Improving Per Processor Memory Use of ns-3 to Enable Large Scale Simulations

WNS3 2015, Castelldefels (Barcelona), Spain

May 13, 2015

Steven Smith, David R. Jefferson
Peter D. Barnes, Jr, Sergei Nikolaev

Lawrence Livermore National Laboratory

LLNL-CONF-667822

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Removing Memory Scaling Limits In ns-3 For Very Large Simulations

- Identify memory scaling issues
- Describe algorithm and data structure changes
- Performance and scalability studies of new approach
  - Memory consumption
  - Route lookup time
  - Total execution speed
- Remaining work

Current ns-3 Memory Scaling

- Required to instantiate full network topology on all MPI ranks
- Limits problem size to memory per compute rank
  - 2GB holds ~47K ns-3 nodes
  - 32GB holds ~750K ns-3 nodes
- Alternative: Renard, *et al*. 2012
  - Federate multiple ns-3 instances
  - Inter-federate messaging implemented by ghost nodes
  - Static routing and topology had to obey a set of rules

Want a general solution, easy to implement, with automatic routing.

Why should you care about memory scaling?

- Large models, too big for one compute node
- Heavyweight nodes
  - DCE applications, DNS servers, …
  - Virtual machines
  - Core routers with large forwarding tables

Why care about automatic routing?

- When routing doesn't matter, need it to "just work"
- Traditional routing overhead is a distraction
- *Cf.* `Ipv4GlobalRoutingHelper`, `Ipv4NixVectorRouting`

Why is fully replicated topology used in ns-3?

- Automatic global naming for nodes: Node-id property of nodes
  - All nodes have an automatic integer label in [0, N)
  - Used by ns-3 attribute system, index into vector structures, labeling output, *etc*.
- NIx vector and GOD routing
  - Since all topology information is available, shortest path calculation is a local operation
  - This was the most significant hurdle to remove

Three Problems to Solve

- Memory scaling
- Global ids for configuration, output
- "Just works" routing

Memory

Global Ids

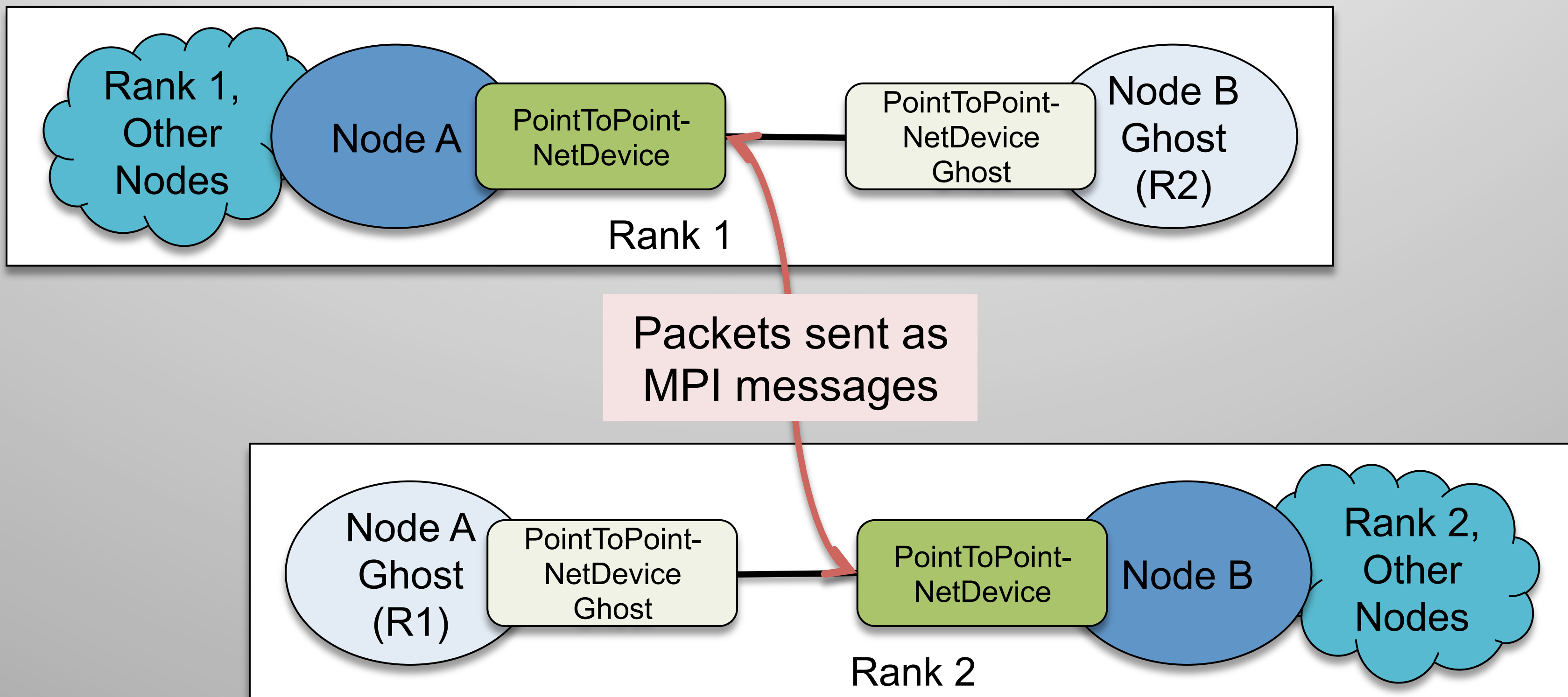
Routing

But how well does it work?

