Control-On-Demand

Samrat Bhattacharjee
Networking and
Telecommunications Group,
College of Computing,
Georgia Institute of Technology.

Gísli Hjálmtýsson, Jennifer Rexford
Network Mathematics Research
Networking and
Distributed Systems,
AT&T Labs — Research.

Active Networking

- Placement of *user-controllable* computations *inside* a network
- Provision of an *uniform* interface to the network
- Current Approaches:

Project	Language	
C-o-D		AT&T
Smart Packets	Spanner/Sprocket	BBN
Netscript	Netscript	Columbia
CANEs		Georgia Tech.
ANTS/Capsules	Java	MIT
SwitchWare	PLAN	Univ. of Penn.

Control-On-Demand — Key Ideas

- Networks already support different *transport* needs
  C-o-D enables networks to support different *control* needs
- Flow *specific* control can be specified by the flow participants
  Each receiver in a multicast can specify different desired flow characteristic depending upon network conditions
- Explicit Flow labeling
  Use the IPv6 Flow label
  Flow Pinning
- Efficiency
  Staying in the fast path

Staying in the *fast* path

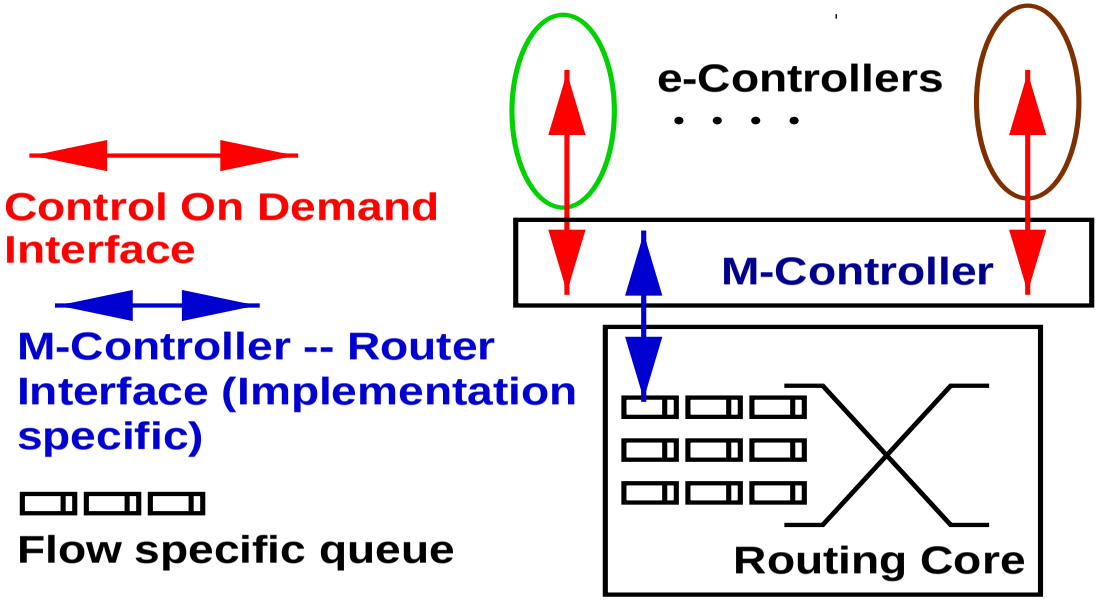
- Reducing on-line requirements
  - Enhancement control

    Flow semantics not modified by controller not being present
  - Best-effort control

    Controller does not have to process *every* packet
- Reducing bandwidth across the interface
  - Frame Peeking
- Lightweight Signaling (using extension headers)

Architecture

- Meta-Controller
  - Interface between routing core and flow specific controller
  - Responsible for installing and initiating flow specific controller



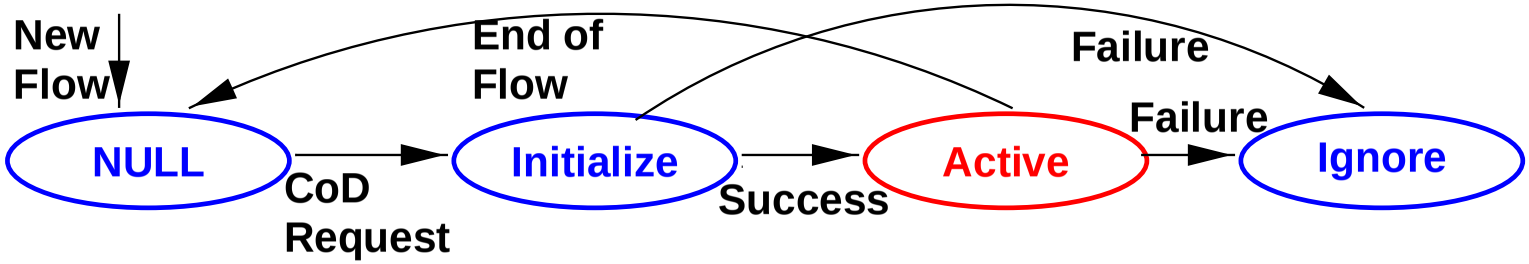
Flow-specific Controllers

- Dynamically installed by the M-Controller
- Uses the Control on demand interface to customize per-flow control
- Controller performance enhanced by:
  - Application level framing
  - Frame Peeking
- C-o-D interface is designed to minimize *coupling* and *synchronization* between the per-flow control and the routing core

