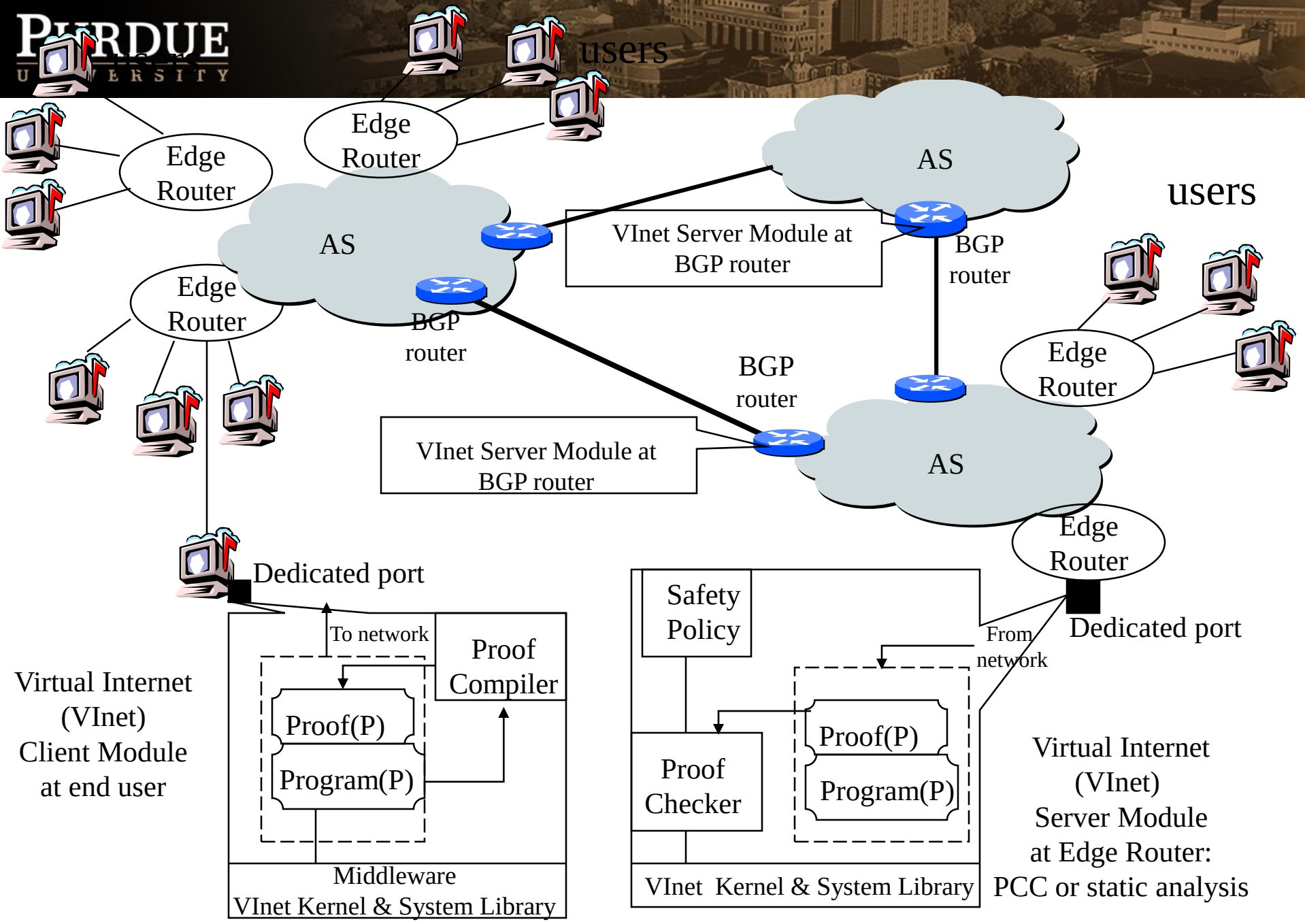
A Secure Programming Paradigm for Internet Virtualization

Sonia Fahmy
Department of Computer Science
Purdue University
http://www.cs.purdue.edu/~fahmy/

Joint work with Ana Milanova, David Musser, Bulent Yener (Rensselaer Polytechnic Institute)

Motivation

- DDoS attacks, malware, spam, ... are rapidly increasing
- Overlay and other end system solutions are insufficient and inefficient
- New applications require network support
- Active networks efforts failed due to ***security*** and ***scalability*** concerns
- Can we resolve these security and scalability concerns, and allow users to customize and virtualize the Internet?



