

# Containing Runaway Network Protocols with Icarus

---

David Wetherall and Andrew Whitaker  
University of Washington, CSE Dept.

DARPA ANets PI Meet, Jackson Hole, June 2001.

# But First, Announcing ANTS2.0!

---

- Available on UW and Utah websites
- Primarily a merge of ANTSR and ANTS1.3
- With thanks to Utah for heavy lifting, and ISI too.
- Features:
  - NodeOS/EE separation, per protocol resource controls, per protocol/application security subsystem, DANTE support, better documentation, nearly unchanged API, many bug fixes ...

# Introduction

---

- What happens if an new protocol is buggy?
- Many mechanisms to protect a single node
- Few mechanisms to protect the network
  - Hop counts (TTLs) or resource quotas
  - Fair Queuing or bandwidth partitioning
- We need more tools here ...

# This Talk

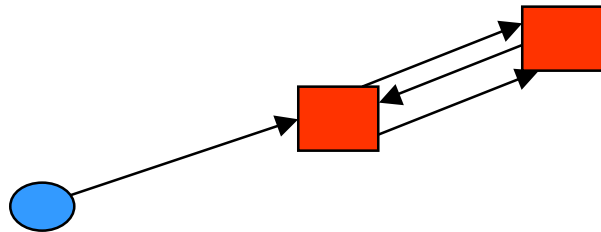
---

1. Runaway protocols and TTLs
2. Icarus, our protection mechanism

# Defining Runaway Protocols

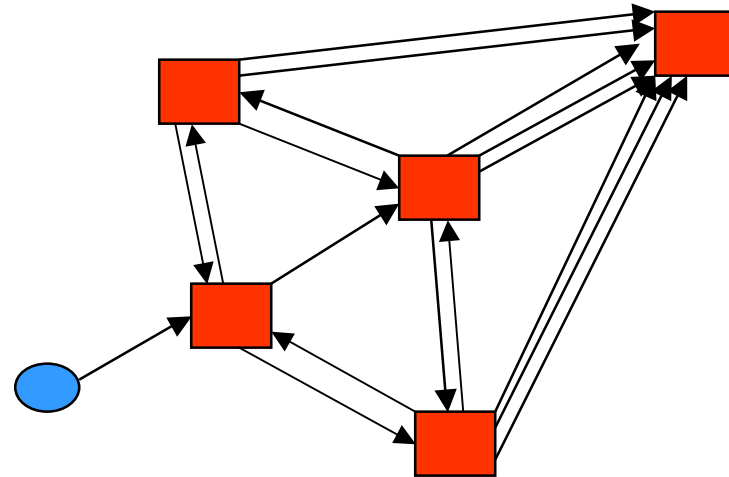
---

- Loops:



- Routing loops
- name server recursion
- HTTP redirection loops,
- mobile IP forwarding loops

- Replica collisions:



- Multicast loops
- buggy flooding

- Treat hops as a rough measure of resource usage

# A Note on Applicability

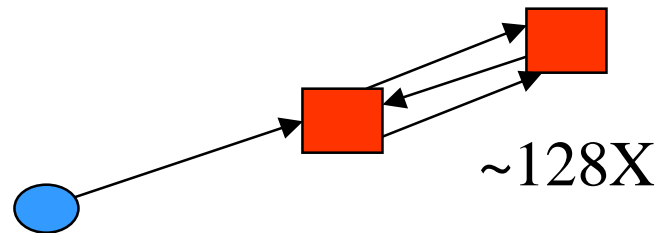
---

- Our work was undertaken in the AN space
  - Network (IP packet-level) protocols
  - Untrusted mobile code
- Our techniques are more broadly applicable
  - Flows/connections and application level routing
  - Experimentation with trusted code
- Overlays, Peer-to-Peer or IOS-NG anyone?

# TTL-based Protection

---

- A looping packet can concentrate packet activity in a small area, saturating the available bandwidth.

