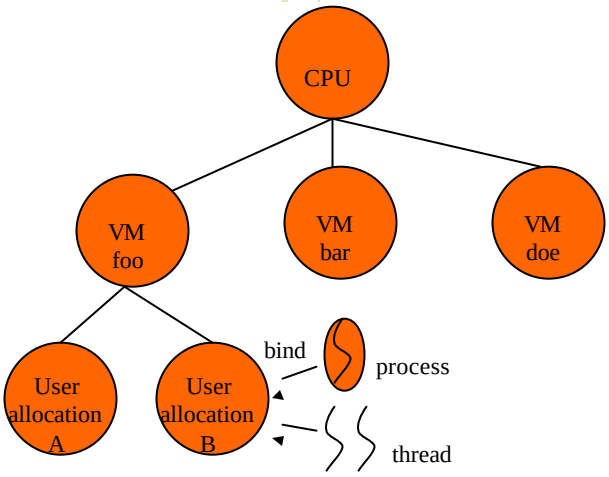
Example Scheduler: CPU

- Hierarchical partitioning using *fair service curves* [MMCN 2001]
- Decoupled delay and rate allocation
  - good for low delay and low rate applications
- Solution to priority inversion
  - lock contention and client/server interaction
- Performance
  - rate/delay guarantees, proportional sharing, minimized unfariness

Service Curve



CPU Sharing Hierarchy



System Implementation

- Extension to Solaris 2.5.1/Linux 2.2.x/2.4
- Deployed on UltraSPARC/Pentium network
  - Ethernet, Fast Ethernet, Myrinet
- Modular subsystems with well-defined interfaces
- Simple command interfaces to launch legacy applications

Basic Costs

- Resource Allocation control
  - create
  - delete
  - bind/unbind
- Function dispatch
  - thread: about 145 microseconds, low variance
  - process: 0.77 to 1.1 ms, application-dependent

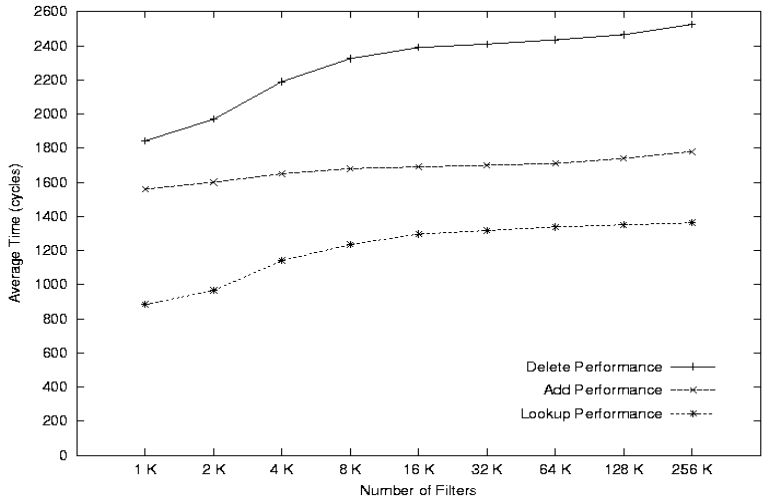
Resource Allocation Costs (microseconds)

Operation	kernel	user
Bind	4.8	9.0
Unbind	2.4	6.6
Create + delete	15.4	19.6

Packet Classification

- Five dimension
  - exact, prefix, range, wildcard
- Database size up to 256 K rules
- Average lookup cost of 7.8 microseconds
  - 1.1 Gb/s for 1000 byte packets
- Add/delete 10.8/14.9 microseconds
  - 67,000 updates per second