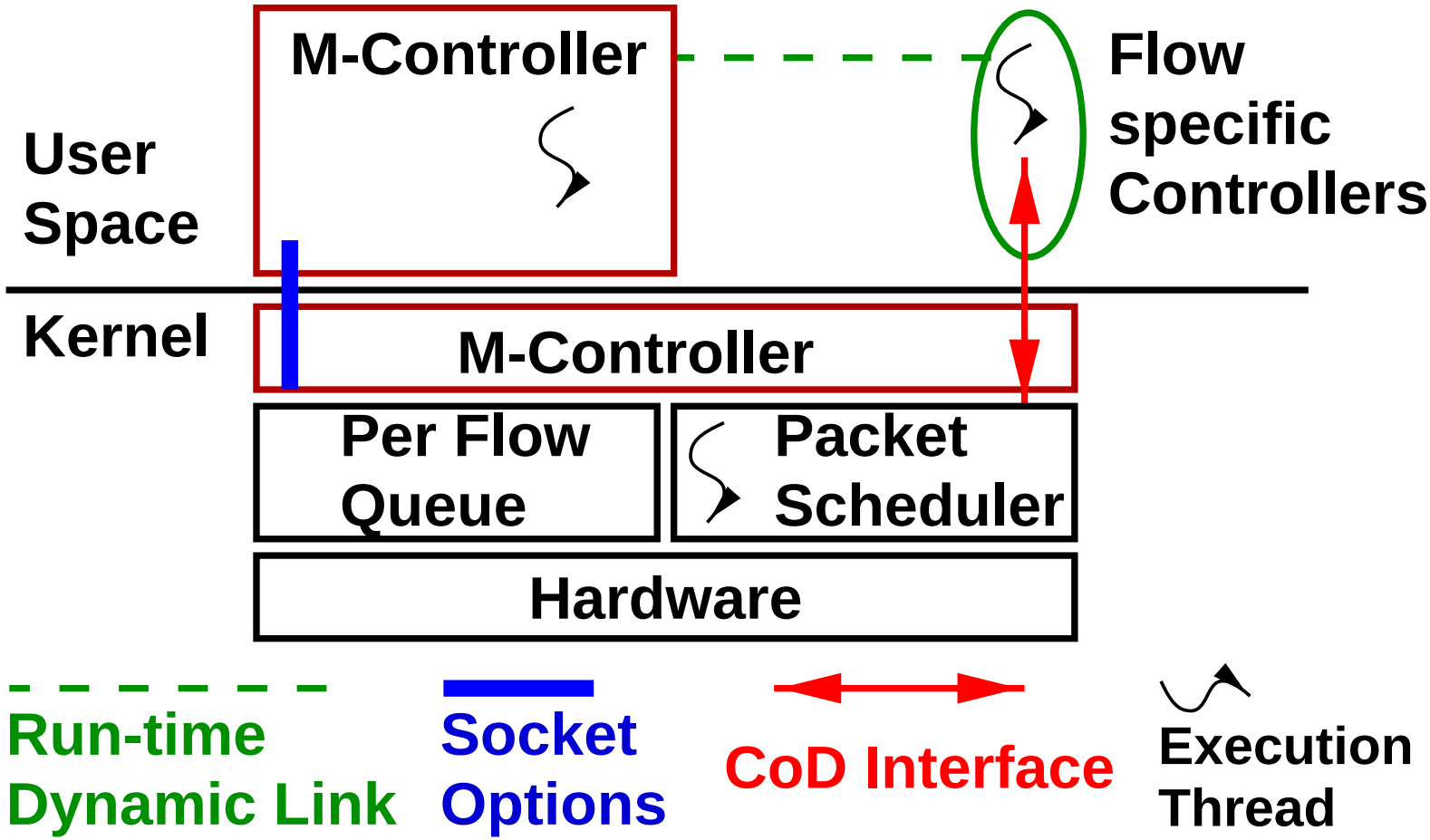
Flow Classification

O(1) flow classification based upon flow id. into 1 of 4 states:

`NULL`	if C-o-D header present
	copy packet to M-Controller
	set state to `Initializing`
	create FCB, cache output port, forward packet
`Initialize`	M-Controller knows about flow,
	FCB exists, keep statistics, forward packet
`Active`	Flow specific controller active, queue
`Ignore`	Error condition, forward packet



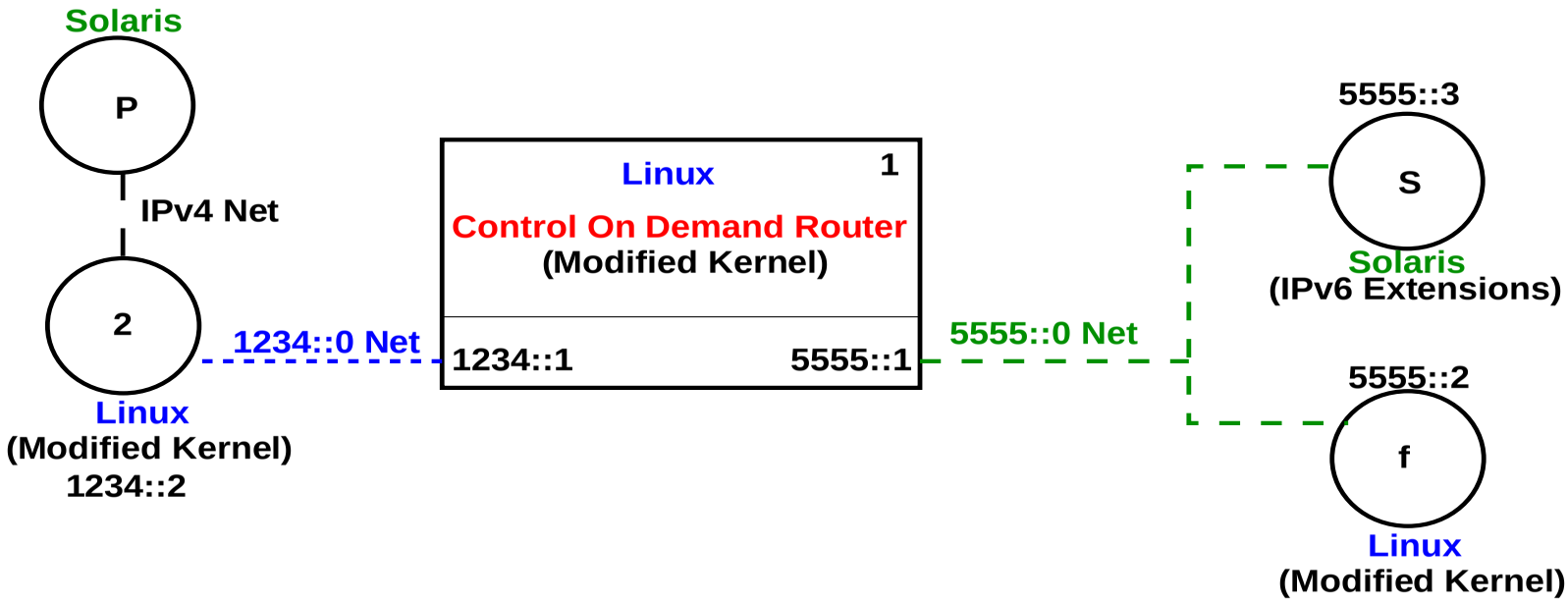
Implementation Architecture



Implementation Specifics

- Linux in-Kernel C-o-D implementation
- M-Controller–Routing-Core interface through socket options on an unconnected raw socket
- Flow specific controllers installed using run-time dynamic linking
- Additions to Linux IPv6 implementation :
  - Flow state caching and flow pinning
  - Per flow queuing
  - Per flow output queue scheduling

Implementation Topology



- Virtual Topology — 10 Mbps Ethernet implementation

C-o-D Controller Primitives

Command	Parameters	Effect
CHANGE-FLOW-STATE	{Flow, State}	Set flow state
EXPOSE-QUEUE	{Start Offset, Length, *Which*}	{Queue Info., # of matches, {matches}}
SET-SCHEDULE	{{Byte$_i$,Rate$_i$}}	After Byte$_i$ bytes served, output rate set to Rate$_i$
SET-PACKET-STATE	{{Packet$_i$, State$_i$}}	Set packet state (e.g. `Discard`)

Applications

- Application-specific traffic shaping
  Video smoothing, Stream thinning
- Application-specific congestion control
- *Reliable Multicast*
- *Variegated Multicast trees*

Application-specific traffic shaping

- **Stream thinning :**

  Upon congestion, application specified headers are parsed to discard *least* important datagrams

  In case of MPEG video — I frames are preserved; B and P frames are discarded first.