Virtual Internet (VInet) Client Module at end user

Virtual Internet (VInet) Server Module at Edge Router: PCC or static analysis

Research Agenda

- Programmability
  - A library of packet manipulation routines
  - Language to compose routines
- Verification technology
  - Extend PCC with a safety policy (to check security and scalability)
  - Generate compact, composable, easy-to-check proofs
- Client-server protocols and scheduling
  - Securely transmit program, proof, payment
- Execution environment
  - Constrain programmability and use adaptive optimizations
- Emulation technology
  - Study performance, security, scalability limits, reliability, placement, partial deployment, under realistic scenarios

Packet Manipulation Routines

- DISCARD(Packets)
- GENERATE(SrcIP,DestIP,Content)
- GROUP_BY_DESTINATION(Packets,DestIP): Packets
- GROUP_BY_DESTINATION(Packets): Packets(i), DestIP(i)
- GROUP_BY_SOURCE(Packets,SrcIP): Packets
- GROUP_BY_SOURCE(Packets): Packets(i),SrcIP(i)
- GROUP_BY_CONTENT(Packets): DestIPs, Content
- CONATINS_STRING(Packets,String): Boolean

Example: Virtual Firewall

- LIFETIME = 2 days ← f(Complexity,lifetime) ≤ $Cost^{up}$
- MyPackets = GROUP_BY_DESTINATION(AllPackets, THIS_IP);
- BadPackets = GROUP_BY_SOURCE(MyPackets, SrcIP);
- DISCARD(BadPackets);

Example: Virtual Spam Filter

- LIFETIME = 5 days
- MyPackets = GROUP_BY_DESTINATION(AllPackets,THIS_IP);
- SourceIP(i),Packets(i) = GROUP_BY_SOURCE(MyPackets);
- Foreach i do
  - If CONTAINS_STRING(Packets(i),Word1) and CONTAINS_STRING(Packets(i),Word2)
    - DISCARD(Packets(i);

Example: Multicast

- LIFETIME = 5 hours
- MyPackets = GROUP_BY_DESTINATION(AllPackets, THIS_SUBNET_IP);
- Broadcast = GROUP_BY_SOURCE(MyPackets,BroadcastIP);
- Destinations,Content = GROUP_BY_CONTENT(Broadcast);
- GENERATE(BroadcastIP,THIS_IP,Destinations);
- GENERATE(BroadcastIP,THIS_IP,Content);
- DISCARD(Broadcast);

Example Safety Policy

```
(domain IP)
(domain Content)
(datatype Packets (Packet IP IP Content))
(declare EndUser ((IP) → Boolean))
(declare Owns ((IP IP) → Boolean))
(declare CanDiscard ((IP Packets) → Boolean))
(declare Discard (((List-Of Packets)) → (List-Of Packets)))
(domain NetworkState)

(declare AnyFilterWith
      ((IP (List-Of Packets) (List-Of Packets))
      → NetworkState))
(defi ne EndUser-owns-only-own-IP
 (forall ?ThisIP
  (if (EndUser ?ThisIP)
   (forall ?IP ((Owns ?ThisIP ?IP) iff (?IP = ?ThisIP))))))
(defi ne CanDiscard-only-owned-IPs
 (forall ?ThisIP ?From ?To ?Content
  ((CanDiscard ?ThisIP (Packet ?From ?To ?Content))
   iff (Owns ?ThisIP ?To))))
(defi ne Discard-axiom
 (forall ?ThisIP ?From ?To ?Content ?Traffi c
  ((Discard (Cons (Packet ?From ?To ?Content) ?Traffi c))
   = ?Traffi c)))
(assert EndUser-owns-only-own-IP
  CanDiscard-only-owned-IPs
  Discard-axiom)
```

```
##########################################
A safety policy required for particular end-users
##########################################

(defi ne If-discard-not-allowed-then-does-not-happen
 (forall ?ThisIP ?From ?To ?Content
  (if (not (CanDiscard ?ThisIP
                (Packet ?From ?To ?Content)))
   (forall ?Incoming ?Outgoing
    ((AnyFilterWith ?ThisIP
      (Cons (Packet ?From ?To ?Content)
            ?Incoming)
      ?Outgoing)
     = (AnyFilterWith ?ThisIP
      ?Incoming
      (Cons (Packet ?From ?To ?Content)
          ?Outgoing)))))))
```

