Making Paths Explicit in the Scout Operating System

David Mosberger and Larry L. Peterson

{davidm,llp}@cs.arizona.edu

The University of Arizona

Outline

1. Motivation
2. Path Architecture
3. Scout Example
4. Conclusions

Network Appliances

- Network Devices
  - cameras
  - disks
- Smart Cards
- Set-top Boxes
- Multimedia Stations
- Scalable Server Nodes

Note: Network appliance is more than just a fancy name for computers. There are some distinctive properties...

NA Characteristics

- Communication-oriented
- Specialized functionality
- Diverse functionality
- Predictable performance
- Scarce resources

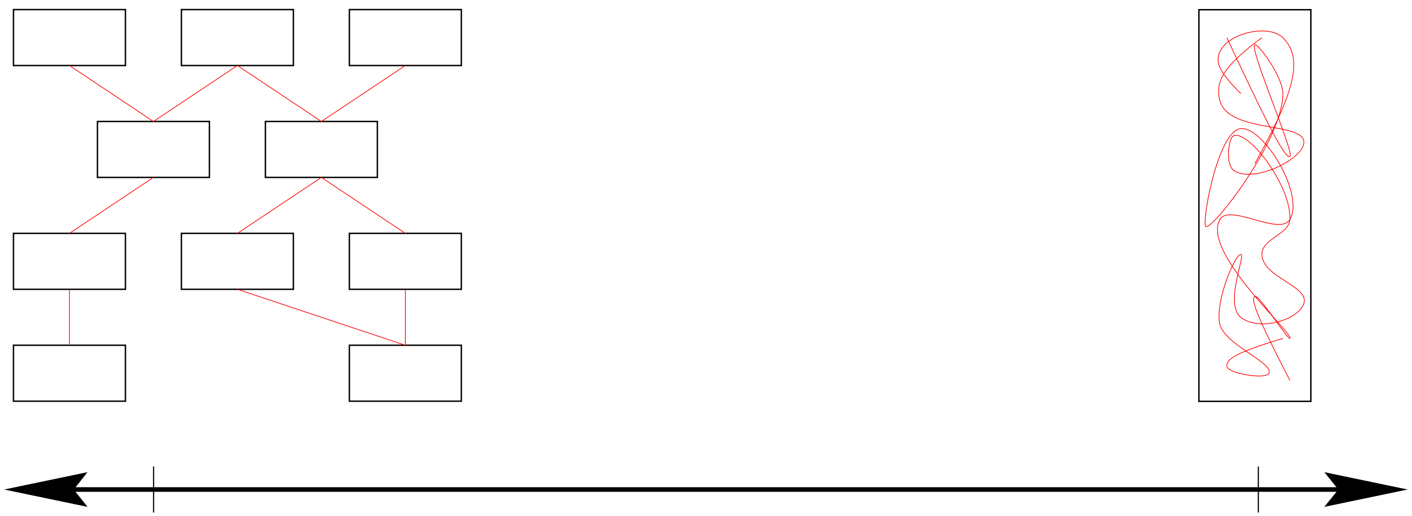
Don't abbreviate Network Appliance during speech!!

Diverse functionality often causes creeping generalizations (bad OS designer habit).

Predictable: reliable and realtime (to some degree)

NA Software

Modular design: Vertical Integration:



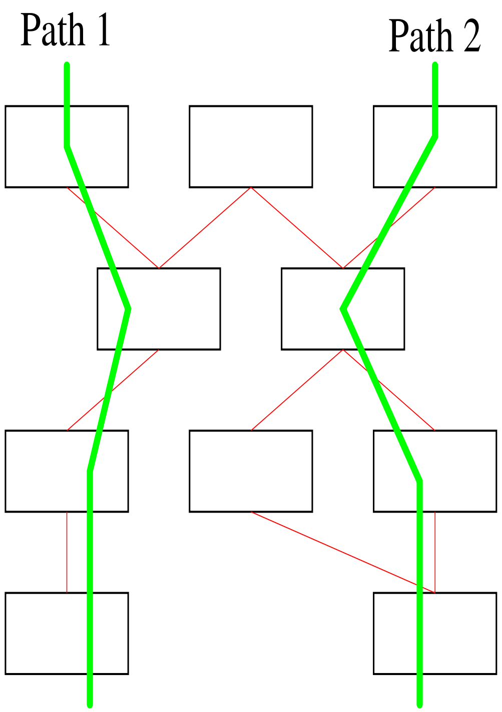
Flexible
Reusable modules
Performance suffers

Best performance
Complex
Expensive

Note: performance suffers since NAs are communication oriented. I.e., most time is not spent in a single module but divided across all the modules that the data crosses.