  If entire frame cannot be fit into output buffer, all packets corresponding to frame is discarded

- **Video Smoothing :**

  Controller provided with client buffer size and frame sizes

  Provides proper rates such that buffers are not over/underflown

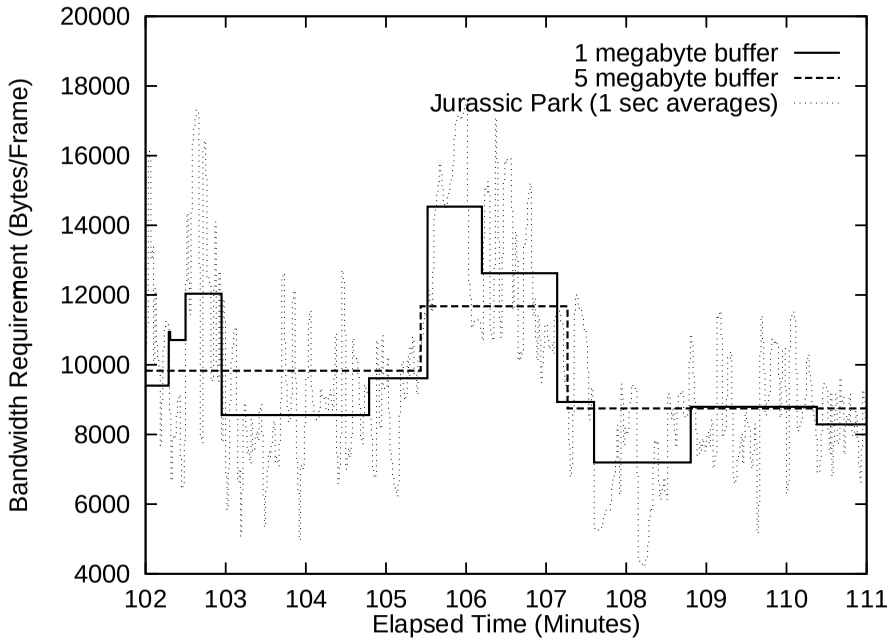
Application of the CoD Interface

Stream Thinning

```
; EXPOSE-QUEUE{offset, length, which}
EXPOSE-QUEUE{0, sizeof(app-level-header), NEW-PACKETS}
if {Qinfo.Bytes-Queued > (0.8 QSize)}
   for (i = 0 ; i < # matches ;  i=i+1)
        f ← (app-level-header) matches[i]
        if ({f.type != I-FR} and {Qinfo.Bytes-Queued > 0.3 QSize })
             Pinfo[Pinfo.valid].id ← matches[i].id;
             Pinfo[Pinfo.valid].state ← PACKET-DISCARD;
             Qinfo.Bytes-Queued −= f.len;
             Pinfo.valid=Pinfo.valid+1;
if {Pinfo.valid > 0}
   SET-PACKET-STATE{Pinfo}
```

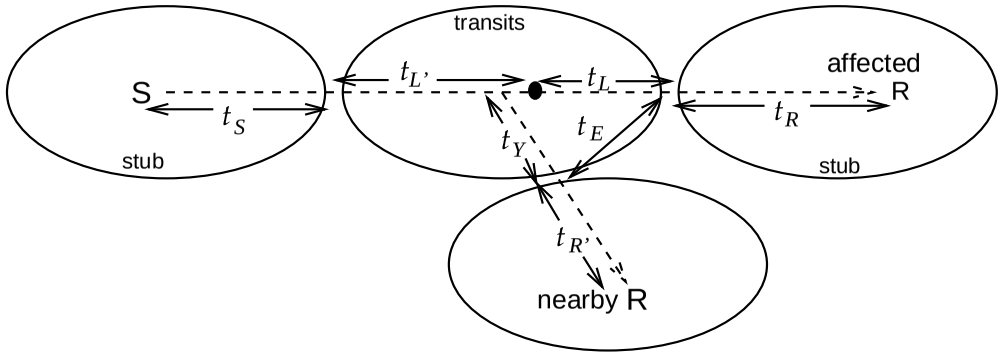
Video Smoothing

- Controller schedules frames to prevent client buffer over/underflow
- Given frame sizes, client buffer size — controller can compute transmission rates over time intervals



Applications in development

- Association between packets within a stream (stream thinning using a Multi-Description Coder (with Vinay Vaishampayan))
  - Video coded into two sub-streams of *equal* importance
  - Controller provides *at least one sub-stream*
- Reliable Multicast
  - Caching at intermediate nodes — *best possible* retransmission scheme



Future Work

- Implementation of other polices — field test of C-o-D interface
  - Reliable multicast
  - Variegated multicast trees
- Analysis of contention at output queues
- Inter-controller communication — *Composition of policies*
- Simulation of Control on demand systems (AN-Sim)
  - Wide-area dynamics of C-o-D
  - Behavior of C-o-D flows in heterogeneous networks