Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp418-422)

On Convergence and Reliability in Self-Configuring Virtual Topologies

ARSO SAVANOVIĆ
Smart Com d.o.o.
Brnčičeva 45
1001 Ljubljana Črnuče
SLOVENIA

Abstract: - Virtual Private Networks (VPNs), Peer-to-Peer networks (P2P), active networks, and overlay research testbeds are examples of virtual topologies on top of the underlying IP infrastructure. Due to the ubiquitous and cost-effective nature of the Internet, the virtual topologies on top of Internet are becoming ever more popular and widespread. However, automatic configuration and management facilities are a necessary prerequisite for successful large scale deployment of virtual topologies. We have developed an automatic discovery protocol for virtual topologies, which has been documented more thoroughly in our previous work. This paper focuses on the analysis of reliability and convergence of this virtual topology self-configuration protocol.

Keywords: - protocol, self-configuration, virtual topology, adaptive system, convergence, reliability

1 Introduction

Virtual Private Networks (VPNs), Peer-to-Peer networks (P2P), active networks, and overlay research testbeds are examples of virtual topologies on top of the underlying IP infrastructure. Virtual topologies can be categorised into two distinct classes: edge virtual topologies comprise only end-hosts and are statically predefined based on user or application requirements, while core virtual topologies comprise distinguished network nodes and they do not depend on user and application requirements. Thus, in the latter case it is beneficial for the virtual topology that each distinguished node be directly connected only with neighbour distinguished nodes. Such virtual topology optimally matches the underlying IP infrastructure.

The autonomy of network elements is among the important networking paradigms, which simplify administration and management of networks and consequently contribute towards better network scalability. The autonomy of network elements encompasses the use of various mechanisms and protocols, which automate different administration and management tasks, such as configuration of elements, fault detection and recovery, detection of changes and element reconfiguration, etc. Manual administration of network elements is unacceptable in large scale networks, which is even more the case when one considers the dynamic nature of IP networks. Virtual topologies are built on top of the dynamic IP infrastructure. Consequently, distinguished nodes should be able to adaptively search for and connect with neighbour distinguished nodes and, thus, adaptively maintain a virtual topology, which optimally matches the current state of the underlying IP infrastructure. This calls for a mechanism, which enables distinguished nodes to automatically and dynamically discover their neighbours in a virtual topology.

In a virtual topology neighbour distinguished nodes are generally not directly connected, i.e. there may be one or more plain nodes (routers) between them. This makes neighbour discovery in a virtual topology a non-trivial problem and fundamentally different from physical neighbour discovery.

In our previous work we have presented a solution for this problem in the form of a network protocol for neighbour discovery in virtual topologies. The neighbour discovery protocol is characterised by some important properties, most notably its reliability, convergence delay, and overhead. In this paper we inspect more closely the convergence and reliability issues related to the protocol operation and consequetly to the self-cofiguration of virtual topologies.

2 Self-configuring virtual topologies

The problem of neighbour discovery in physical networks is trivial: a node simply transmits the “who is there?” message on the physical link and the neighbour at the other end of the link is able to receive this message and return its identity in response.

However, this simple principle cannot be applied in a virtual topology, where neighbours are generally not directly connected, but are instead *virtual neighbours*, i.e. there may be one or more plain routers between them. This makes neighbour discovery in a virtual topology a non-trivial task and fundamentally different from physical neighbour discovery.

The straightforward approach is to manually configure each distinguished node with addresses of its neighbour distinguished nodes. However, this is an administrative nightmare and it scales poorly due to the fact that both, data networks and virtual topologies on top of them are dynamic systems, whereas manual configuration is a static operation.

The protocol we have developed enables distinguished nodes in the network to atomatically and dynamically discover neighbouring distinguished nodes and, thus, facilitates self-configuring virtual topologies [9, 10, 2].

The protocol is essentially used by distinguished nodes to constantly poll the status of the underlying IP network. Whenever a network change is detected, which is relevant for the structure of the optimal virtual topology, distinguished nodes update their configuration and consequently also update the structure of the virtual topology.