Performance

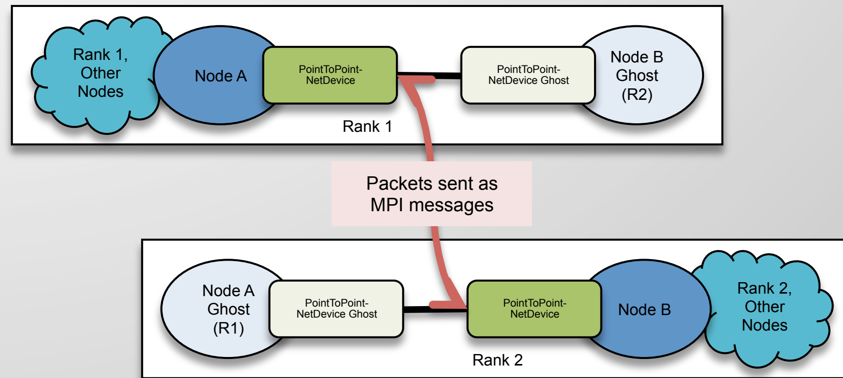
Partitioning Topology Across Compute Ranks

- Partition topology across ranks
- Ghost Node and NetDevices for cross-rank ns-3 nodes



Ghost Node and NetDevice Initialization

- Need to match ghost node and device on one rank with real indexes on another
- Require P2P IPs to be globally unique (yuck)
- Match based on IP pair



- Real Nodes exchange indexes at initialization
- Packets carry destination indexes

Matching needs work. Global ID assigned to channel would be better.

Node Global Ids

- Added Global id and retained existing Node id
  - Global id's are [0,N) unique across entire model
  - Node id is [0,n) unique to nodes on a rank
  - Automated assignment of global id's
    - AssignGlobalIds method call on GhostIdHelper class
  - Minimized code changes
  - Current implementation doesn't scale well
    - Vector of number of nodes for each rank, to enable [0,N)
- Global Id used for Config paths and some output

Alternatives to Rank-Local Node Ids + New Global Id

- Replace with single node id.
  - Globally unique on [0,N)
  - *But not [0,n) on each rank.*
    - — Impact to code is being looked at.
      - – Vectors indexed by node id become maps
    - — Haven't completed conversion for performance comparison.
- Related proposal:
  - Allow non-contiguous GIDs
  - User-assigned positive GIDs, checked for uniqueness
  - Automatically assigned negative GIDs, in blocks by rank (can be done scalably)

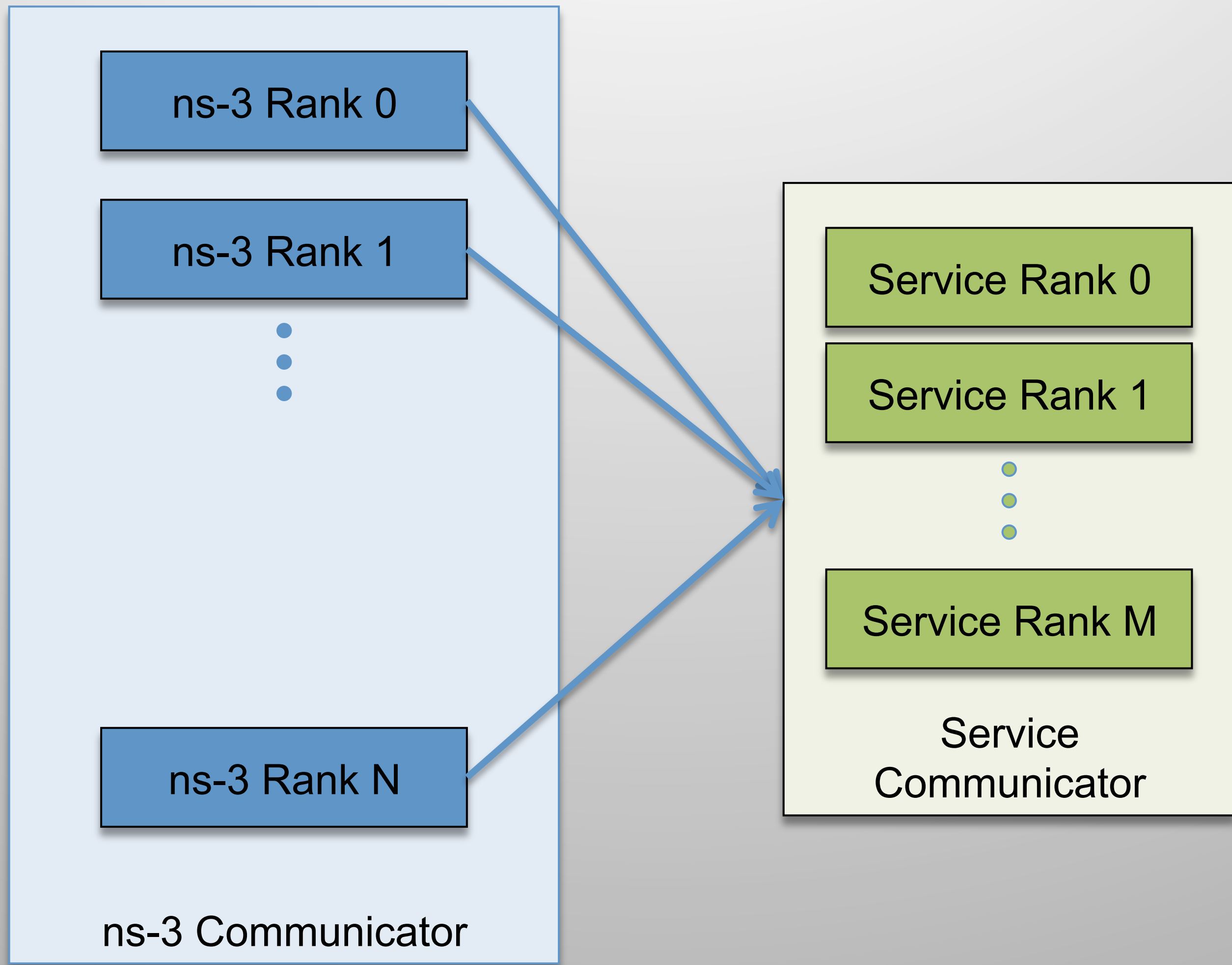
Global and node IDs need further discussion and work.

