Algorithms to Optimally Use Hardware Forwarding Resources in a Router Domain

B.G.Prasanthi,
Research Student,
S.K.University,
Anantapur.

Dr.T.Bhaskara Reddy
Associate Professor
S.K.University,
Anantapur.

ABSTRACT

Routers and Forwarders in a Router domain would likely have different amount of hardware Resources. The least capable switch should not hold the performance of the Router network domain to ransom. This write up lists some smart choices the forwarders and routers in the Router domain can make to most efficiently utilize the hardware forwarding capabilities of each node in the network.

In a Router domain, all forwarders and routers belonging to a single Router domain keep every other Forwarder or router updated of all the host routes for each host known and present in the network.

The above is done for every subnet in the Router domain. Each forwarder and router installs all the routes in the fast path forwarding database (for example hardware forwarding tables).

In this new scheme, all the routers and forwarders learn all the routes, build the topology table for all subnets, but the difference is the router/forwarder can choose to put subnet of the routes in the fast path database (hardware database). This would save of the hardware resources, without sacrificing the network performance in most cases. In the cases where network path chosen is potentially suboptimal a further set of enhancements comes and loosens the optimization to further improve the network performance without wasting the forwarding resource utilization.

Keywords

Routers, Forwarders, Router Domain, Host, Fast path Database

Related work

An example of AI-based allocation algorithm is presented in Kichkaylo and Karamcheti (2004). This algorithm achieves resource optimality to improve throughput of applications and satisfy resource constraints at the same time. However, it focuses on a special application kind where components produce or consume data streams. The CANS framework [Fu 2003] enables dynamic deployment of components in the parts of network to ease handling of protocol conversation, data transcending, and mapping incompatible network partitions together. CANS optimize solutions using different application-related metrics, e.g. overall throughput. However, applications in CANS consist of components which are mapped into a direct sequence such as single sink-source chains. The AIRES [Wang et al. 2004] is an informed branch and bound allocating algorithm that aims to support software design automation. The AIRES uses a static resource model where all the devices share the same link. Thus, AIRES targets a limited case of the application allocation. Another approach is offered by HADAS [Ben-Shaul et al. 2004] and DecAp [Malek et al. 2005], both of which are decentralized agent-based systems. In HADAS, all the hosts and single components are autonomous entities that use a simple negotiation algorithm to discover resources and allocate components. However, HADAS has a significant drawback. The negotiation process is not time-limited and therefore agents cannot predict duration of allocation and deployment that is unacceptable in user-centric systems. The DecAp algorithm is based on the assumption that the host resources are not always known before the allocation. DecAp uses a distributed auctioneer algorithm to allocate components that are agents consuming or offering a resource. DecAp focuses on reliability of the system by solving the problem of disconnected operation, where the system continues functioning in temporary absence of network. However, we assume that disconnection of networking hosts has to be handled separately at the network level. This assumption simplifies algorithm and allows focusing on user-related metrics of applications that we find more important.

Algorithm

Algorithm exploits the following:

Step 1:All the Router Area border routers(aka Domain Border Routers)advertise all the subnet. Prefix/mask as a route in to the Router domain.

Step 2: Every router or forwarder in the domain is notified of all host routes for all subnets in the domain.

Step 3: Area border routes are the ones with the glean adjacency for the subnet, and are responsible for generation of the Router probe request to discover an unknown host.

Greedy optimization algorithm

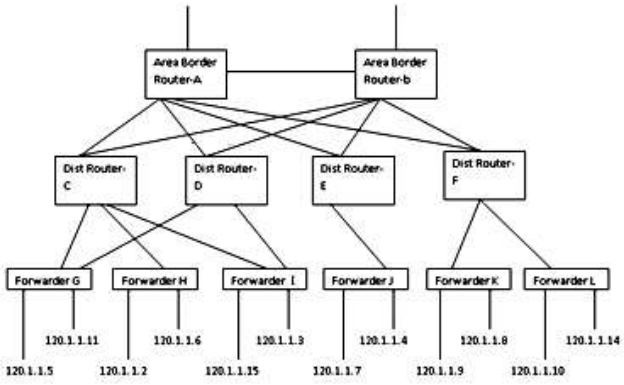
Each router or forwarder before installing the host route in hardware performs an Immediate less specific lookup. If a less specific route is present and the immediate less specific route has the same set of adjacencies as the new route to be installed then, the installation of the new host route in hardware is unnecessary and optimized out. Two set of adjacencies are considered as equal if they have the member adjacencies each with the same load distribution weights.

