Network and Domain Autoconfiguration: A Unified Approach for Large Dynamic Networks

Kyriakos Manousakis and John S. Baras, University of Maryland
Anthony McAuley and Raquel Morera, Telcordia Technologies

ABSTRACT[1]

Configuration management is critical to correct and efficient operation of large networks. Where the users and networks are dynamic and ad hoc, manual configuration quickly becomes too complex. The combination of the sheer number of nodes with heterogeneity and dynamics makes it almost impossible for the system administrator to ensure good configuration or even correct operation. To achieve the vision of pervasive computing, nodes must automatically discover their environment and self-configure, then automatically reconfigure to adapt to changes. Protocols such as DHCP, DDNS, and mDNS provide some degree of host autoconfiguration, but network administrators must still configure information such as address pools, routing protocols, and OSPF routing areas. Only limited progress has been made in automating the configuration of routers, servers, and network topology. We propose the first unified attempt to combine both self-configuration of much of the host, router, and server information with automatic generation and maintenance of hierarchy under the same algorithmic framework. Testbed implementations show the approach is practical, while simulations reveal its scalability, rapidity, and efficiency with respect to network performance.

INTRODUCTION

Future commercial, military, and emergency networks require changes to traditional network management. Configuration management, in particular, must ensure correct and efficient network operation through setting parameters such as:

- IP Addresses of an interface
- Network parameters (e.g., default maximum transmision unit, MTU, size)
- Server addresses (e.g., for DNS or certificate authority server)
- Routing information (e.g., default route or routing protocols)
- IP address pools (e.g., for DHCP or MADCAP server)
- Security keys

While protocols such as Dynamic Host Configuration Protocol (DHCP), Dynamic Domain Name Servers (DDNS), and mobile DNS (mDNS) have allowed more autoconfiguration, network administrators must still manually configure much of this information. We need new protocols that are able to configure all these parameters, especially in routers and servers.

In many cases configuration management must also construct hierarchies (e.g., routing areas and security domains) for scalability, efficiency, and manageability. Today, the construction of hierarchies is a manual process performed offline by experts because it requires difficult optimizations. For example, in the creation of Open Shortest Path First (OSPF) areas in dynamic networks, the savings in reduced routing overhead must be balanced with the overhead of hierarchy maintenance and mobile node reconfiguration. Figure 1 shows an example of how OSPF areas, with aggregation at area boundaries, reduce the number of OSPF link state advertisement (LSA) packets in a network (routing overhead). When all nodes are placed into one area, the routing overhead grows quadratically with the number of nodes (n); however, with a two-level hierarchy with $\sqrt{n}$ nodes per OSPF area, overhead grows much less rapidly. This does not mean, however, that, for example, a 25-node network should be divided into 5 domains with 5 nodes in each domain. The configuration management must take into account the increased hierarchy maintenance and mobility management overheads. Further adding to complexity, Figure 1 shows that the lowest routing overhead is achieved by keeping nodes with similar velocities into the same areas.

We believe configuration management must be **cheaper** (e.g., more plug and play), **more robust** (e.g., no human intervention), **faster** (e.g., in seconds), and **better optimized** (e.g., to the link error rate) than current approaches. Moreover, configuration management must be able to

[1] *Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance (CTA) Program, Cooperative Agreement DAAD19-2-01-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.*

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

0163-6804/05/$20.00 © 2005 IEEE

deal with more **network dynamics** (e.g., varying topology) and be more **scalable** (e.g., to support 10,000 nodes). In some cases the configuration must be done with little or no fixed infrastructure.

This article proposes the autoconfiguration of most host, router, and server information, including the automatic generation and maintenance of hierarchy, under the same architectural, algorithmic, and protocol framework. We highlight existing work in the area of network and hierarchy autoconfiguration. We present our approach to network autoconfiguration using a framework based on the IP Autoconfiguration Suite (IPAS). Some of the key algorithms used in network and hierarchy autoconfiguration are then described. We also describe the testbed implementation and indicative simulation results.

Related Work

This section characterizes and gives some representative examples of network autoconfiguration protocols (although important in other layers, we concentrate on network layer autoconfiguration). We begin by dividing network configuration protocols into two basic categories:

- **Server-based protocols**. Configuration servers (e.g., Point-to-Point Protocol, PPP, and DHCP servers) allow hosts (clients) to dynamically obtain globally routable addresses and other configuration parameters when they establish a link. DHCPv6 prefix delegation servers can also configure pools of addresses to stub networks. In all cases a network administrator must preconfigure the servers themselves (e.g., subnet prefix).
- **Distributed protocols**. There are several approaches to configuration without a server, the best known being IPv6 stateless autoconfiguration of link-local addresses. The IETF Zeroconf Working Group has also defined a suite of distributed protocols, such as mDNS, for environments that lack configuration servers. Most of this distributed configuration is limited to single-segment networks, where all participating nodes can communicate directly or multiple such networks are connected on the same router.