NIx Vector Routing

- NIx vector
  - Computes shortest path from source to destination
    - — Breadth first search (BFS) avg. complexity is *O*(nodes+links) work
    - — If each node computes one route need *O*(nodes * (nodes+links))
  - Store route at source as vector of NIC indices at each hop
    - — Send vector with the packet
- Distributed topology makes the shortest path impossible to compute locally.
- Routes are needed at arbitrary simulation times
  - Would be complex to integrate a distributed route calculation embedded in the parallel event processing loop

Need on-demand route lookup without involving other ns-3 ranks

Separate Parallel Simulation From Parallel Nlx Vector Route Calculation



NIx-Vector Routing Service (RS)

- Dedicated set of ranks for route service
  - Use Parallel Boost Graph Library (PBGL)
    - — Scalable, parallel breadth-first search algorithm
  - Lower memory footprint
    - — Service ranks, *s*, ns-3 ranks, *n*: *s* << *n*
  - First come first served
    - — ns-3 route queries become sequential bottleneck
  - (TBD) Support replication of route service
- When a NIx vector is required the ns-3 simulation rank queries the route service

Changes to User Scripts for Distributed NIx Vector Routing

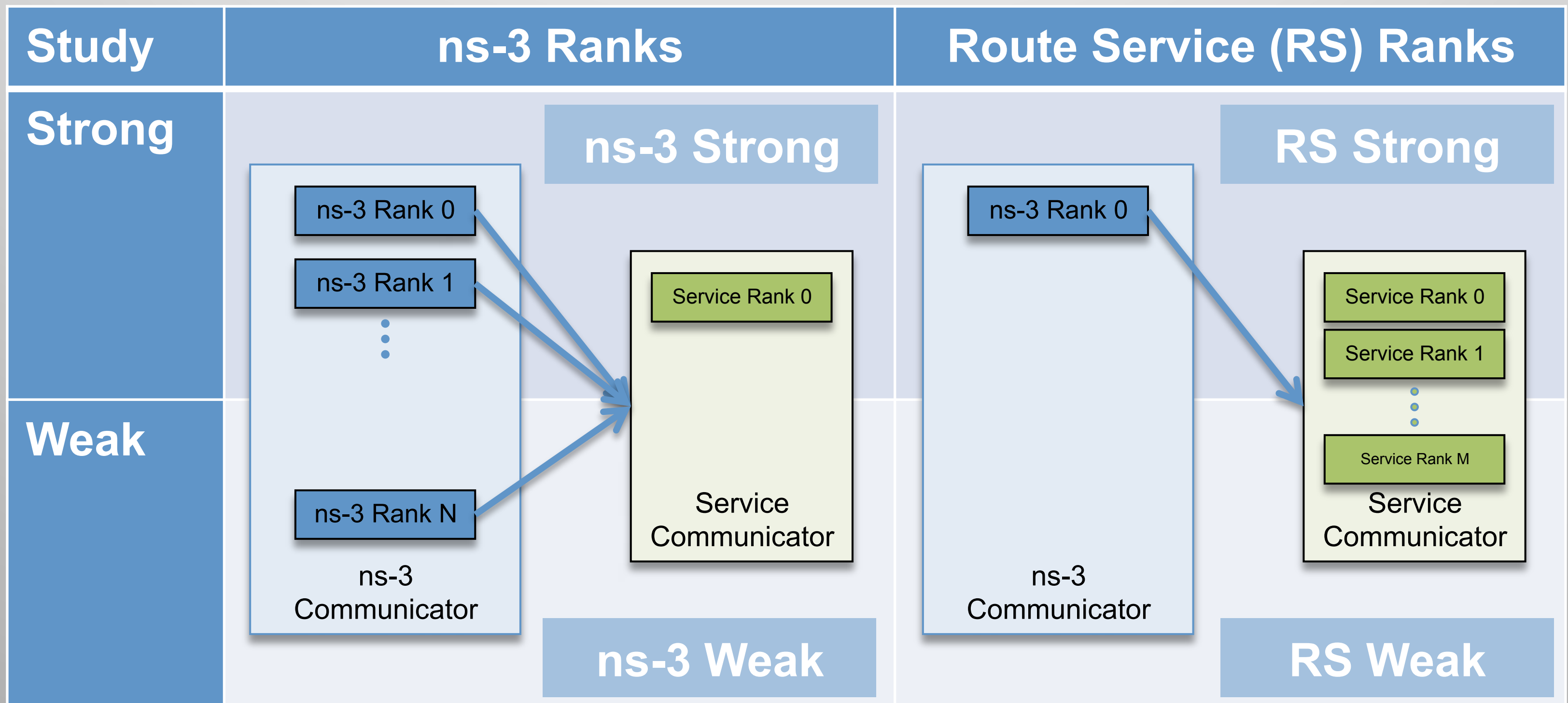
- Designate route service ranks from command line
  - Call `DistributedRouteServerInit()`, exit when it returns:

    ```
    if (DistributedRouteServerInit (argv)) exit 0;
    ```
  - Standard arguments

    ```
    --DistributedRouteServerSize=n  --DistributedRouteServers=m
    ```
- In `InternetStack::SetRoutingHelper(…)`
  - Replace `Ipv4NixVectorHelper` with `Ipv4NixVectorParallelHelper`
- Once topology defined, populate the route service
  - `Ipv4NixVectorParallelHelper::PopulateRoutingTables ()`
    - Network topology sent to route servers
    - Currently only support static topology simulations