In case of an, hardware resource condition the subnet routes are always prioritized and installed, so that forwarding is correct. The host routes which would forward out using a different set of adjacencies as compared to less specific route are given the next priority in hardware for table space, in case there is no space to accommodate them, these host routes, then the less specific route has to forward using the slow path(software).

This is a safe algorithm and avoids creation of any routing loop, sub optimal forwarding etc.

But the optimization cannot be turned for the routes that are seen from forwarders that share a common set of redundant distribution switches. put a figure here to illustrate the case where the optimization helps and where it does not help.

For an example



FORWARDIND TABLE COMPUTED FROM THE ROUTING TABLE USING ADAPTIVE ALGORITHM (assume all links have same cost):

Forwarder G:

Prefix	Next Hop
120.1.1.5/32	120.1.1.5
120.1.1.11/32	120.1.1.11
120.1.1.2/32	Router C
	<< Less specific does not have an *equal*set
120.1.1.6/32	Router C
	<< Less specific does not have an *equal*set
120.1.1.0/24	ECR(Router C || Router D)

Forwarder H:

Prefix	Next Hop
120.1.2/32	120.1.2
120.1.2/32	120.1.1.6
120.1.2/32	Router C

Forwarder I:

Prefix	Next Hop
120.1.1.15/32	120.1.1.15
120.1.1.3/32	120.1.1.3
120.1.1.2/32	Router C
	<< Less specific does not have an *equal*set
120.1.1.8/32	Router F (Forwarder J is a candidate, but is not as optimization cannot be done with less specific ,its RD(2)<FD(3))
120.1.1.10/32	Router F
120.1.1.0/24	ECR(Router A || Forwarder B)

Router F

Prefix	Next Hop
120.1.1.4/32	Router E(Forwarder K is a candidate, but is not used as optimization cannot be done with less specific, its RD(2)<FD(3))
120.1.1.10/32	Router E(Forwarder K is a candidate, but is not used as optimization cannot be done with less specific, its RD(2)<FD(3))
120.1.1.9/32	Forwarder K
120.1.1.8/32	Forwarder K
120.1.1.10/32	Forwarder L
120.1.1.0/24	ECR(Router A ||Forwarder B)

Router B:

Prefix	Next Hop
120.1.1.5/32	ECR(Router C||Router D)
120.1.1.11/32	ECR(Router C||Router D)
120.1.1.2/32	Router C
120.1.1.6/32	Router C
120.1.1.15/32	ECR(Router C||Router D)
120.1.1.15/32	ECR(Router C||Router D)
120.1.14/32	Router E
120.1.1.7/32	Router E
120.1.1.9/32	Router F
120.1.1.8/32	Router F
120.1.1.10/32	Router F
0.1.1.0/24	Glean

Router A:

Prefix	Next Hop
120.1.1.5/32	ECR(Router C||Router D)
120.1.1.11/32	ECR(Router C||Router D)
120.1.1.2/32	Router C
120.1.1.6/32	Router C
120.1.1.15/32	ECR(Router C||Router D)
120.1.1.3/32	ECR(Router C||Router D)
120.1.1.4/32	Router E
120.1.1.7/32	Router E
120.1.1.9/32	Router F
120.1.1.8/32	Router F
120.1.1.10/32	Router F
120.1.1.0/24	Glean

Optimization Algorithm

Our algorithm produces better results as compared to the above one, as it's more aggressive but needs more information, to get it's job done right, like no routing loops.