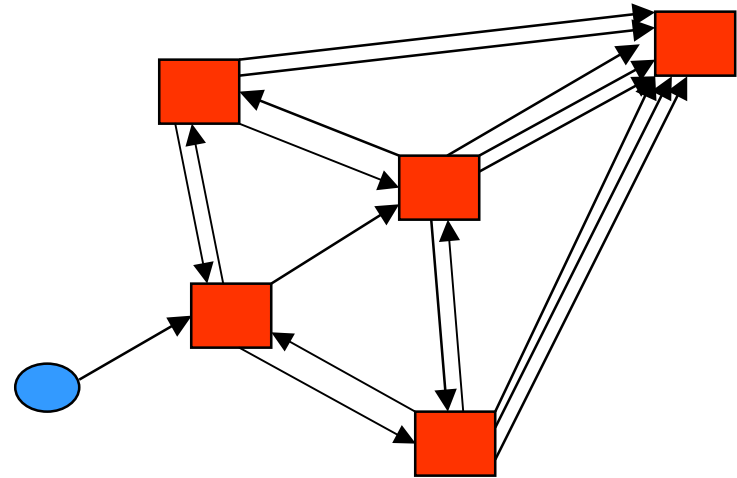


- For  $TTL \leq 255$  could have  $\sim 128X$  blowup.
- This is bad, but of little import if loops are rare

# TTLs and Multicast

---

- Add “multicast” and loops can be frightful!
- For  $TTL \leq 255$  could have  $\sim 2^{128}X$  blowup.
- For “strict”  $TTL \leq 255$  could have 128X blowup
- Plus “strict” TTLs are difficult to use and require large values for large groups ...



# Fair Queuing Protection

---

- Can ensure each user gets their fair share at nodes, and so will limit blowup. Useful.
- But blowup means one user can crowd others with few resources ... denial of service
- And there is no feedback about faults

# Icarus

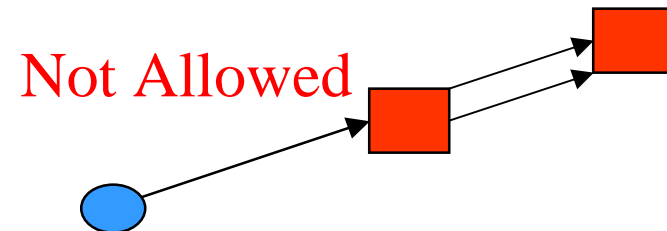
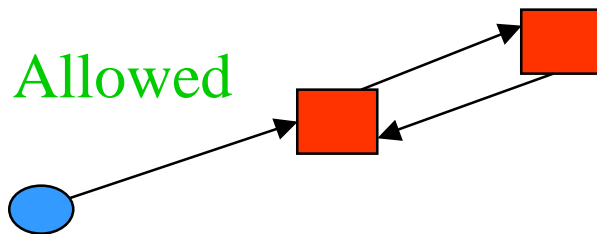
---

- A framework for containing runaway protocols
  - Prevent them from consuming too many resources
- Goal is to improve on TTL but still be efficient
  - Little header space, computation and node state
- Broadly useful rather than strictly fair/correct
  - Works for many different protocols/environments

# Icarus Containment Check

---

- Safe protocols adhere to *generalized loop freedom*: a packet (and descendants) traverses a link at most once in either direction:

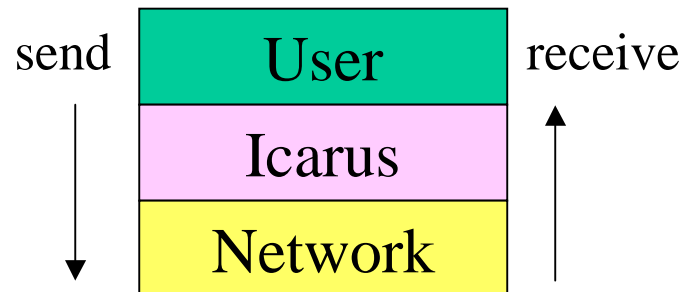


- protocols that violate this are said to be “runaway”
- subsumes unicast loops and replica collisions
- captures reasonable usage for a range of protocols

# Icarus Protocol Framework

---

- Transparently interpose a logical protocol layer beneath the user protocol:



- The Icarus layer implements a *containment policy*:
  - A *containment algorithm* performs packet pre- and post-processing. Icarus protected state can be stored in the packet header and at the network nodes.