Paths: An Alternative Solution

- Modularity for:
  - flexibility
  - managing complexity
- Paths to solve problems requiring global context:
  - improve code quality
  - make better resource management decisions



Paths can be interpreted as a (mostly) automatic way to derive a vertically integrated system form a modular system.

Paths: Examples

- Synthesis [Massalin & Pu]: Specialize code for entire path
- Fbufs [Druschel & Peterson]: Fast data-transfer by knowing “domains” crossed
- ILP [Clark & Tennenhouse, Abbott]: Collapse data-processing of several layers
- Migrating Threads [AlphaOS, Spring, FLUX]: Improved accounting for thread crossing servers

Paths: Fundamental Idea

Expose and exploit non-local context...

...in a systematic manner and based on a modular system.

- Synthesis: Know *all* the code and runtime-constants.
- Fbufs: Know the set of "domains" crossed early and at any point in time.
- ILP: Know the sequence of data manipulations.
- Migrating threads: At any point, know who to "charge."

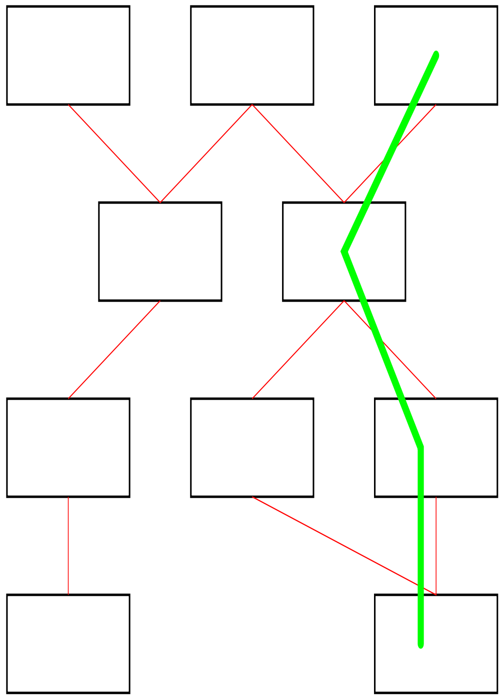
Need to say something about elevating paths from an ad-hoc solution to a fundamental OS abstraction.

Outline

1. Motivation
2. Path Architecture
3. Scout Example
4. Conclusions

Path Architecture

- Router Graph
  - defined at build time
- Invariants
  - specified at open time
- Path
  - sequence of fixed routing decisions



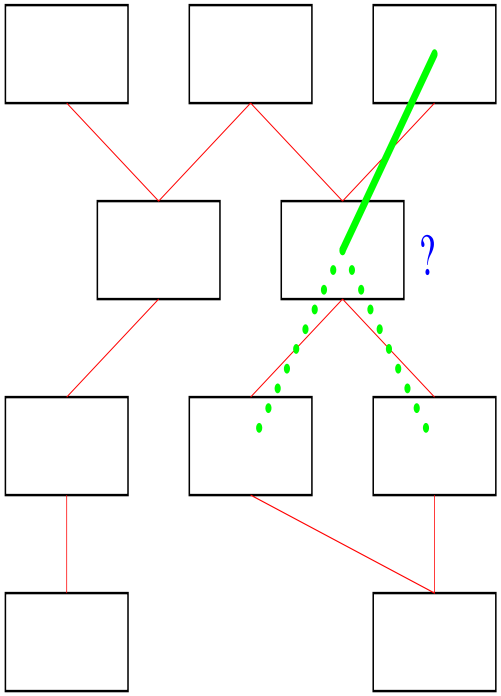
Transition: Now to the fine print (sequence of problems that make this hard).