Of the methods to avoid duplicate addresses assignments, we classify them into three categories:

- **Conflict detection allocation [1, 2]**. Nodes must perform proactive duplicate address detection (DAD) to check if their address is already used by another node (in IPv6 the Neighbor Discovery protocol performs DAD).
- **Best effort allocation [3, 4]**. Servers try to allocate unused addresses, based on their knowledge of addresses assigned so far. The new node also utilizes DAD to guarantee that the assigned address is free.
- **Conflict-free allocation [5]**. Servers are assigned a disjoint address pool, and the servers in turn only unallocated addresses to new nodes. Some address reclamation is generally needed, but DAD can be avoided.

For the hierarchy configuration, we divide the approaches into two broad categories:

- **Topology-based hierarchy**. Ephremides [6] introduced the idea of a distributed approach to creating a two-level hierarchy. The idea of hierarchy in ad hoc networks was revisited in the context of mobile multimedia wireless networks by Gerla [7].
- **Environment-aware hierarchy**. By taking into account more about the network environment, the hierarchy can be more efficient and robust. McDonald [8], for example, describes a more robust hierarchy in mobile environments using the probability of path availability with respect to time.

There is, however, no existing work that attempts to combine both network and hierarchy configuration for large heterogeneous networks under a unified framework.

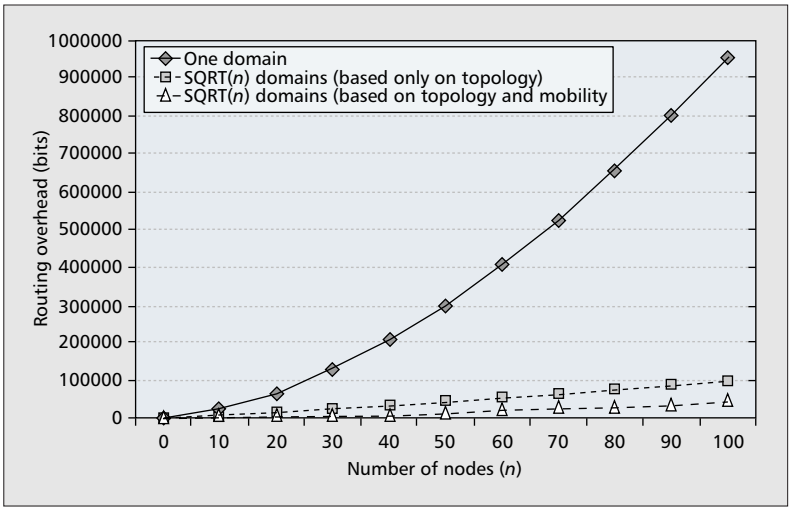


Figure 1. *Reducing routing overhead using a two-level hierarchy.*

IP Autoconfiguration Suite

This section describes our architectural and protocol framework for automatic configuration and dynamic reconfiguration in large-scale dynamic networks. To provide a unified architectural and protocol framework for configuring all networking parameters, we incorporated additional capabilities (e.g., to carry routing protocol and domain information) into the already existing IPAS [9].

Figure 2 shows the IPAS configuration process as a closed feedback loop. The IPAS uses a single set of mechanisms for collecting and distributing all configuration information, thus reducing overhead and complexity. The adaptive configuration agent (ACA) distributes new configuration through Dynamic Configuration Distribution Protocol (DCDP) to nodes in each subnet. DRCP configures the interfaces within a subnet. Interfaces configured by Dynamic and Rapid Configuration Protocol (DRCP) report configuration information and nodes' capabilities to the configuration server via the Yelp Announcement Protocol (YAP). The configuration server stores this information in the configuration database. To complete the cycle, the ACA node contacts the configuration database to get the latest configuration information in order to help decide when to reconfigure the network, starting the cycle once more. When a new node enters the network, its DRCP mechanism