# Icarus Containment Classes

---

		<b>Routing</b>		
		restricted	arbitrary	flow
<b>Replication</b>	unicast	IP	anycast	mobile IP
	multicast	DM-PIM	link state flooding	SM-PIM

Each protocol class can have its own containment policy

# ANTS Implementation

---

- Separate containment policies from user protocols
  - The Icarus containment policies are represented as subclasses of `Capsule` that implement interfaces

```
class RestrictedUnicastCapsule extends Capsule
implements RestrictedUnicast
```
  - User protocols join a protocol class by extending an Icarus capsule class.

```
class MyCapsule extends
RestrictedUnicastCapsule
```

# Restricted Unicast

---

## Routing

	restricted	arbitrary	flow
unicast	IP	anycast	mobile IP
multicast	DM-PIM	link state flooding	SM-PIM

- Unicast protocols that use default routes

# Restricted Unicast Policy

---

- Let packet route towards a fixed destination
- Assume routing protocol supplies stable loop-free routes and use a TTL field to break up transient loops
  - It's worked well in the past ...
- No overhead beyond that of IP

# Arbitrary Unicast

---

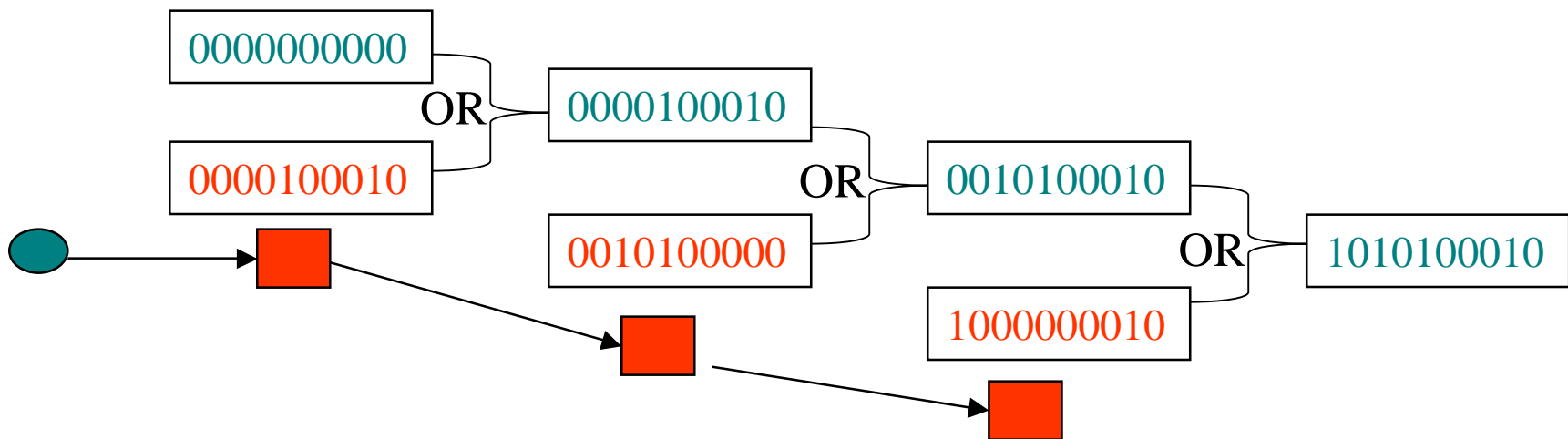
		Routing		
		restricted	arbitrary	flow
unicast		IP	anycast	mobile IP
multicast		DM-PIM	link state flooding	SM-PIM