Path Creation

$p \leftarrow$ `pathCreate`$(router,\ invariants)$:

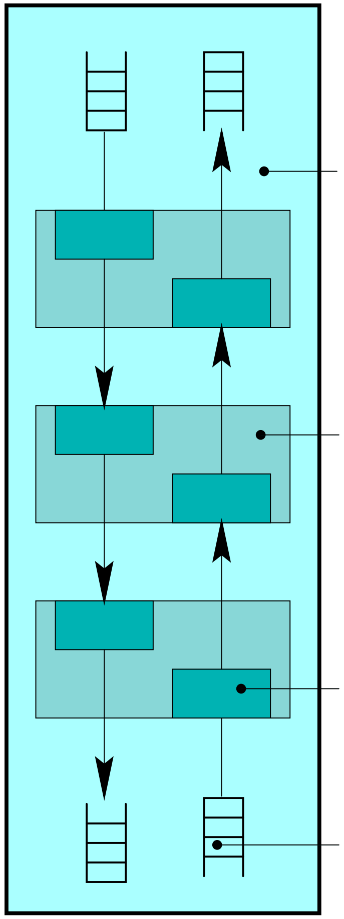
1. Create incrementally:
   Each router makes (local) routing decision, then forwards request.

2. Optimize globally:
   Apply path transformations that match created path.



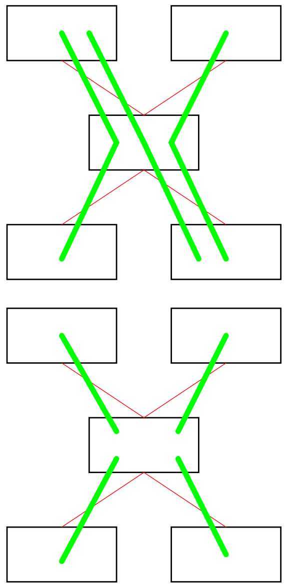
Path Object

- Each router contributes a stage
- Interfaces to move data through path
- Queues to decouple scheduling



How to Use Paths

- Wide vs. narrow
  - per data item
  - per connection
- Long vs. short
  - Long for best optimization
  - Short for generality or minimum memory usage



⇒ path architecture must be policy-free

Mention “dynamic routing decision” with the “generality” bullet.

Path Execution

- Threads:
  - Created by routers
  - First-class objects
- Path execution:
  - Router assigns thread to initiate execution of one of its paths
  - Thread inherits scheduling parameters from path
  - Same thread can cross several paths

Finding Paths

- Usually obvious
  - e.g., file descriptor, window handle, ...
- Sometimes data-dependent (e.g., based on network headers):
  - Each router provides demux operation:

    $$(pset', next\text{-}router) \leftarrow \texttt{demux}(pset,\ msg)$$

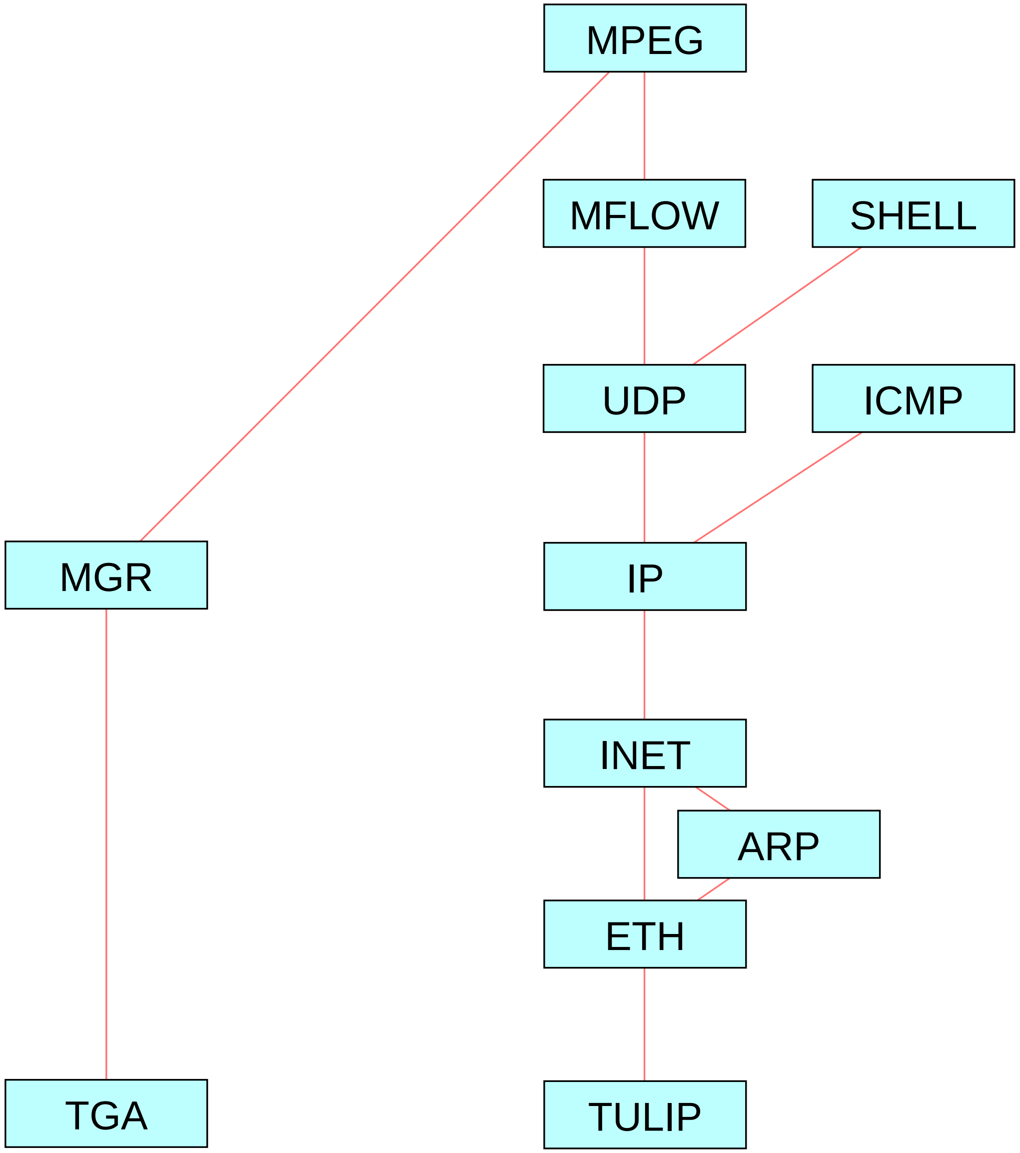
  - If final $pset'$ contains exactly one path, deliver to that path, otherwise drop