The protocol uses three message types and operates in two different modes, the coordination mode and the discovery handshake mode. A node coordinates its activities with other members of the virtual topology by periodically advertising itself to other nodes via an emulated communications bus and by processing other nodes' advertisements. The core of the protocol is the discovery handshake, which performs the majority of tasks related to neighbour discovery, detection of changes, and node (re)configuration.

Figure 1 illustrates the operation of the neighbour detection protocol. Each member of a virtual topology keeps an updated table of known virtual neighbours, which contains neighbour IDs and an associated timer that facilitates the aging of neighbour entries (soft state). A node in virtual topology advertises itself by periodically (with the period of TO_{HW}) transmitting the advertisement message Hello_World to an emulated communications bus, which distributes the advertisement to all other members of the same virtual topology. The implementation of an emulated communications bus can be multicast-based, with one dedicated multicast group per virtual topology, server-based, or a combination of both.

Upon reception of an advertisement message from a node, other nodes in the virtual topology each send the

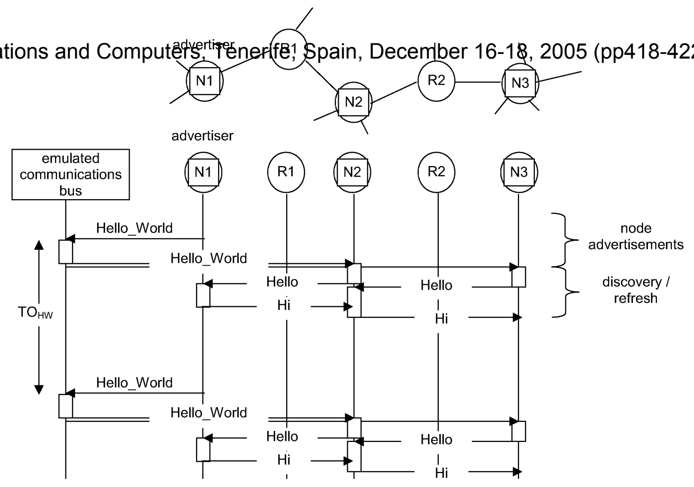

Figure 1: Protocol operation overview

"advertiser" their IDs within the Hello message. If the source node and the advertiser are virtual neighbours, then the Hello message reaches the advertiser, which in response returns its own ID within the Hi message. However, if the source node and the advertiser are not virtual neighbours, then the actual virtual neighbour, i.e. the first node of the virtual topology, which is en route to the advertiser, intercepts and blocks the Hello message and responds with the Hi message containing its own ID. In any case, the sender receives the Hi message from its actual virtual neighbour and either refreshes (for known neighbours) or adds (for a new neighbour) the entry for this node in the neighbour list. When a neighbour's entry in the neighbour list times out, the neighbour is deleted from the neighbour list and is not considered to be neighbour anymore by the particular node.

Packet losses, which can occur in the Internet, are dealt with in two ways: with periodic advertisements and with timeouts for neighbour entries. The periodic advertisements have the effect of message retransmission in case of a detected loss: the Hello_World message spawns the Hello/Hi exchange and when either of the three protocol messages is lost, then the next Hello_World advertisements causes the "retransmission", which eventually leads to a successful sequence of all three protocol messages and, thus, successful neighbour detection/refreshment. On the other hand, timeout values for entries in the neighbour list are set to $m * TO_{HW}$, where m is a small integer. Consequently a virtual topology node deletes its neighbour from the neighbour list only after it has failed to receive m consecutive Hi messages. This prevents erroneous deletion of an actual neighbour when up to $m-1$ consecutive messages get lost during the transfer. Note, however, that in spite of these mechanisms, which help the protocol deal with packet losses in the network, there is always a small residual effect of these losses present, which makes the protocol operation probabilistic. This is discussed further in the next section.