Packet Classifier Performance



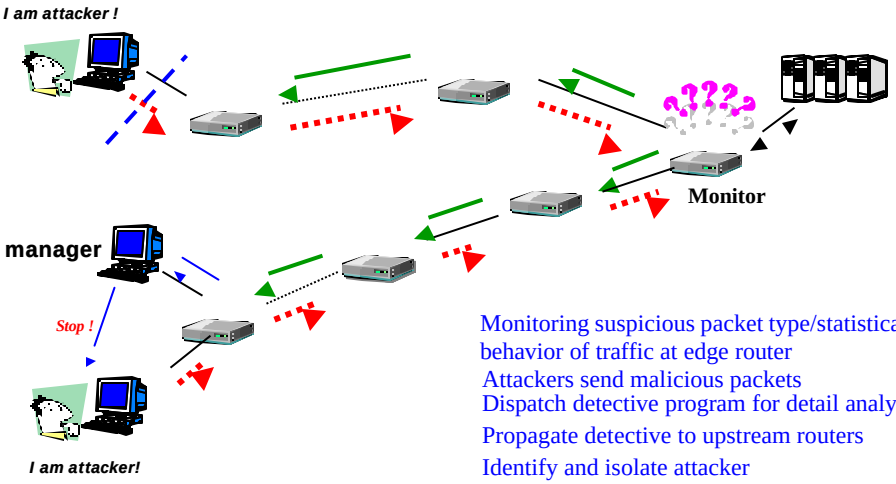
Example Service: SmartTrack

- Track back to sources of DDoS attacks
- When an attack is detected, launch a CROSS program detective at affected routers, and recursively trace back upstream
- Enable a victim domain to take various actions in response to an attack

SmartTrack Mechanism

- Attack monitors deployed at edge router
- Check for suspicious packet/statistical behaviors of incoming traffic
- When attack detected, dispatch a detective program to perform more careful traffic analysis
- Propagate detectives to upstream routers
- Result: identify and isolate attackers

SmartTracking



Monitoring suspicious packet type/statistical behavior of traffic at edge router
Attackers send malicious packets
Dispatch detective program for detail analysis
Propagate detective to upstream routers
Identify and isolate attacker

Further information

- System Software and Architecture Lab
  - http://ssal.cs.purdue.edu
  - CROSS paper: Resource Management in Software-programmable router OS (*IEEE JSAC*, March 2001)
- People
  - David Yau, project director (*yau@cs.purdue.edu*)
  - Prem Gopalan (*gopalapk@cs.purdue.edu*)
  - Seung Chul Han (*han@cs.purdue.edu*)
  - Feng Liang (*liangf@cs.purdue.edu*)

CPU/network Scheduling

- Network respond application
  - driven by received packets
  - do some CPU computation, send some network data out
- Total delay budget of 3.5 seconds
  - CPU one second, network 2.5 seconds
  - CPU two seconds, network 1.5 seconds
- Allow both *rate* and *delay compositions*

Rate Composition

Udpburst CPU rate	Greedy CPU rate	Achieved bandwidth (Mb/s)
5%	95%	3.6
10%	90%	7.8
15%	85%	9.8
20%	80%	9.8

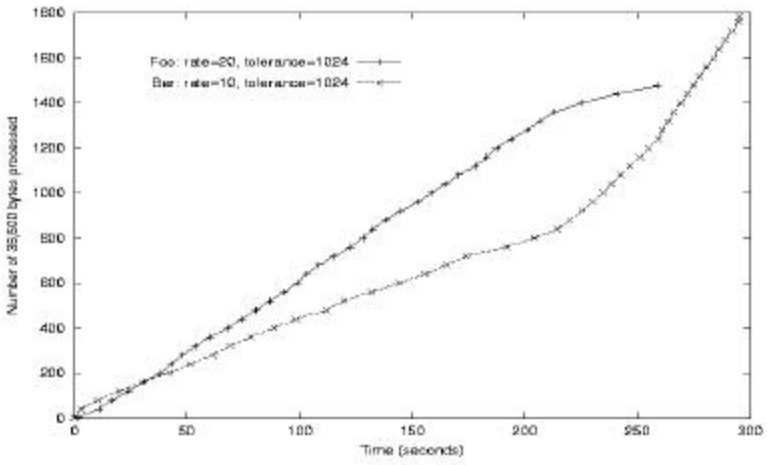
Delay Composition (microseconds)