- Unicast protocols that use custom routes

# Arbitrary Unicast Policy

---

- Use a bloom filter field to track a packet's path:



- If the bloom filter changes, the packet is not looping, if it doesn't change then it *might* be looping

# Dealing With False Positives

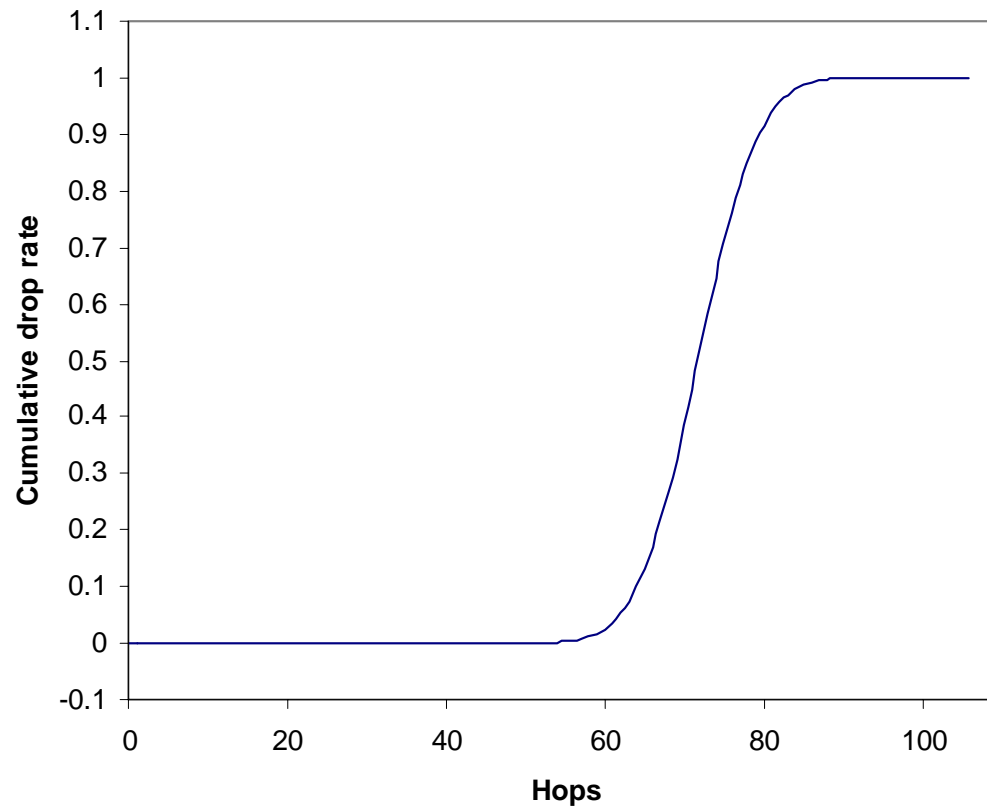
---

- We could increase the size of the bloom filter
  - requires linear space
- Instead, allow few collisions before dropping.
  - *reprises*: do not reset bloom filter
  - *failures*: reset the the bloom filter
- But, we lose detection accuracy.