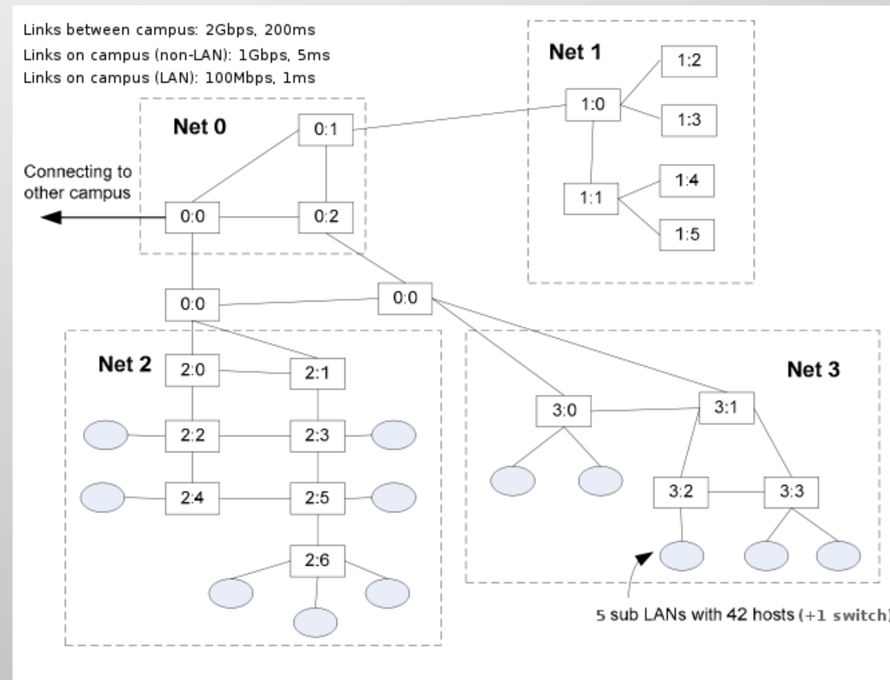
Performance Studies

- Memory scaling
- Route lookup speed
- Execution performance x4



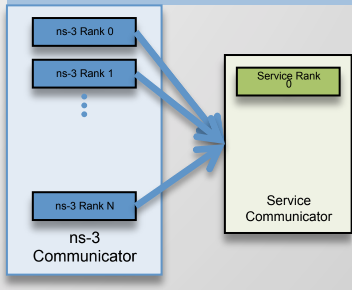
All Studies Use Modified NMS Campus Problem

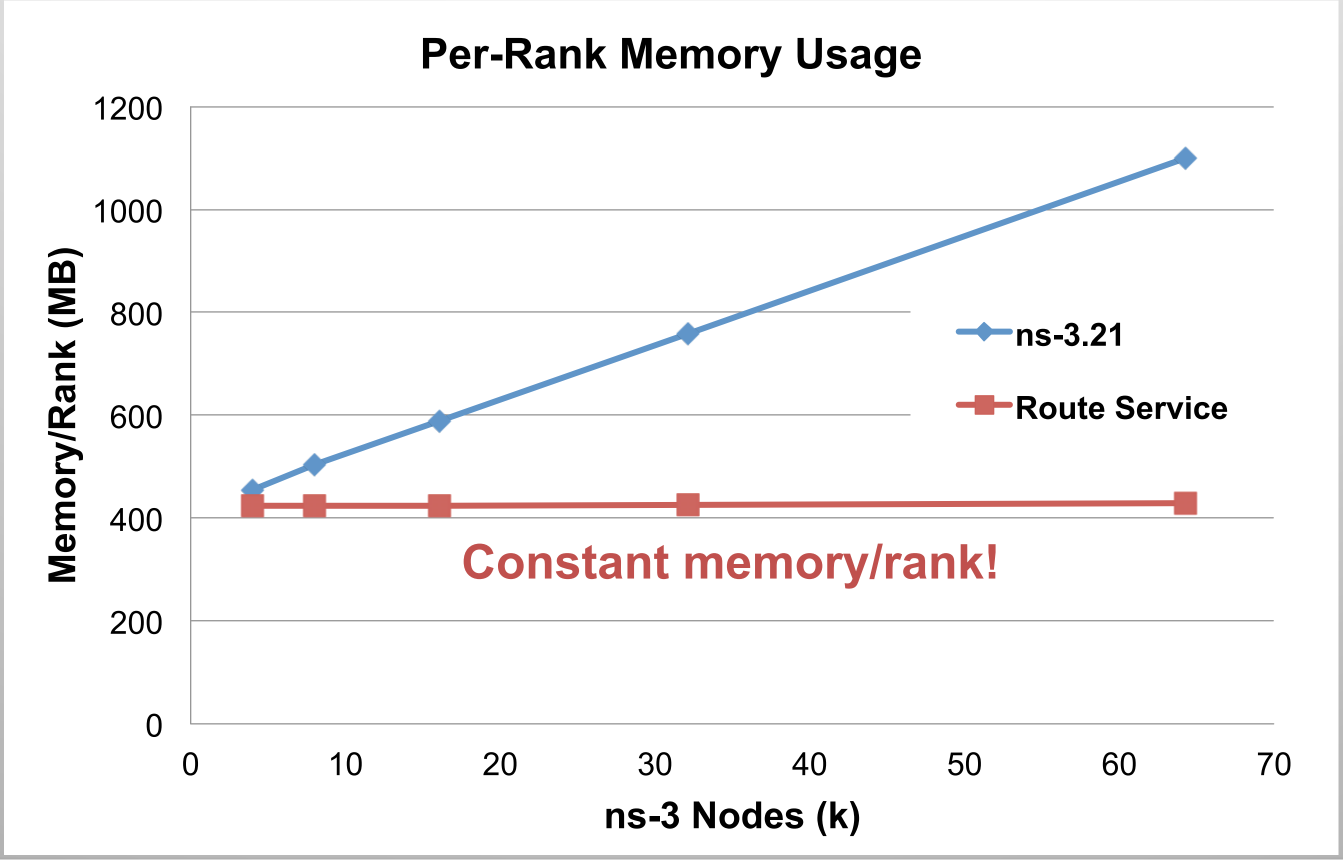
- Changed inter campus network from ring to Watts-Strogatz graph
  - *k*=4, *b*=0.05 for all runs
  - ~250 clients on campus networks
- Communication is between clients with random destinations
- (Need to merge Renard's version, much nicer)