Fig. 2. Formal Requirements and Safety Policy for Packet Filtering Programs.

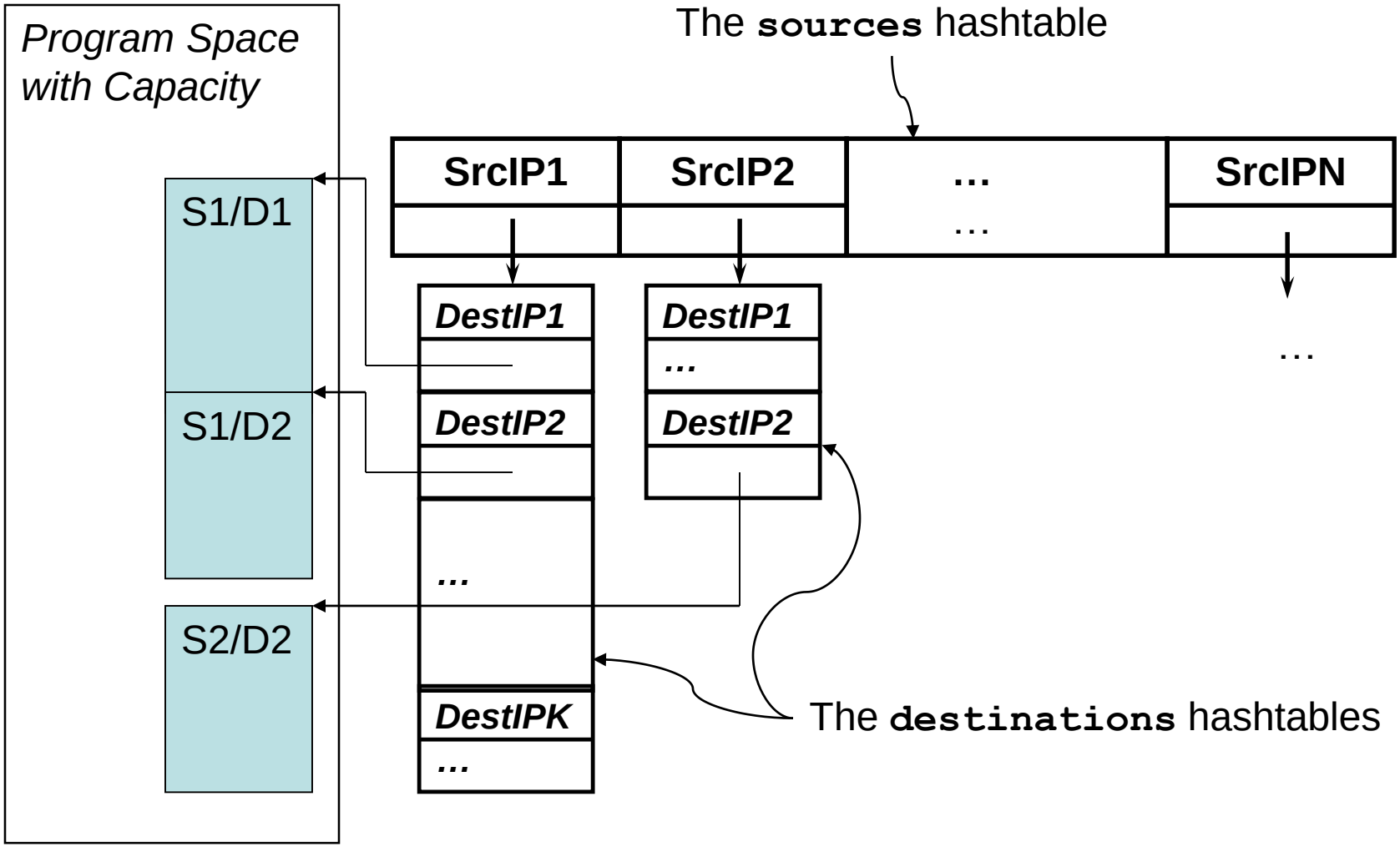
Proof of Safety Policy

```
(declare IP1 IP)
(assert (EndUser IP1))
(declare Filter1
 (((List-Of Packets) (List-Of Packets)) → NetworkState))
(define Filter1-relation-to-AnyFilterWith
 (forall ?Incoming ?Outgoing
  ((Filter1 ?Incoming ?Outgoing)
   = (AnyFilterWith IP1 ?Incoming ?Outgoing))))
(assert Filter1-relation-to-AnyFilterWith)
(define EndUserRule
 (forall ?From ?To ?Content ?Incoming ?Outgoing
  (if (not (?To = IP1))
   ((Filter1 (Cons (Packet ?From ?To ?Content)
                   ?Incoming)
             ?Outgoing)
    = (Filter1 ?Incoming
               (Cons (Packet ?From ?To ?Content)
                     ?Outgoing))))))
(assert EndUserRule)
##############################################
(define EndUser-IP1-observes-policy
 (forall ?From ?To ?Content
  (if (not (CanDiscard IP1 (Packet ?From ?To ?Content)))
   (forall ?Incoming ?Outgoing
    ((AnyFilterWith IP1
          (Cons (Packet ?From ?To ?Content)
                ?Incoming)
          ?Outgoing)
     = (AnyFilterWith IP1
          ?Incoming
          (Cons (Packet ?From ?To ?Content)
                ?Outgoing)))))))
```

```
(!(conclude EndUser-IP1-observes-policy)
 (pick-any From To Content