Run	Mean CPU delay	S. d. CPU delay	Mean net delay	S. d. net delay	Mean total delay	S. d. total delay
1	1.06	0.007	2.31	0.136	3.39	0.144
2	2.00	0.096	1.48	0.136	3.49	0.172

Disk Scheduling

- Program: search through an input file sequentially for some pattern
- Two groups of 10 processes each
  - group one: reading 65,588 kbytes, with allocation of rate 10
  - group two: reading 55,789 kbytes, with allocation of rate 20
- Equal CPU allocations, disk placement not controlled

Proportional Disk Sharing



Memory Scheduling

- Footprint application
  - repeatedly touch a set of *n* distinct pages
- Result summary [JSAC 2001]
  - isolation properties
  - utilization of over-reserved pages
  - reclaim of reserved pages

Related Work

- Router Plugins (Washington U)
  - extensibility, quick resource binding through *gates*
- Extensible Router (Princeton)
  - kernel built from scratch, *path* abstraction
- Bowman (Georgia Tech/U Kentucky)
  - Posix user-level implementation for portability

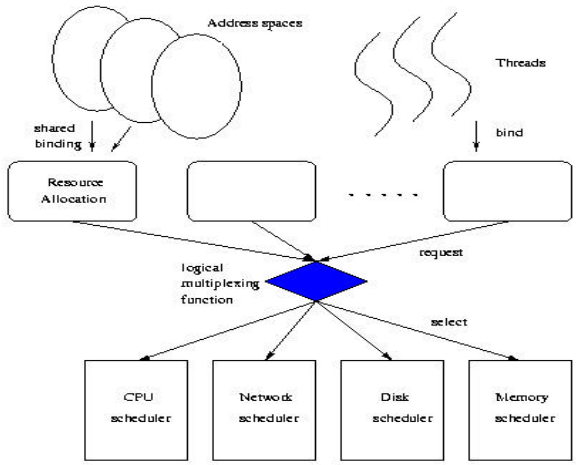
Related Work (cont'd)

- Active node with ANTS (MIT)
  - *externally* certified program capsules
- Flexible *end-system* scheduling
  - Resource Containers (Rice)
  - Software Performance Units (Stanford)
  - Reservation Domains (AT&T)

Conclusions

- Resource management important for software-programmable routers
- Presented system prototype as solution step
  - packet classification
  - router program dispatch
  - unified and orthogonal resource abstraction
  - schedulers for major resource types

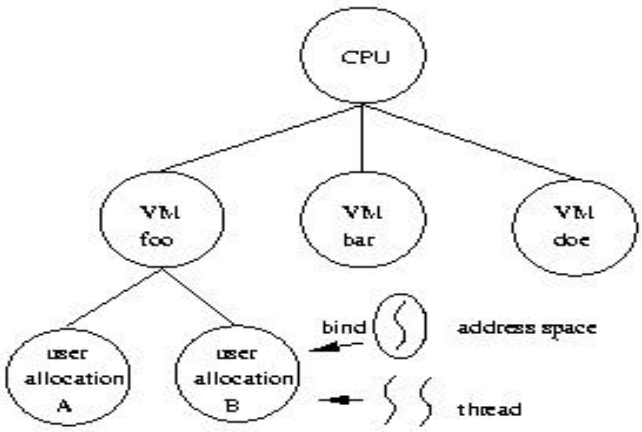
Resource Allocation



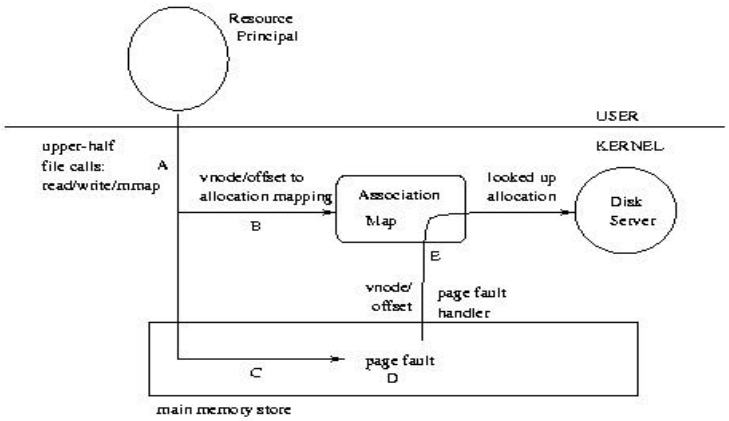
Scope of System Integration

- CPU scheduling
  - threads and processes
- Network scheduling
  - packets in Stream buffers
- Memory scheduling
  - page frames, MMU reference bits
- Disk scheduling
  - buffer header structures