Memory Use Per Rank

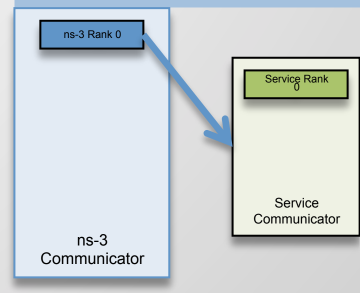
- Weak scaling study
  - Single route service rank
  - ns-3 ranks in proportion total number of ns-3 nodes (but small, 260 nodes/rank)

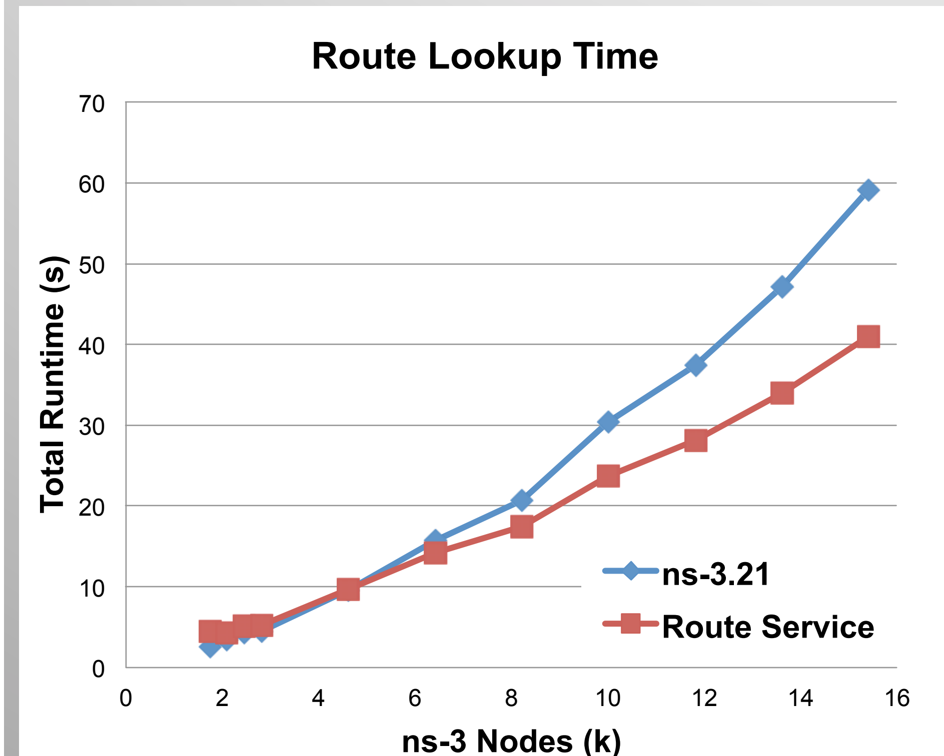


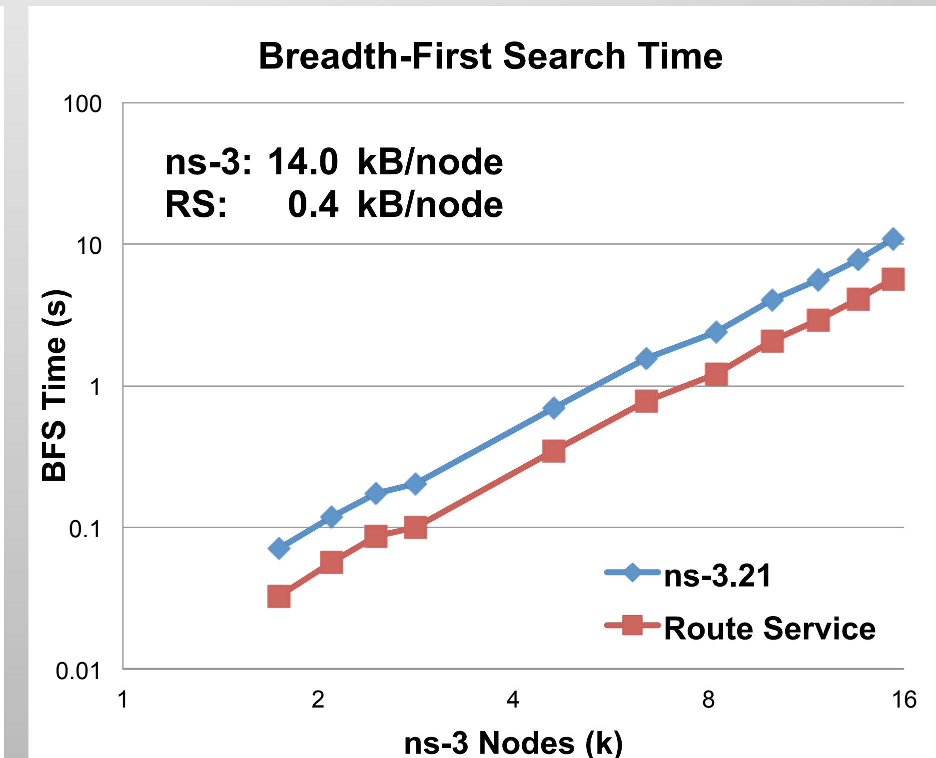


Route Lookup Performance

- Route service has larger constant overhead
  - Makes remote request to server task
  - Slower for small models
- Route service 2x faster than ns-3 BFS
  - Smaller structure – better cache utilization?
  - Faster neighbor traversal