3 Convergence and Reliability Analysis

In this section we are interested in convergence delay of the neighbour detection protocol, more precisely in estimating the upper bound on convergence delay. Convergence delay is defined as a time difference between the time, when a relevant change occurs in the network infrastructure, and the time, when this change has been successfully detected and self-configuring virtual topology updated accordingly. Neighbour detection protocol has to deal with the following basic scenarios for the network infrastructure change:

- a new virtual topology node is added to the network: the new node and its neighbours have to mutually detect and configure each other as neighbours and a node has to detect optional "removal" of an existing neighbour due to the change (i.e. the new node is placed in between two virtual neighbours, so the former neighbour is effectively "removed" form the neighbour list),
- an existing virtual topology node is removed from the network infrastructure: neighbours have to both detect the removal of a neighbour and detect "new" neighbours, which show up after the removal of the node,
- changes in the underlying Internet only, e.g. router/link addition or removal: from the virtual topology perspective these changes translate to one of the three cases—removal of an existing neighbour, addition of a new neighbour, and the combination of both of the above—which have to be detected by virtual topology nodes.

It turns out that the virtual neighbour detection as presented in section 2 performs the two functions (discovery of a new neighbour and detection of the removal of an existing neighbour) independently. The reason for this is quite simple: a virtual topology node cannot be both at the same time, a new neighbour and an existing neighbour to some other node. Consequently, the convergence delay can be analysed independently for these two cases, and since we want to estimate the upper bound on convergence delay, it suffices to analyse only the worst-case scenario among the relevant scenarios listed at the beginning of this section.

The worst-case scenario for the removal detection of an existing neighbour is as follows: immediately after the virtual neighbour of some node sends the last Hi message, this neighbourhood relationship is broken due to a change in the network, e.g. a route changes or the neighbour node crashes/is actually removed. The maximum convergence delay TCD_{MAX} corresponding to this event is given as (1).

$$TCD_{MAX} = m * TO_{HW} + RTT_{MAX}/2 \qquad (1)$$

The first term of (1) reflects the fact that a node deletes its neighbour from the neighbour list only after it detects m consecutive missing messages Hi from the neighbour, in order to account for packet losses in the network (see section 2). The second term represents the transfer delay of the last Hi message from the (former) neighbour, where RTT_{MAX} represents the maximum round trip delay between a pair of nodes in a virtual topology and is typically around 1s. Note that despite the above, packet losses in the network can cause the node to temporarily and erroneously delete the neighbour from the neighbour list. The probability of this erroneous event can be obtained after a short derivation and is given with $P_{ERR} = (1 - (1 - p_L)^3)^m$, where p_L gives the probability of a packet loss in the network. Thus, the reliability of the protocol, i.e. the probability that the protocol operates as desired in spite of network losses, is given as (2).

$$P_{REL} = 1 - P_{ERR} = 1 - (1 - (1 - p_L)^3)^m \qquad (2)$$

Typically, network losses are in the range $p_L = 2\% \div 5\%$. If we want the reliability of the protocol to be, for example, $P_{REL} \geq 99.9\%$ for these typical losses, then m should be set to $m \geq 4$ according to (2).

The worst case scenario for the detection of a new neighbour, on the other hand, is when:

- a change causing some node to get a new neighbour occurs immediately after the last Hello_World message of that particular node has been distributed via an emulated communications bus and
- due to packet losses only the n-th following sequence of Hello_World/Hello/Hi messages has been successful.

The maximum convergence delay TCU_{MAX} corresponding to this event is given as (3), where RTT_{MAX} represents the maximum round trip delay between a pair of virtual topology nodes.

$$TCU_{MAX} = n * TO_{HW} + RTT_{MAX} \qquad (3)$$

Note that n can be arbitrarily large (albeit with an extremely small probability) due to packet losses. Again, a short derivation shows that the probability that the new neighbour is detected within the time window of TCU_{MAX}, i.e. only after n messages Hello_World, is given with (2), where m is replaced with n. Thus, equation (2) again represents the reliability of the neighbour detection, although with a slightly different meaning. If we again assume typical network loss probability and want the reliability of the protocol to be e.g. $P_{REL} \geq 99.9\%$, then (2) yields $n \geq 4$.

Finally, to provide an example of a complete protocol configuration, we must select an appropriate value for the retransmission period TO_{HW}. Since neighbour detection is in many ways analogous to routing protocols in IP networks, we want the convergence delays of the two to be comparable. Given that the OSPF is among the best Internet routing protocols in this respect and that its typical convergence time is around 30s÷40s, we conclude from (1), (3), and from typical values for n, m, and p_L that the value of $TO_{HW} \approx$ 10s yields a delay for virtual neighbour detection, which is comparable to the OSPF convergence delay.