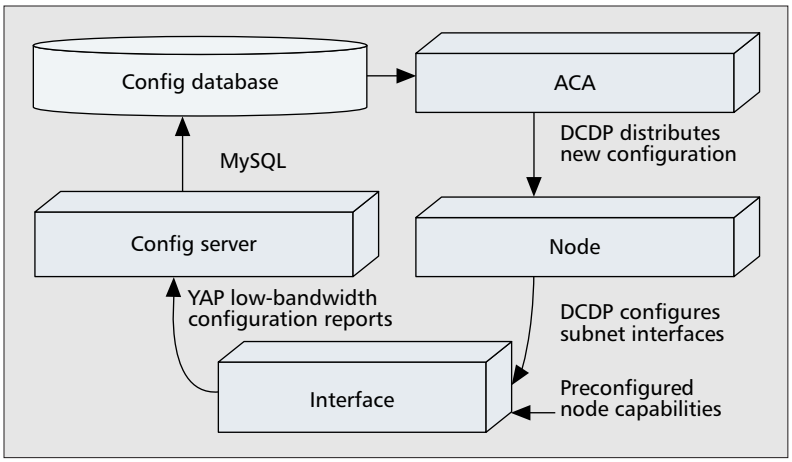


Figure 2. *IPAS components.*

requests configuration. If the DRCP server does not have an available address to configure a node, it requests addresses from its local DCDP process. If the DCDP process does not have available addresses, it requests addresses from its neighboring DCDP processes. The main functionality of the five modified IPAS components (all extended to allow for hierarchy generation and routing protocol configuration) are listed below:

ACA. The brain of IPAS, the ACA makes global decisions about configuration and reconfiguration (e.g., which nodes should be in which domain, which routing protocols to run, or which address pool to use).

DCDP. DCDP is a robust, scalable, low-overhead, lightweight (minimal state) protocol designed to distribute configuration information (e.g., DNS server's IP address or position in the routing hierarchy). Designed for dynamic wireless networks, it operates without central coordination or periodic messages and does not rely on a routing protocol.

DRCP. DRCP borrows heavily from DHCP, but adds features critical to roaming users. DRCP automatically detects the need to reconfigure (e.g., due to node mobility) through periodic advertisements. In addition, DRCP allows:

- Efficient use of scarce wireless bandwidth
- Dynamic change of address pools for server failover
- Minimizing broadcasts
- Clients to be routers

YAP. YAP is a simple bandwidth-efficient reporting mechanism with three elements:

- Clients on every node periodically reporting their capabilities, configuration, and operational status to relays
- Relays forwarding information to a server
- A server storing the information in a configuration database [9].

Configuration information database. The configuration information database can be centralized or distributed depending on the type of network under consideration and its requirements. It is used for storing and accessing information related to the configuration status of the network.

Robustness is a common concern for approaches relying on a single central server. Although IPAS uses only one central server at a time per domain, it can elect a new ACA in a distributed manner using a periodic beacon protocol [10] if the current ACA becomes unavailable. Moreover, after initial configuration, ACA is only used for optimization, while distributed mechanisms (DCDP, DRCP) maintain correct configurations. For example, the IP address pools are distributed to all nodes in the network, enabling DCDP to configure and reconfigure nodes without any help from the ACA. This distributed configuration also allows IPAS to handle higher levels of dynamics than approaches based only on centralized servers.

Configuration Algorithms

This section describes a few algorithms run in modules of the IPAS for address distribution and hierarchy generation and maintenance.