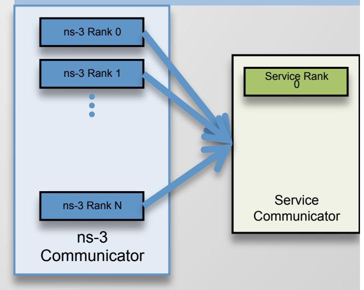


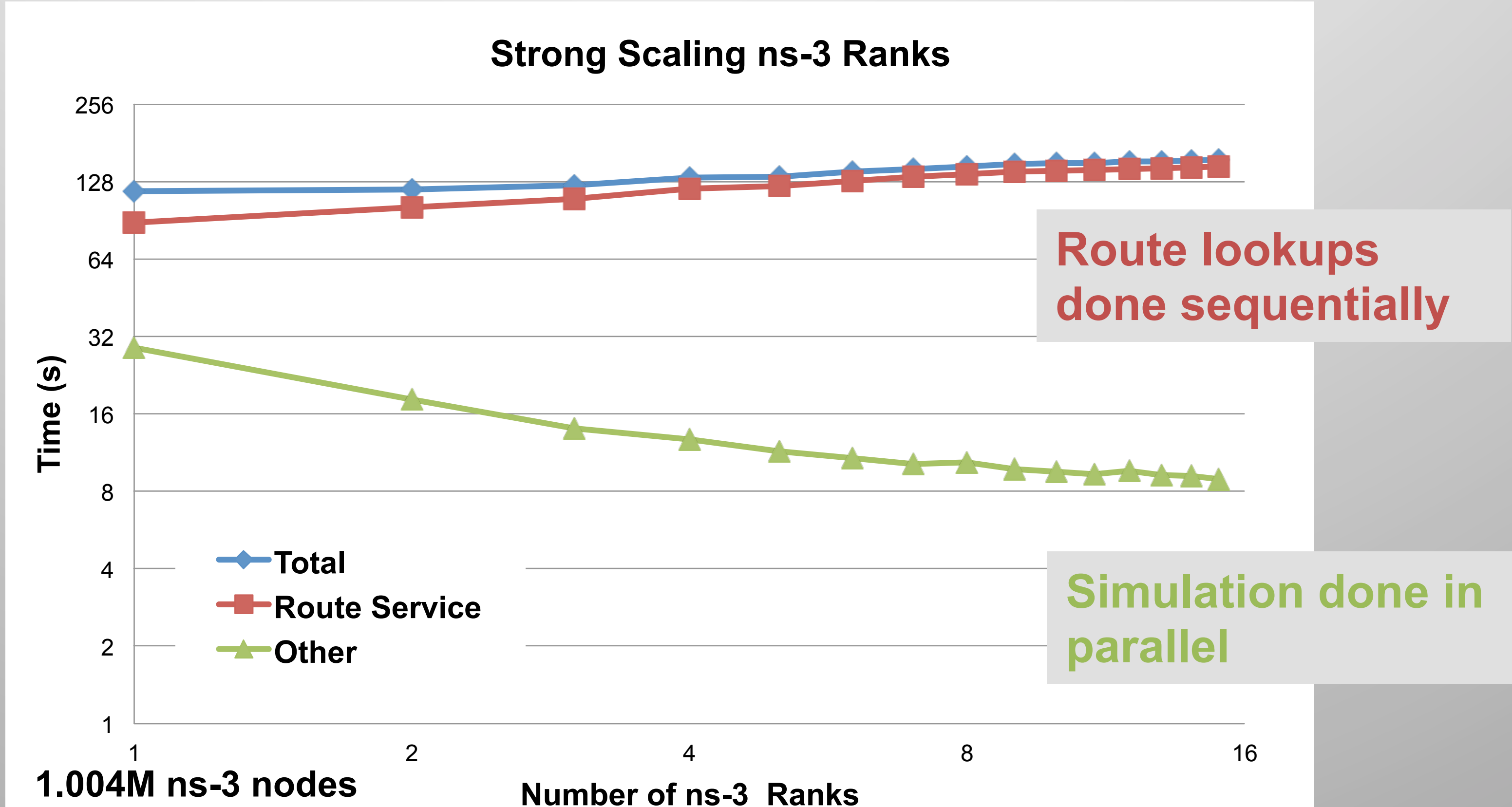
Strong Scaling ns-3 Ranks

- Expect constant time for route lookups
  - Amdahl's law in action
  - Growth in lookup time not understood