  (assume (not (CanDiscard IP1 (Packet From To Content)))
   (pick-any In Out
    (!cases
     (assume (To = IP1)
      (!by-contradiction
       (assume (not ((AnyFilterWith IP1
                          (Cons (Packet From To Content) In) Out)
                     = (AnyFilterWith IP1 In
                          (Cons (Packet From To Content) Out))))
        (dseq
         (!(conclude (Owns IP1 To))
          (!right-instance
           (!mp (!uspec* EndUser-owns-only-own-IP [IP1])
                (EndUser IP1)) [To]))
          (!absurd
           (!right-instance CanDiscard-only-owned-IPs
                            [IP1 From To Content])
           (not (CanDiscard IP1 (Packet From To Content))))))))
     (assume (not (To = IP1))
      (dseq
       (!setup left (AnyFilterWith IP1
                          (Cons (Packet From To Content) In) Out))
       (!expand left (Filter1 (Cons (Packet From To Content)
                                    In) Out)
                     Filter1-relation-to-AnyFilterWith)
       (!reduce left (Filter1 In (Cons (Packet From To Content)
                                       Out)) EndUserRule)
       (!reduce left (AnyFilterWith IP1
                          In (Cons (Packet From To Content) Out))
                     Filter1-relation-to-AnyFilterWith)))))))))))
```