Outline

1. Motivation
2. Path Architecture
3. Scout Example
4. Conclusions

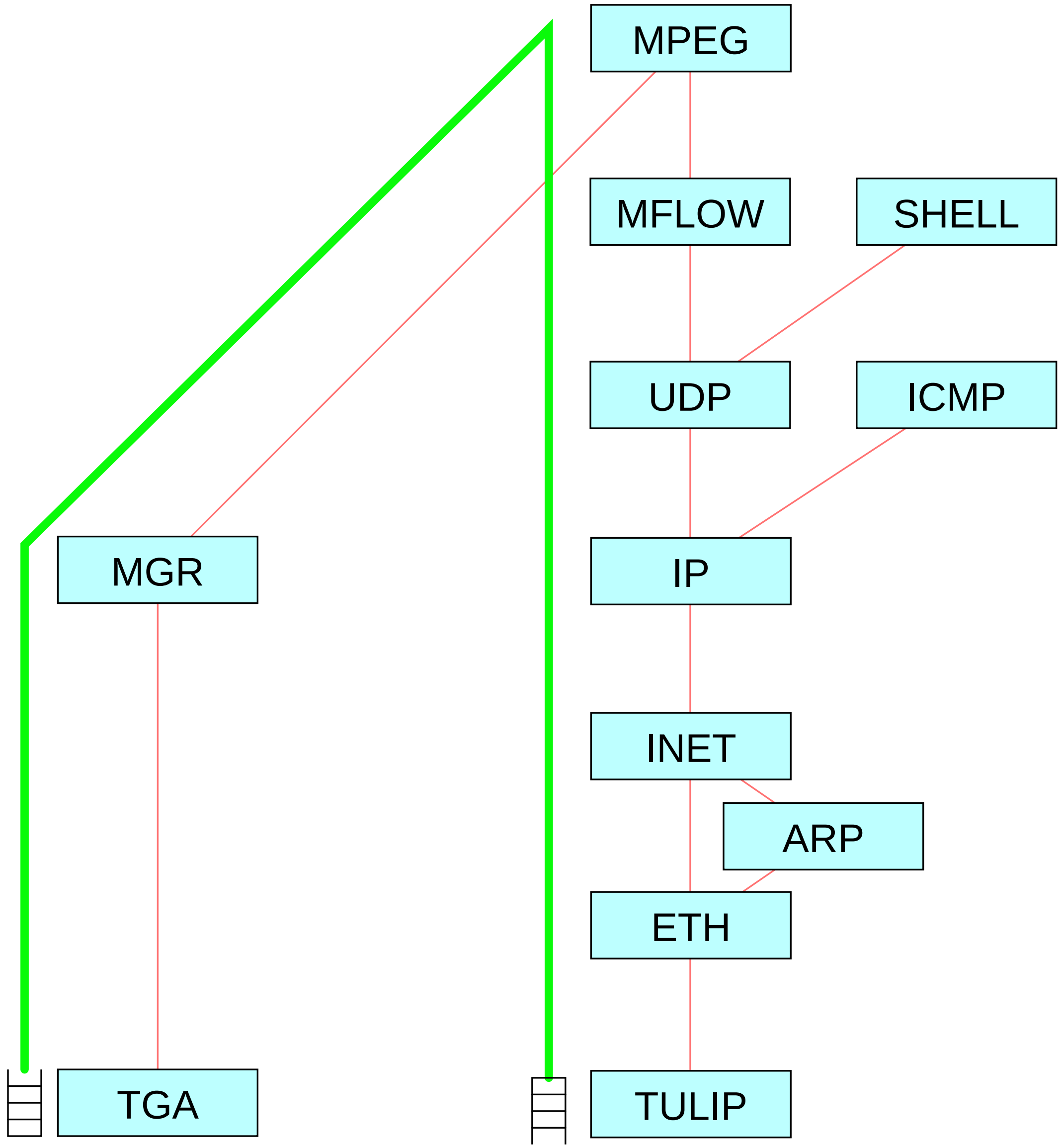
See paper and demo outside for more.

Scout MPEG Demo



- Scout: path-based OS, targeted at Network Appliances
- MPEG (tell me about picture)
- Problems due to lack of path

Scout MPEG Demo



- Explain with path:
  - Only one path shown, but can have many
  - Know in Ethernet driver whether packet belongs to MPEG vs. ICMP.
  - Can schedule entire processing according to bottleneck queue (input or output) and with appropriate policy (e.g., RR vs. EDF)

Resource Management Benefits

- Early segregation of work $\Rightarrow$ avoids priority inversion
- Schedule according to bottleneck queue
- Discard unnecessary work early
- Could use for admission control

Conclusions

- Defined path architecture
- Implemented Scout, a path-based OS, and demonstrated MPEG decoder
- Many questions need more research:
  - Resource scheduling & path policies
  - Applying paths outside of networking domain:
    - File systems [Hartman & Chiu]
    - Secure paths [Spatscheck]
    - Windowing systems [Khune]

- Global comment: With paths, we start to look at the system differently.

Scout Project Info

http://www.cs.arizona.edu/scout/

Acknowledgments

Wanda Chiu, Robert Graham, John Hartman, Abhiram Khune, Dave Larson, Brady Montz, Rob Piltz and Todd Proebsting.