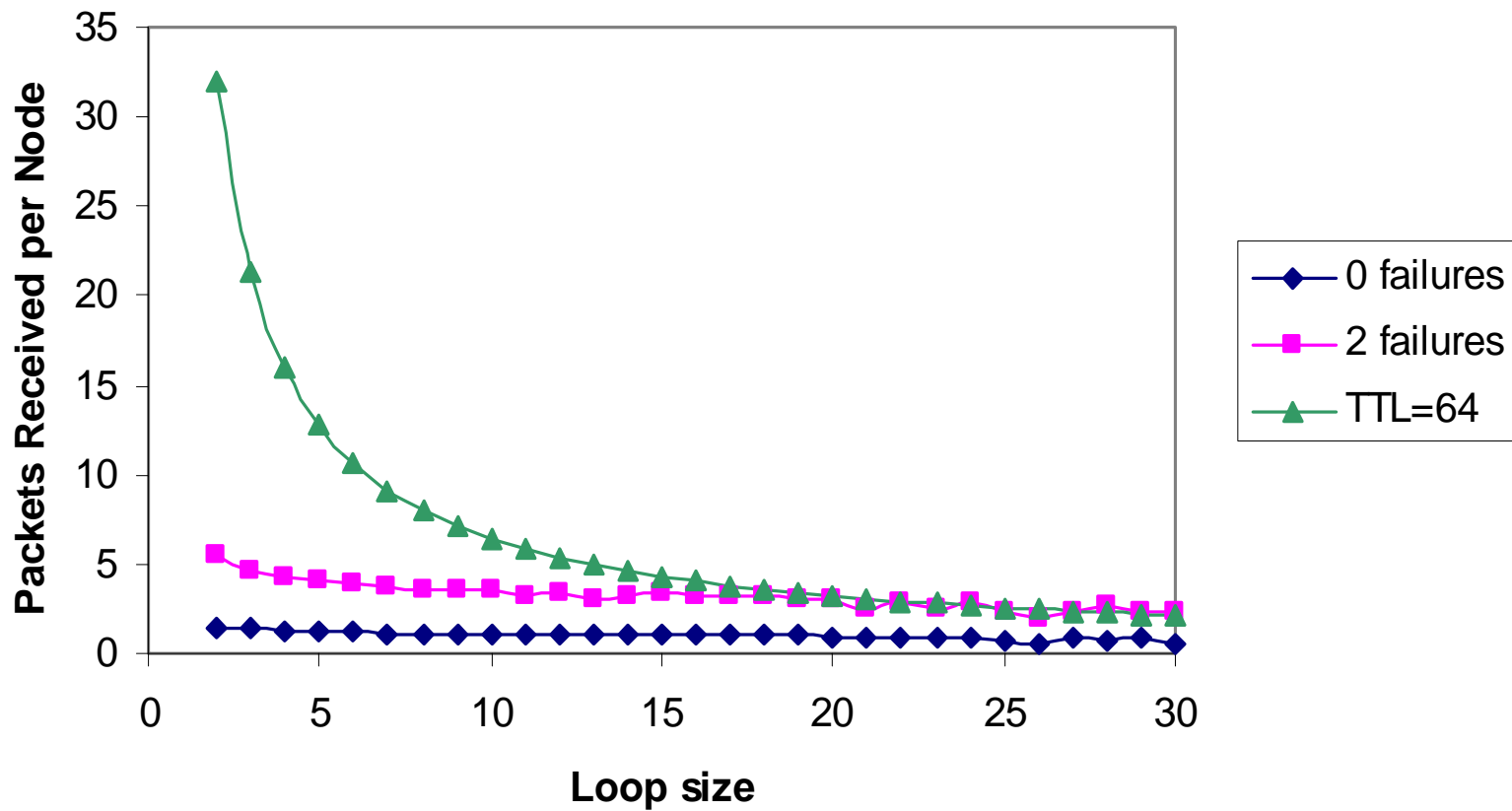
# False Positives

---

n=64  
m=4  
2 failures  
2 reprieves



# Loop Detection Accuracy



# Other Containment Classes

---

- Flow-based
  - Use bloom checks while establishing flow state
  - No subsequent checks when using flow
  - For multicast, merge (join) unicast flows
- Single packet multicast
  - RPF constraints or per packet flood state
- See the paper for details

# Conclusions

---

- Icarus provides lightweight safety checks that prevent runaway protocols
  - A precise TTL that checks a strong property
  - Minimal burden for the protocol developer
- Icarus is suited to active networks, application level overlays, and “IOS-NG”