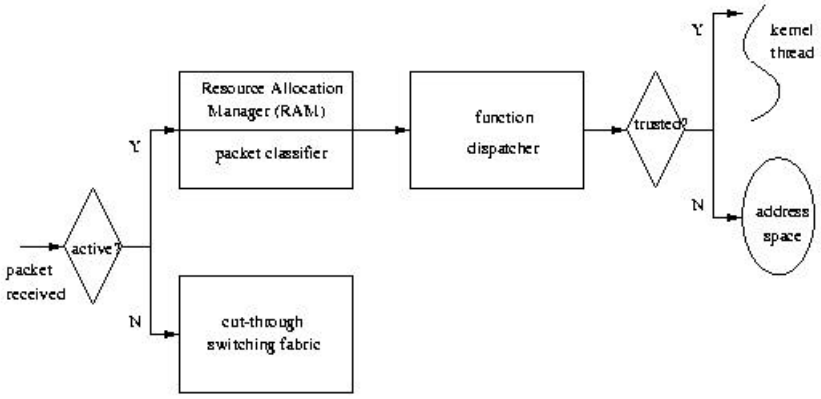
CPU Sharing Hierarchy



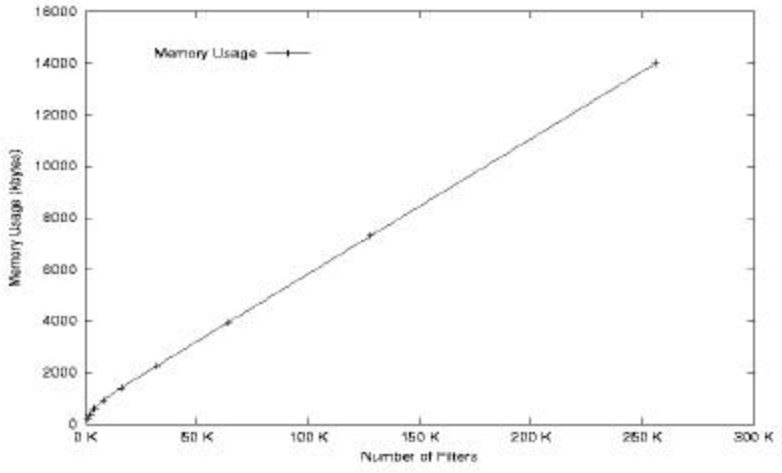
Disk Reservation Binding



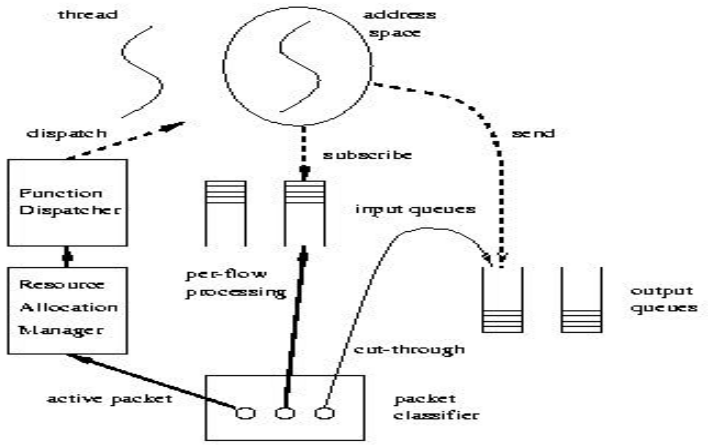
Packet Processing Paths



Packet Classifier Memory



CROSS Processing Paths



Resource Allocation API

- Create/delete
  - named by object system-wide *key*
- Bind/unbind
  - affect calling thread/process
  - key to *fine-grained* resource management
- Control
  - change scheduling parameters, owner, ...
- User-level access through pseudo-device