Performance

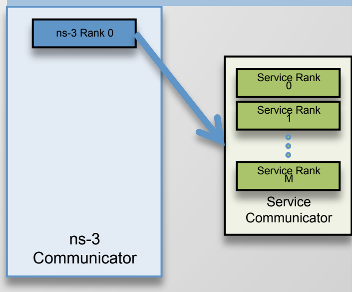
ns-3 Strong

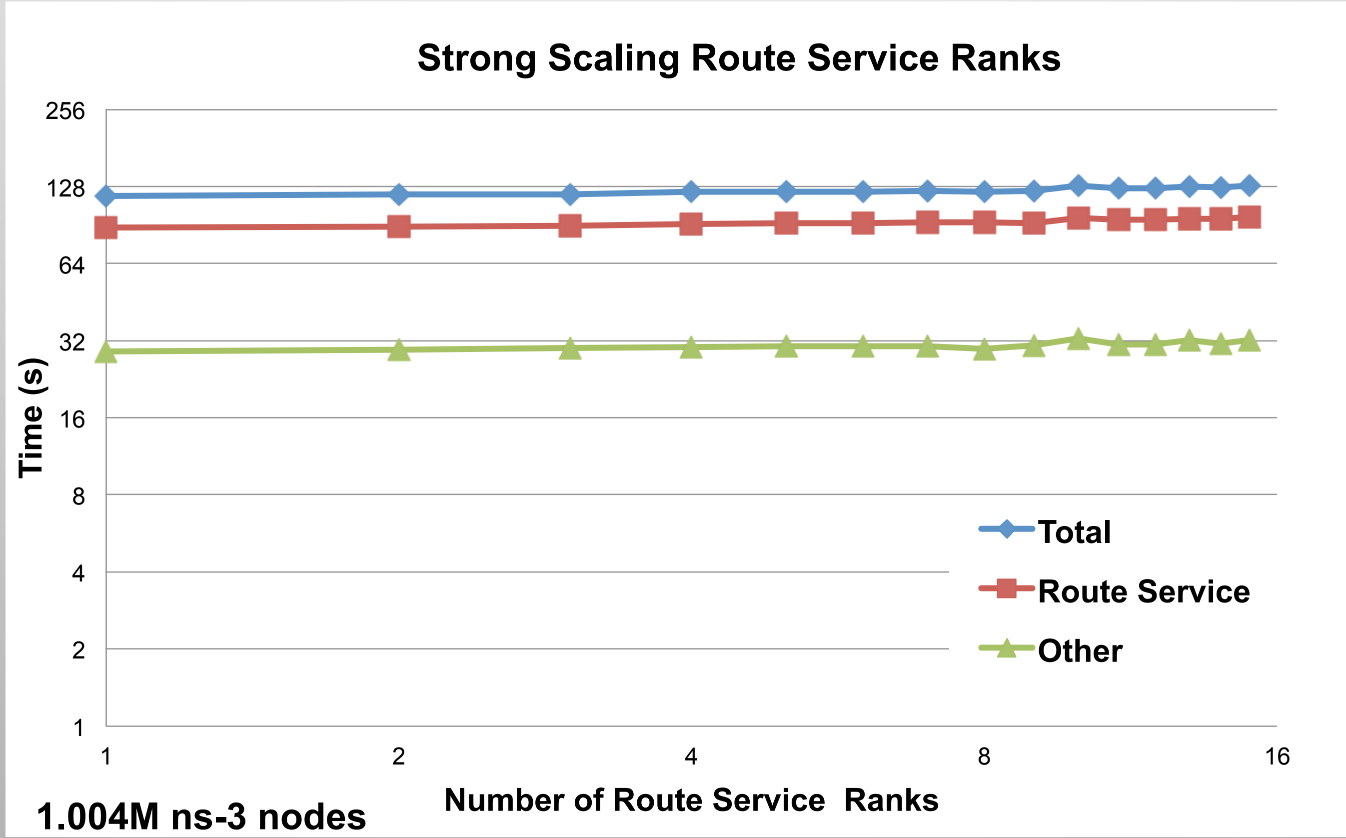




Strong Scaling Service Ranks

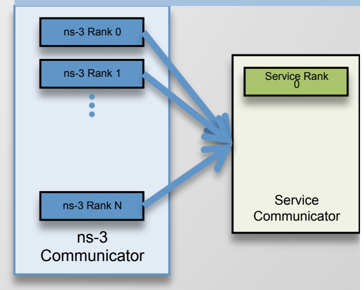
- Expect route service to get faster
  - Topology not high enough degree for PBGL
  - Average degree 1.004, PBGL scales well with 15

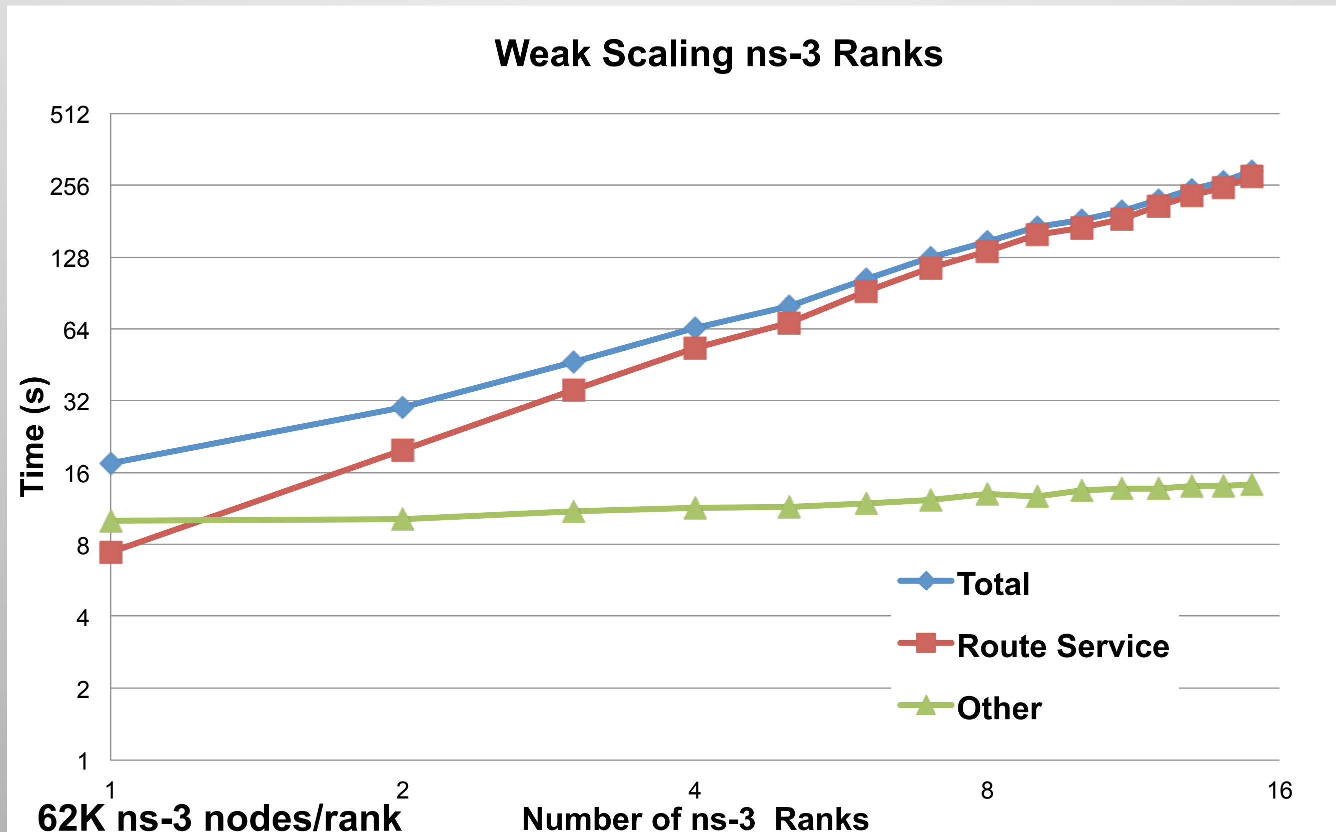




Weak Scaling ns-3 Ranks

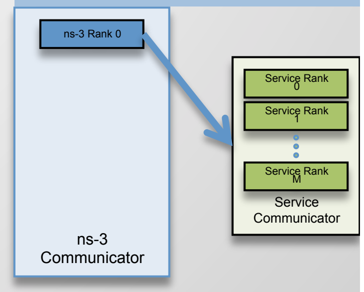
- Expect route service to get slower
  - Breadth-first search is *O*(Nodes+Links)