It's needs to do *one* of the following :

A new Router route distribution protocol is used that propagates the link state topology of the network, and every host route is advertised with the source forwarder (first forwarder to which the lost is connected). Now if the link state topology is known along with the source forwarder for the host route, any forwarder or router in the network can safely compute the adjacencies That can be used to reach the host(these adjacencies may have different load distribution values), but there is a guarantee that there will never be a routing loop. The forwarder can ignore the load distribution when performing the equality check in step IIa below, and just treat all the adjacencies as equal cost. Picking up the less optimal non looped paths enables the algorithm in step IIa to perform an

aggressive less specific match, as it takes all the paths to reach the destination, Irrespective of the load distribution.

Another alternative to the above is, play with the link costs such that the cost of the distribution links is less than that of the forwarder to router uplinks. Then use EIGRP for route distribution and exploit a concept similar to variance. Exploit concepts the EIGRP Concepts of feasible distance, reported distance, feasible successor to always prevent a routing loop, but still pick up less optimal paths. If the less specific's adjacency set, consists of the same adjacency members as the host route, then treat the routes as equal, if not increase the member set of host routes by including the feasible successors, now if the sets have the same members, optimization can be done. Picking Up the less optimal non looped paths enables the algorithm in step IIa to perform an aggressive less specific match, as it takes all the paths to reach the destination, irrespective of the load distribution. That is why is called adaptive algorithm optimization algorithm.

Optimization techniques

Technique 1:

Each router or forwarder before installing the host route in hardware performs a less specific lookup. if a less specific route is present and the less specific route has the same equivalent set of adjacencies as the new route to be installed then, the installation of the new host route in hardware is considered unnecessary and optimized out. Two set of adjacencies are considered equivalent if they have the same member adjacencies irrespective of the load distribution weight see the options in 4 & 5 to see when this is possible.

Technique 2:

In case of, an out of hardware resource condition the subnet routes are always prioritized and installed, so that forwarding is correct. the host routes which would forward out using the a different set of adjacencies as compared to the less specific route has to forward using the slow path(software).

Technique 3:

This adaptive algorithm use suboptimal paths to forward traffic In the network at times, from the access forwarders. This weakness can be mitigated, by making the algorithm adaptive, and take a feedback from the actual user traffic in the network. Making the algorithm adaptive does not slow down the forwarding of data traffic, that is does not put the algorithm in the data path, but rather the algorithm samples copies of the traffic being forwarded in the hardware, and matching the less specific subnet routes. On a catalyst switch like 3750 this could be implemented by forwarding packets and hardware but also copying the packets to a special CPU Queue or DI. Packets received on this queue are not forwarded, but just used for sampling of the end destinations for which there is active traffic. From this sampled traffic the software further isolates those destinations, for which an optimal path is present in the network with possibly a different load distribution, than the less specific prefix route. For each of these DAs if space is present in hardware the software installs a dedicated/32or /128 entry. when the hardware resources fall below a particular watermark, the software looks at the activity bits of a statistics bucket attached to these /32 or /128 entries that are not be used are garbage collected. if the platform has small number of statistics buckets, then other schemes can be used like cycling the buckets through the entries that may be candidates for garbage collection or using an elaborate divide and conquer scheme using fixed statistics buckets to find HW entries that are not used recently.

FORWARDIND TABLE COMPUTED FROM THE ROUTING TABLE USING ADAPTIVE ALGORITHM (assume all links have same cost):

Forwarder G:

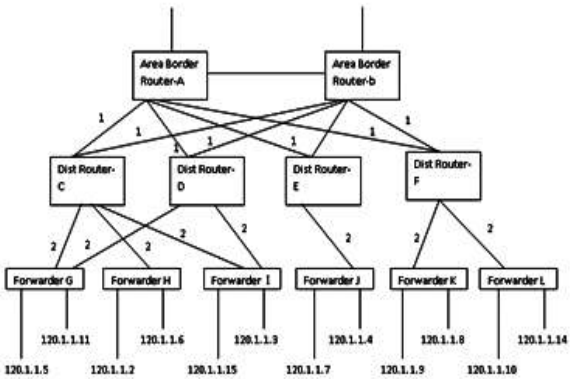
Prefix	Next Hop
120.1.1.5/32	120.1.1.5
120.1.1.11/32	120.1.1.11
120.1.1.2/32	Router C
	<< Less specific does not have an *equal*set
120.1.1.6/32	Router C
	<< Less specific does not have an *equal*set
120.1.1.0/24	ECR(Router C || Router D)

