



# **SURYA: Solaris Forwarding Performance Project**

**Sangeeta Misra@sun.com**  
**Solaris Core Networking**



# Surya project goals

- Improve IPv4 forwarding throughput and scalability through:
  - Improved IPv4 forwarding table lookup scheme
  - Streamlined IPv4 forwarding path

# Overview of topics

- Concepts of Solaris IP forwarding implementation
- Modifications to Solaris IPv4 lookup scheme
- Streamlining of IPv4 forwarding path
- Results
- Future work
- For more information...

# Concepts of Solaris forwarding implementation

- *Route entry*
  - Describes how to reach destination host or network
- *Cache table*
  - Contains routes to hosts only
  - Route entries are generated when IP receives a outbound packet that is to use that route.
  - Route entries contain more information than those in forwarding table.

# Concepts of Solaris forwarding implementation (contd.)

- *Forwarding table*
  - Contains routes to hosts and networks
  - Route entries are generated manually, by routing daemons or by the system
  - Route entries do not have link-layer information
  - Goal of forwarding table lookup is to find the most specific address that matches a given destination (dest).

# Pre-Surya IPv4 forwarding lookup scheme and its limitation

- Per-netmask hash table with each entry pointing to a bucket that is associated to a linked-list of route entries
- Linear search through the linked-list of route entries of a bucket was expensive
- Forwarding lookup scheme did not scale

# Surya's new lookup scheme

- Investigated alternative schemes:
  - Address-directed forwarding lookup schemes
  - Radix tree scheme
- Selected Radix tree and ported FreeBSD's Radix tree implementation (version 1.36.2.1)

# Surya's new lookup scheme: Radix tree

- The dest address is considered as a sequence of bits
- The internal nodes of the tree represent a bit position to test (ON = right branch, OFF = left branch)
- The leaves of the tree represent a host or a network address

# Surya's new lookup scheme: Radix tree (contd.)

- If one ends up at the leaf, one does a masking operation to see if there is a match
- If there is no match, one backtracks up the tree for a more general match (ie default route)

# Comparison between the two schemes

- Worst-case search length for Radix tree scheme is the number of bits in the address
- Worst-case search length for pre-Surya scheme was the number of routes.
- Example: For 2000 route entries, that means 32 compares for Radix versus 2000 compares

# Radix Tree algorithm

- References

- Skowler, K 1991. "A Tree-Based Packet Routing Table for Berkeley Unix," Proceedings of 1991 Winter USENIX Conference, pp.93-99
- Wright, G.R. and Stevens, W.R. 1995. "TCP/IP Illustrated Volume 2: The Implementation." Addison-Wesley Reading, Mass. pp 559-599.

# Goals for Forwarding path re-architecture

- Optimizing IPv4 forwarding path
- Introduction of ***incomplete*** route entries in cache table through forwarding path
- Prevent creation of cache table route entries for off-link dests for forwarding purposes

# Pre-Surya forwarding path logic

- Search cache table for route entry of next hop G. If not found, complete ARP resolution for it.
- Create cache table route entry of next hop G
- Create cache table route entry for off-link dest D
- Send packet out using cache table route entry of dest D

# Limitations of Pre-Surya forwarding path logic

- Creation of cache table route entry for every off-link dest through forwarding path is undesirable
- In general, delaying insertion of cache table route entry until after completion of link-layer address resolution is undesirable

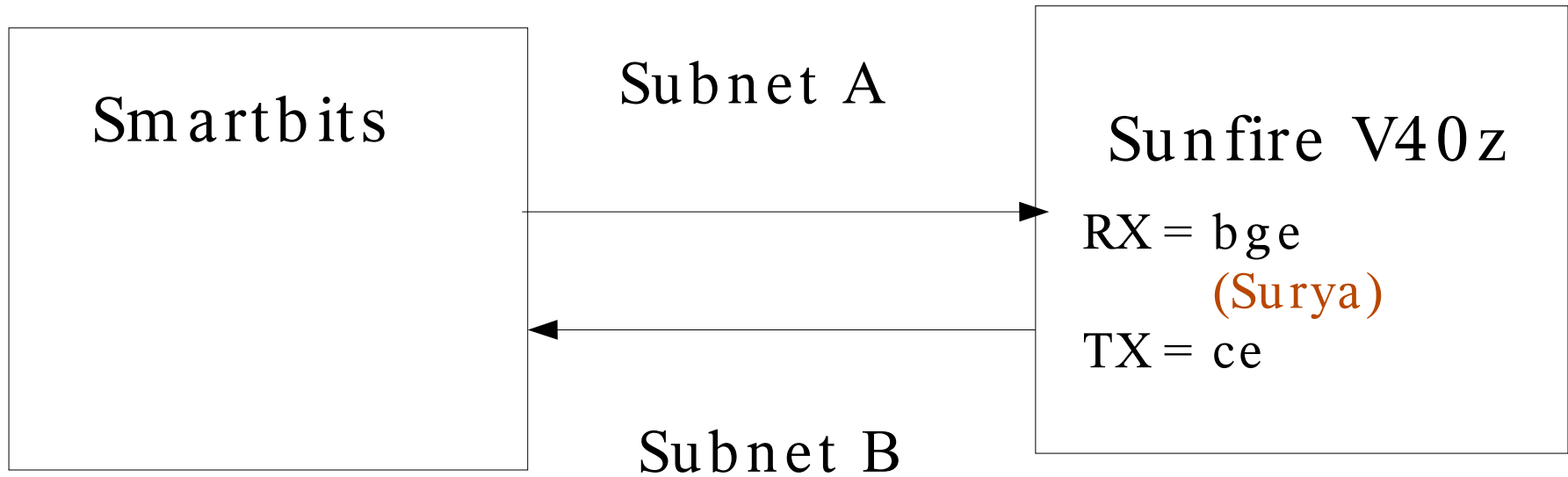
# Surya's re-architected forwarding path