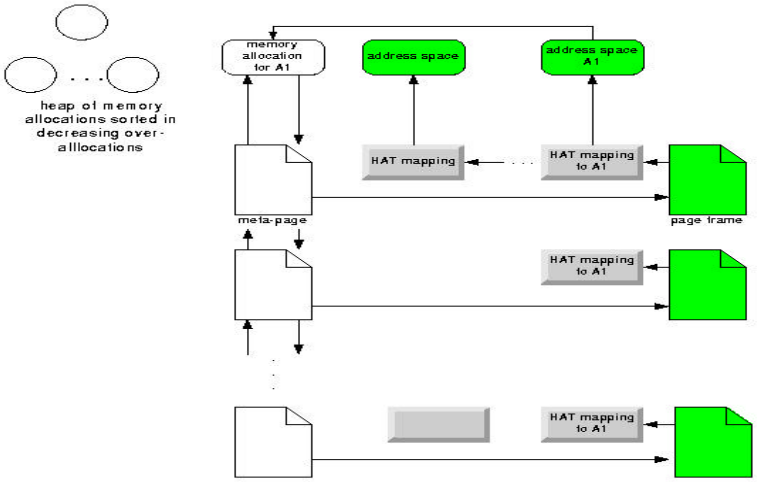
System Integration

- Leverage against Solaris gateway OS
  - support for existing application
  - immediate access to software development platform
- Implication
  - need to work with existing Solaris abstractions
  - threads/processes, stream buffers, page frames, buffer header structures, ...

Memory Scheduler

- Guaranteed *share* per allocation
  - minimum number of page frames that allocation can map simultaneously
- Guaranteed-share scanner algorithm
  - consider pages for replacement in decreasing *over-allocation* order
  - second chance to referenced pages
    - allow reserved but unused pages to be utilized

Memory Allocation



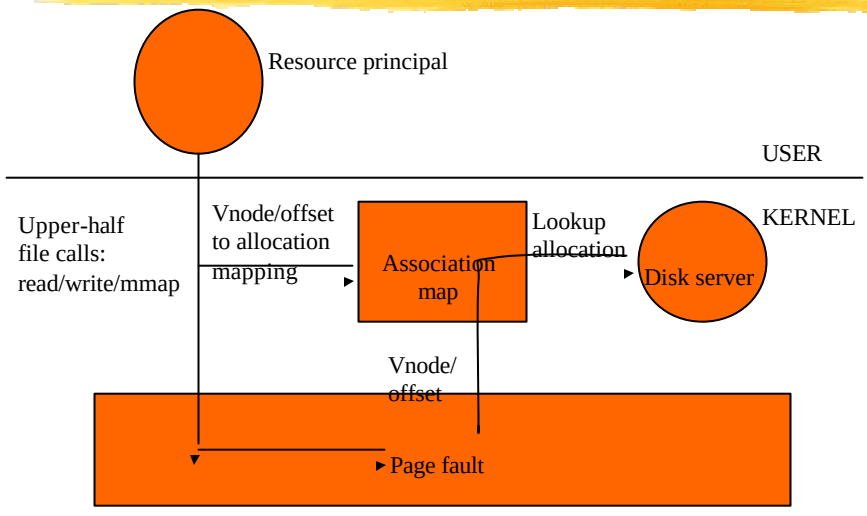
Disk Scheduler

- Provide proportional sharing
  - conflict with efficiency goal to minimize seek time overhead
  - notion of *eligibility* to balance between the two goals, using *tolerance* parameter
- Integrated with file systems
  - problem: applications do not access disk directly!

File System Disk Access

- Resource Allocation bound to resource principals
- Resource principals do read/write/mmap system calls
  - disk accesses avoided unless file system page faults
- Page faults occur in *interrupt* context!
- Solution: *Association Map* on vnode/offset

Association Map



Bandwidth scaling with 64 and 128 byte packets