4 Conclusions

Due to the ubiquitous and cost-effective nature of the Internet, the virtual topologies on top of Internet are becoming ever more popular and widespread. However, automatic configuration and management facilities are a necessary prerequisite for successful large scale deployment of virtual topologies.

In this paper we have shortly described a discovery protocol for self-configuring virtual topologies and analysed its operation with respect to reliability and convergence delay. The results of the analysis indicate that protocol parameters can be configured in a trade-off manner, such that the protocol meets the criteria and expectations with regards to the two properties. We note that there are other important properties of the protocol, most notably protocol overhead, which should be considered and can pose contradictory requirements.

The networking community has already developed various discovery mechanisms for virtual topologies, for some examples see e.g. [11, 1, 12, 8, 13, 4, 14, 5, 3]. However, majority of this work does not deal with neighbour discovery and topology adaptation, but rather with topology mapping, server discovery, and service discovery in virtual topologies, which is not directly related to our work. Furthermore, as far as the author is aware, there are some simulation results but no analytical evaluation results for the discovery mechanisms, which are more closely related to our work, most notably [6, 7].

Acknowledgement

The work presented in this paper has been sponsored by the Ministry of Science and Technology of Republic of Slovenia. Most of the work has been carried out at the Jožef Stefan Institute in Ljubljana, Slovenia.

References

[1] Yuri Breitbart, Minos Garofalakis, Rajeev Rastogi, S. Seshadri, and Avi Silbershatz. Topology discovery in heterogeneous ip networks. In *IEEE INFOCOM 2000*, 2000. Tel Aviv, Israel.

[2] Lawrence Cheng, Alex Galis, Arso Savanović, Borka Jerman Blažič, and Janez Bešter. Self-Management GRID Services—A Programmable Network Approach. *Lecture Notes in Computer Science*, (3038):141–148, 2004.

[3] Prithviraj Dasgupta. Incentive Driven Node Discovery in a P2P Network Using Mobile Intelligent Agents. In *Proceedings of the International Conference on Artificial Intelligence, IC-AI '03,*, volume 2, pages 750–756, June 2003.

[4] Adriana Iamnitchi and Ian Foster. On Fully Decentralized Resource Discovery in Grid Environments. In *Proceedings of International Workshop on Grid Computing*, November 2001.

[5] Pedram Keyani, Brian Larson, and Muthukumar Senthil. Peer Pressure: Distributed Recovery from Attacks in Peer-to-Peer Systems. In *Proceedings of the NETWORKING 2002 Workshop: Web Engineering and Peer-to-Peer Computing*, pages 306–320, May 2002.

[6] Sylvain Martin and Guy Leduc. RADAR: Ring-based Adaptive Discovery of Active neighbour Routers. In *IWAN 2002*, pages 62–73, December 2002.

[7] Sylvain Martin and Guy Leduc. A Dynamic Neighbourhood Discovery Protocol for Active Overlay Networks. In *IWAN 2003*, pages 151–162. Springer-Verlag, December 2003.

[8] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing For Large-Scale Peer-to-Peer Systems. In *Proceedings of International Conference on Distributed Systems Platforms*, pages 329–350, November 2001.

[9] Arso Savanović. Automatic Discovery of Neighbour Active Network Nodes. Technical Report IJS DP-8725, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia, January 2003.

[10] Arso Savanović and Borka Jerman Blažič. Dynamic neighbour discovery for self-organising active networks. In *Proceedings of ConTEL 2003*, pages 213–219. University of Zagreb, 2003. June 11–13, 2003, Zagreb, Croatia.

[11] R. Siamwalla, R Sharma, and S. Keshav. Discovering internet topology. Submitted to IEEE INFOCOM '99, 1999.

[12] Skitter. `http://www.caida.org/tools/measurement/skitter/`.

[13] Ion Stoica *et. al*. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of SIGCOMM 2001*, pages 149–160, August 2001.

[14] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Networks. In *Proceedings of International Conference on Distributed Computing Systems ICDCS'02*, pages 5–14, July 2002.