Forwarder H:

Prefix	Next Hop
120.1.2/32	120.1.2
120.1.2/32	120.1.1.6
120.1.2/32	Router C

Forwarder I:

Prefix	Next Hop
120.1.1.15/32	120.1.1.15
120.1.1.3/32	120.1.1.3
120.1.1.2/32	Router C
	<< Less specific does not have an *equal*set
120.1.1.8/32	Router F
	(Forwarder J is a candidate, but is not as optimization cannot be done
	with less specific ,its RD(2)<FD(3))



120.1.1.10/32	Router F
120.1.1.0/24	ECR(Router A || Forwarder B)

Router F

Prefix	Next Hop
120.1.1.4/32	Router E(Forwarder K is a candidate,

	but is not used as optimization cannot be done with less specific, its RD(2)<FD(3))
120.1.1.10/32	Router E(Forwarder K is a candidate, but is not used as optimization cannot be done with less specific, its RD(2)<FD(3))
120.1.1.9/32	Forwarder K
120.1.1.8/32	Forwarder K
120.1.1.10/32	Forwarder L
120.1.1.0/24	ECR(Router A ‖Forwarder B)

Router B:

Prefix	Next Hop
120.1.1.5/32	ECR(Router C‖Router D)
120.1.1.11/32	ECR(Router C‖Router D)
120.1.1.2/32	Router C
120.1.1.6/32	Router C
120.1.1.15/32	ECR(Router C‖Router D)
120.1.1.15/32	ECR(Router C‖Router D)
120.1.14/32	Router E
120.1.1.7/32	Router E
120.1.1.9/32	Router F
120.1.1.8/32	Router F
120.1.1.10/32	Router F
120.1.1.0/24	Glean

Router A:

Prefix	Next Hop
120.1.1.5/32	ECR(Router C‖Router D)
120.1.1.11/32	ECR(Router C‖Router D)
120.1.1.2/32	Router C
120.1.1.6/32	Router C
120.1.1.15/32	ECR(Router C‖Router D)
120.1.1.3/32	ECR(Router C‖Router D)
120.1.1.4/32	Router E
120.1.1.7/32	Router E
120.1.1.9/32	Router F
120.1.1.8/32	Router F
120.1.1.10/32	Router F
120.1.1.0/24	Glean1

ALGORITHM:

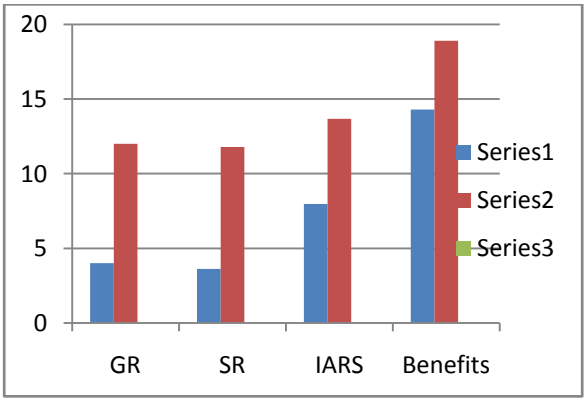
Measurement-Period, t ∈ (Cx.1,Mx)
for every neighbor node j do
Sij ←a monitoring scheme for the link from node i to node j
if Sij == PASSIVE or ACTIVE then
monitor egress traffic to node j
else if Sij == COOPERATIVE then
monitor egress traffic from node i to node k that node j overhears
end if
if node i received a cooperation request (.) from node j then
overhear cross traffic from node j to node .
end if
end for
(2) At the end of a Measurement-Period, t = Mx
for every neighbor j do
record measurement results from node i to node j
if node i received a cooperation request (.) from node j then
send node j a report of overhearing traffic from node j to node .
end if
end for
3) During an Update-Period, t ∈ (Mx,Mx + Ux)
process a measurement report(s) from other nodes, if any
(4) End of an Update-Period, t = Mx + Ux (or, t = Cx)
for every neighbor j do
calculate the quality of link from node i to j using Eq. (2.1)
run the transition algorithm (in Figure 2.2) for node j
if transition to COOPERATIVE then
choose node k that node j can overhear
send a cooperation request (k) to node j
else if transition to ACTIVE then
schedule active probe packets
end if
end for

Results:

We have implemented the proposed routing algorithm in C-Programming with respect to hardware routing resources in an IP domain. We have tested our algorithm on a set of 50 nodes with 14 routers. We explore the tradeoff between the performance of routing with generic, symmetric, IARS (implemented system) .