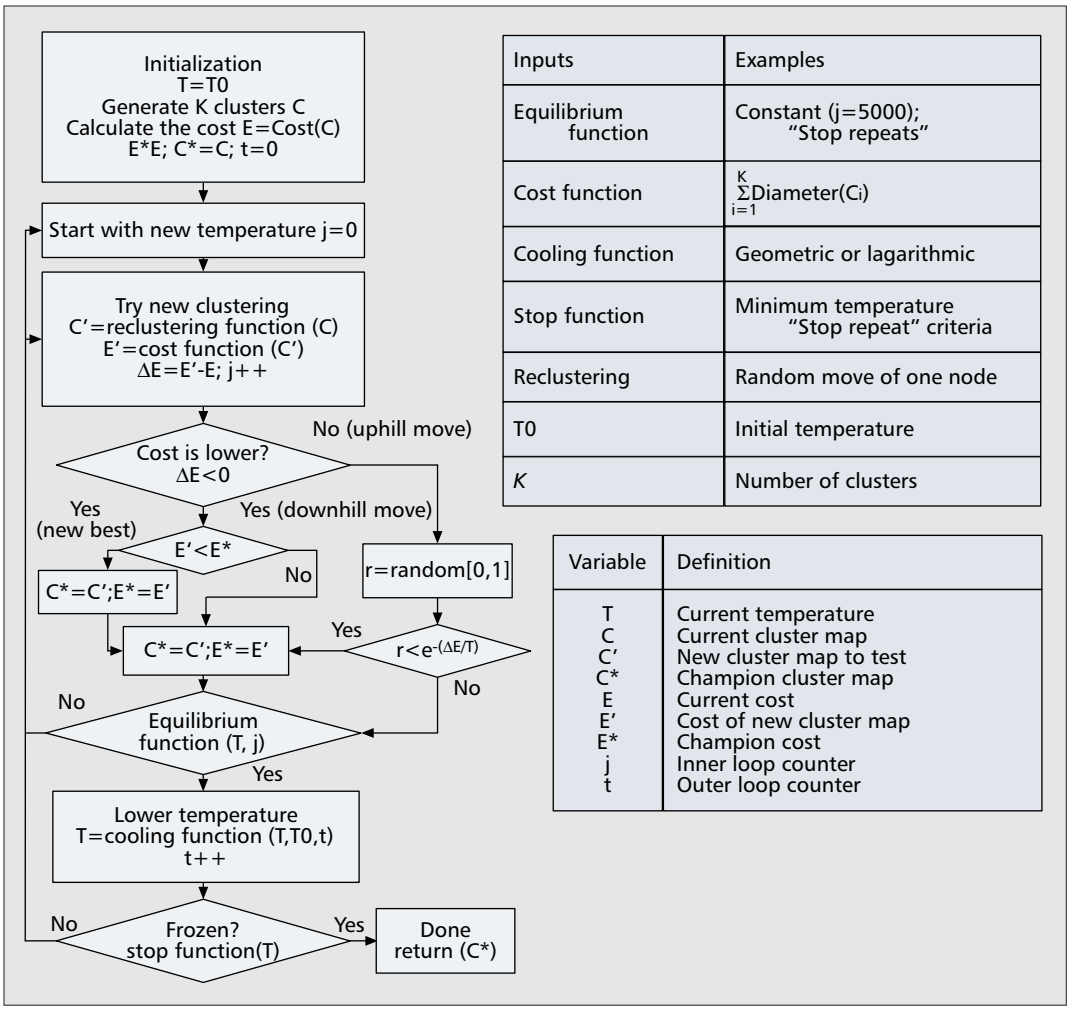
DCDP Address Pool Splitting Algorithm

Each DCDP process can split the address pool. If a DCDP process receives a request for addresses from DRCP (or from a protocol like DHCP), then its default is to offer a pool of addresses from a contiguous set (e.g., 256 address from a "/24" prefix). The contiguity of the offered pools results in smaller routing tables and better routing aggregation. If the request has been initiated from a neighboring DCDP process, DCDP defaults to split its pool of available addresses into two contiguous address pools of binary (power of two) and as similar as possible size. One is kept locally and the other is offered. This binary splitting approach distributes the available addresses, so partitioned networks can continue to operate with the address pools they possess. This mechanism ensures the conflict free assignment of addresses, so a DAD algorithm is not required, which makes the approach simple and scalable.

This simple distributed splitting algorithm is complemented by ACA algorithms to manage the addresses. Initially ACA distributes the address pools to ensure good aggregation in the routing domain (e.g., OSPF areas). DCDP only distributes addresses within each routing domain. Ideally, each routing domain can then advertise its domain using a single IP address prefix. ACA also performs the function of reclaiming "lost addresses" and ensures that good address aggregation is maintained.

ACA Hierarchy Generation

Even though the generation of hierarchy has been shown to be beneficial for the performance and scalability of large dynamic networks, it is not an easy task to accomplish. The automatic and robust organization of a large dynamic network into a hierarchical structure consists of two phases: hierarchy generation and hierarchy maintenance. In our approach [11], the generation uses knowledge of the entire network to achieve optimality with respect to a set of prespecified objectives (thus differing from most existing approaches described earlier). The modules that constitute the proposed hierarchy generation framework are:



Inputs	Examples
Equilibrium function	Constant (j=5000); "Stop repeats"
Cost function	$\sum_{i=1}^{K}$ Diameter(C_i)
Cooling function	Geometric or lagarithmic
Stop function	Minimum temperature "Stop repeat" criteria
Reclustering	Random move of one node
T0	Initial temperature
K	Number of clusters

Variable	Definition
T	Current temperature
C	Current cluster map
C′	New cluster map to test
C*	Champion cluster map
E	Current cost
E′	Cost of new cluster map
E*	Champion cost
j	Inner loop counter
t	Outer loop counter

Figure 3. *The simulated annealing algorithm for network partitioning.*

The IP address pools are distributed to all nodes in the network, enabling DCDP to configure and reconfigure nodes without any help from the ACA. This distributed configuration also allows IPAS to handle higher levels of dynamics than approaches based only on centralized servers.