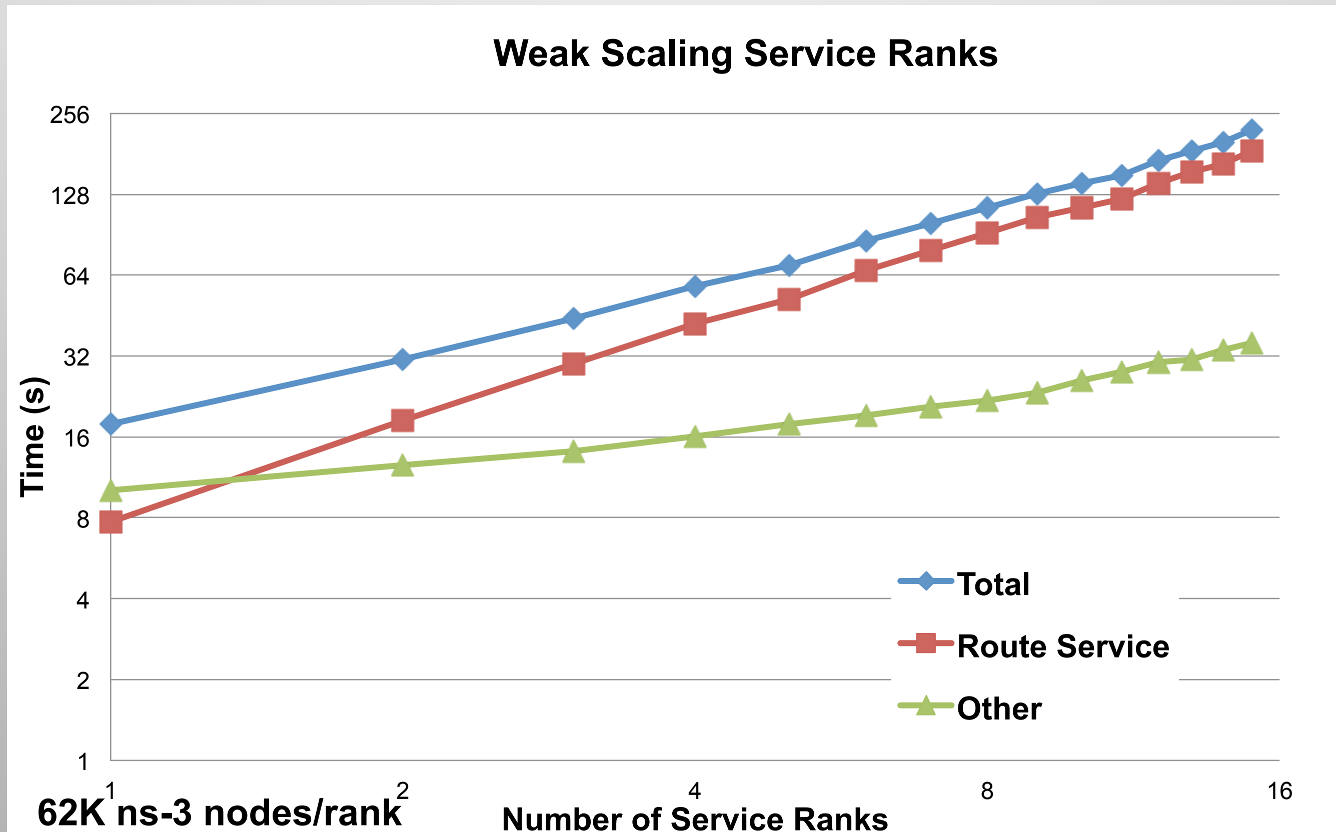




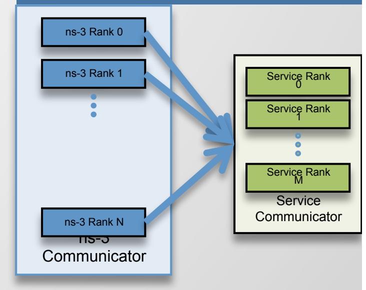
Weak Scaling Service Ranks

- Expect route service to get faster as more ranks used
  - Topology not high enough degree for PBGL
  - Average degree 1.004, PBGL scales well with 15





Large Scale Demonstration Problem



- Problem
  - 1.9M Campus LANS, 250 clients per LAN
  - 486M ns-3 nodes
  - 3840 ns-3 tasks, 256 NIx vector tasks
  - Limited to 400 route lookups to keep run times short
- Total runtime 18.7 minutes
- Completes!
- Average route lookup took 1.775s
  - Every node requiring one route would take 9982 days!

Ask Ken and David for a scalable "just works" routing algorithm.

Conclusion

- Achieved perfect memory scaling
- Implemented distributed NIx vector route service
  - Limited scalability: serializes route lookups
- Cleanup in process
  - Performance patch for existing NIx-vector
  - Use separate MPI communicator
  - Streamline `DistributedRouteServerInit (argv)`
  - Complete NodeId performance study
  - Replicated route service
  - Submit route service for review
  - Submit memory scaling for review

Future Work

- Matching needs work.
  - Global ID assigned to channel would be better.
- Global and node IDs need further discussion and work.
- Ken Renard’s NMS Campus model refactoring
- Understand lookup time increase in strong scaling of ns-3 Ranks
- Better benchmark topology to show PBGL scaling
- Scalable “just works” routing algorithm

Lawrence Livermore
National Laboratory