Hardware forwarding routing table

	Generic routers	Symmetric routers	IARS	Benefits
Technique 1	4 %	3.62 %	7.98 %	14.3 %
Technique 2	12 %	11.8 %	13.68%	18.90



CONCLUSION

Both algorithms found equally good solutions in most of the cases, as demonstrated by our preliminary experiments. However, this adaptive algorithm is more robust and stable than the greedy one. The algorithm based on evolutionary computing also demonstrated a better calculation time, although it had a little time overhead on small applications. As can be seen, this algorithm has a success ratio bigger than 18% for large applications. The fact that the success ratio of this algorithm increases with the application size is (or appears to be) due to an increased number of additional constraints imposed by the application components that greedy algorithm cannot handle. The limitation of the current implementation of this algorithm is the resource model that supports only specific types of resources, such as hosts and network

link physical constraints. We identify a need to use a generic resource model, where each host can provide any kind of resource. This generic model has to take into use, for example, displays, microphones and other peripheral devices offering additional services for the user. Moreover, in a future implementation of the algorithm.

REFERENCES

1 . Ben-Shaul I. et al, 2004. Dynamic adaptation and deployment of distributed components in HADAS. The IEEE Transactions on Software Engineering, 27, 9, 769-787.Fu X., 2003. Infrastructure support for accessing network services in dynamic network environments. Phd thesis. NewYork University.

2 Gross T. et al, 1999. Adaptive distributed applications on heterogeneous networks. In Proceedings of the 8th Heterogeneous Computing Workshop. 209.Kejariwal A. and Nicolau A., 2005. An Efficient Load Balancing Scheme for Grid-based High Performance Scientific Computing. In Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC'05).CA. USA. 217- 225.

3 Kichkaylo T., 2005. Construction of Component-Based Applications by Planning. Phd thesis. New York University.Kichkaylo T. et al, 2003. Constrained Component Deployment in Wide-Area Networks Using AI Planning Techniques.In Proceedings of International Parallel and Distributed Computing Symposium (IPDPS'03). Nice. France.

4 Kichkaylo T. and Karamcheti V., 2004. Optimal resource-aware deployment planning for component-based distributed applications. In Proceedings of 13th IEEE International Symposium on High Performance Distributed Computing.150 – 159.

5 Malek S. et al, 2005. A Decentralized Redeployment Algorithm for Improving the Availability of Distributed Systems. In Proceedings of 3rd International Working Conference on Component Deployment (CD'05). Grenoble. France.Michalewicz Z. and Fogel D. B. 2000. How to solve it: modern heuristics. Berlin: Springer.

6 Perttunen M. et al, 2007. A QoS Model for Task-Based Service Composition. 4th International Workshop on Managing Ubiquitous Communications and Services (MUCS 2007), Munich, Germany, 25 May.

7 Ranganathan A. and Campbell R., Autonomic Pervasive Computing Based on Planning. In Proceedings of the First International Conference on Autonomic Computing (ICAC'04), 17-19 May 2004, New York, NY, USA, pp. 80-87.Roman M. et al, 2002. "A middleware infrastructure for active spaces". Pervasive Computing, IEEE, Vol. 1, 4, pp. 74-83.

8 Satyanarayanan M., 2001. "Pervasive computing: vision and challenges", IEEE Pers. Commun., Vol. 8, 4, pp. 10-17.Tanenbaum A. and Maarten S., 2002. Distributed Systems: Principles and Paradigms. Prentice Hall.

10 Wang S. et al, 2004. Component Allocation with Multiple Resource Constraints for Large Embedded Real-Time Software Design. In Proceedings of IEEE Symposium on Real-Time and Embedded Technology and Applications(RTAS'04). Toronto. Canada.