- Search cache table for route entry for next hop G. If not found, create it, mark it as ***incomplete*** and insert into the cache table
- Complete all IP processing and queue the packet into the next hop cache table entry's internal queue.

# Surya's re-architected forwarding path (contd.)

- Send out ARP request for link-layer address resolution of the next hop G
- Upon ARP resolution completion, mark next hop's cache table entry as **complete** and send out all packets that were queued in its internal queue

# Forwarding throughput test setup



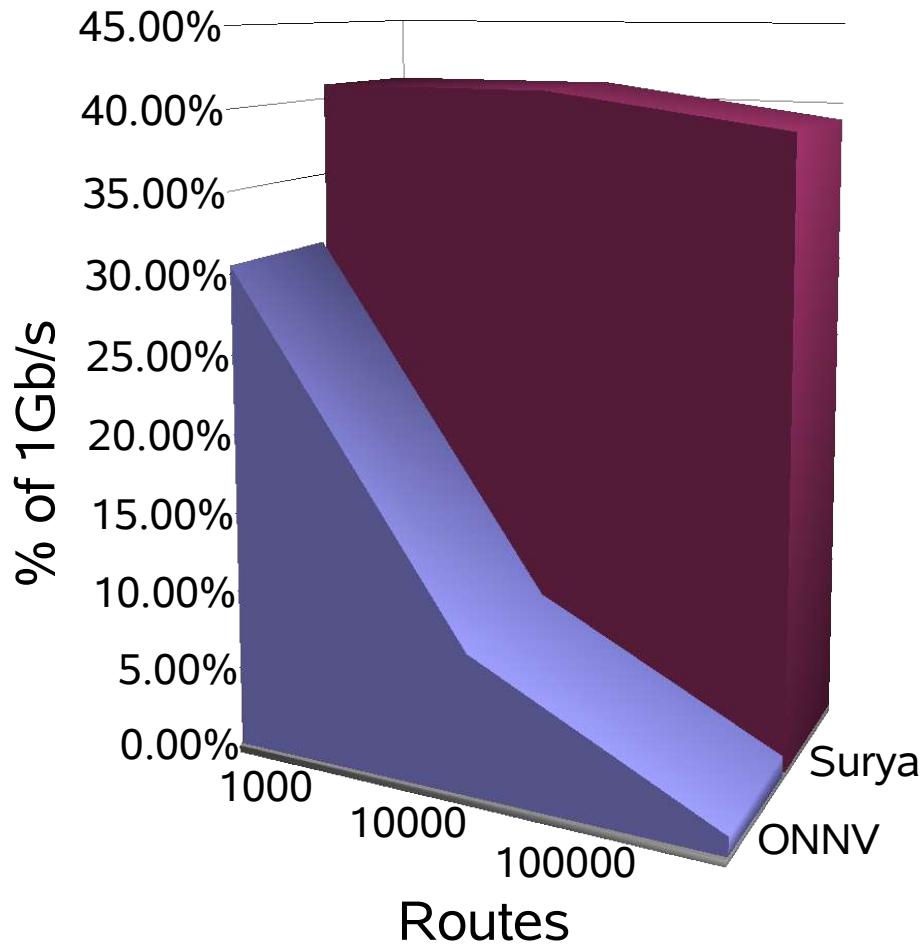
Packet size = 128 bytes

Acceptable frame loss = 0.5 %

Data Packet Protocol = IP Exp.protocol 253

# Surya IP forwarding performance

## Throughput vs Number of routes



# Future work

- Analysis of forwarding throughput bottlenecks below IP
- Other routing related features
  - ECMP support
  - Address IPv6 forwarding throughput and scalability
  - Policy-based routing
- IP data path re-factoring
- Quagga routing daemon suite

# For more information

- [www.opensolaris.org/os/community/appliances/announcements/solnetapp](http://www.opensolaris.org/os/community/appliances/announcements/solnetapp)
- Interested in Solaris routing related projects/features?  
Email to: [networking-discuss@opensolaris.org](mailto:networking-discuss@opensolaris.org) or  
[sangeeta.misra@sun.com](mailto:sangeeta.misra@sun.com)