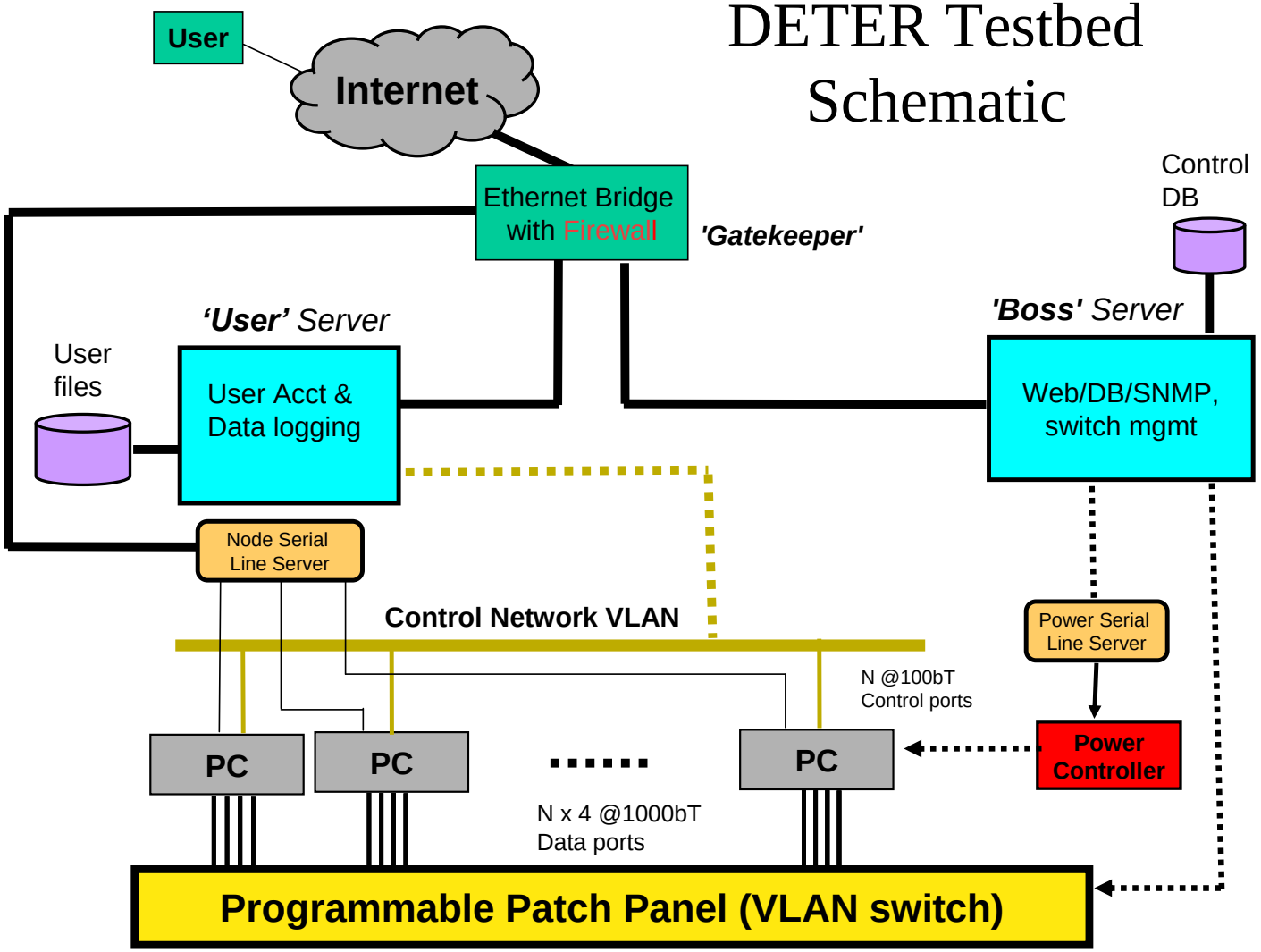
Fig. 3. A model of a filtering program and the proof of the Safety Policy in Figure 2.

Dynamic Execution Environment



Program classes and adaptive optimizations

DETER Testbed Schematic



Source: DETER USC-ISI team; DETER is based on U. of Utah Emulab

Related Work

- Active networks, e.g., ANTS, PLAN, … http://nms.lcs.mit.edu/activeware/
  - Need useful programmability, while balancing security and scalability
- Liquid software project http://www.cs.arizona.edu/liquid/
  - We restrict programs to compositions of well-defined routines, and reuse lemma library
- Model checking
  - We place less responsibility on servers and use a higher level programming language
- Overlay networks
  - We allow efficient, network-level, operations
- Emulation, e.g., Click modular routers
  - Does not address remote programmability, security, or scalability

Conclusions and Planned Work

- Virtualizing the Internet enables several exciting services and a better Internet
- Security and scalability constraints and a library of lemmas enable efficient and secure virtualization using PCC
- Planned research on programmability, verification technology, client-server protocols and scheduling, execution environment, and emulation technology