Optimization algorithm: simulated annealing (SA). SA [11] has been widely used to tackle different combinatorial optimization problems. The process of obtaining the optimum configuration is similar to that followed in a physical annealing schedule. Figure 3 highlights the general steps in the algorithm. The process starts with an initial temperature value, T_0, which is iteratively decreased by the cooling function until the system is frozen (as decided by the stop function). For each temperature, the SA algorithm takes the current champion configuration C^* and applies the recursive function to obtain a new configuration C' and evaluates its cost, E'. If E' is lower than the cost of the current E^*, C' and E' replace C^* and E^*. Also, SA, utilizing the Metropolis criterion, randomly accepts a new configuration C' even though E' is greater than E^* to avoid local minima. In the latter case C' and E' replace C^* and E^*, respectively. We modified SA to obtain a two-level hierarchy configuration composed of K domains. The cost of a particular configuration C^* is associated with a specific cost function that matches the objectives of the network. To make the algorithm run in real time on thousands of nodes we traded off a small loss in optimality for a dramatic reduction in speed of convergence. Specifically, we adjusted the:

- Cooling schedule: We applied a faster cooling schedule (geometric instead of logarithmic) at the risk of converging into a suboptimal solution.
- Termination condition: The algorithm terminates when the same solution is observed for a *StopRepeat* iterations. By simulation analysis we selected a *StopRepeat heuristic* such that the quality of the solution is minimally degraded.
- State transition probabilities: In each iteration of the algorithm a new domain map is generated. Instead of using uniform distribution to generate new solutions, we made the generation phase aware of the targeted objectives.
- Initial solution: The algorithm is based on randomly searching the surface of feasible solutions. This search begins from an initial solution that is obtained randomly. In this work we have shown that if the initial solution is better than a random one, the algorithm with high probability will converge faster.
- Feasible solutions generation mechanism: In each SA iteration new feasible solutions are generated to be evaluated against the currently optimal one. The mechanism that generates these solutions is the dominant factor in the speed of convergence of the algorithm. Carefully selected neighborhood structures, data structures, and functions were introduced to

Although the maintenance of the hierarchy quality over time is proportional to the information available, a random maintenance scheme is reasonably good. Thus we can adapt the maintenance algorithm to utilize more or less information depending on the goals and the characteristics of the network.

Class	Objective #1	Objective #2	Objective #3	Multi-objective
Physical domain characteristics	Balanced and limited sized	Balanced and limited diameter	Minimum number of border routers	Objectives #1 and #3
Interdomain node mobility	Nodes move in a similar direction	Nodes move with similar velocity	Large link expiration times	Objectives #1 and #2

Table 1. *Example hierarchy generation objectives for each domain.*

reduce as much as possible the impact of the mechanism on the convergence time of the algorithm without affecting the quality of the final solution.

• Cost functions. The most important part of the optimization is defining the hierarchy generation objectives. The general framework proposed here (IPAS) and the use of SA provide us with the flexibility to select any combination of objectives. Table 1 presents some typical objectives that we have selected for hierarchy autoconfiguration. One class targets the physical characteristics of the generated hierarchy (e.g., diameter of the generated OSPF areas) and the second class is related to node mobility characteristics (e.g., robust hierarchy). These objectives are translated into cost functions [11] that depend on parameters collected from the network in real time. In many cases we found that using multiple objectives simultaneously, instantiated in complex functions, provides the best performance [11].

• Constraints. During the optimization of these cost functions from the SA algorithm, not all hierarchy maps are acceptable (feasible). These limitations are incorporated to the algorithm as optimization constraints. Such constraints limit the search space of SA. We have enforced that all members of a domain must communicate among themselves without the need to use external links, which are links that involve non-member nodes.

DRCP Hierarchy Maintenance

Topology changes, due to node mobility or link failures, cause nodes to become disconnected from their original domain. Those nodes that lose connectivity to their domain must reassociate with any of their neighboring domains. If the disconnected nodes have more than one neighboring domain, a decision must be made on to which domain to connect. A localized (distributed) approach is preferred for speed and reduced overhead. However, since the maintenance algorithms utilize local information, the quality of the hierarchy degrades. The rate of quality degradation depends on a) the effectiveness of the generation phase on producing "good" quality hierarchy, and b) the effectiveness of the maintenance phase in maintaining the quality of the hierarchy. Four classes of local maintenance are:

- **(A0): Random**, which does not use any metrics but randomly assigns nodes to neighboring domains
- **(A1): Independent**, which uses metrics unrelated to hierarchy generation objectives (e.g., IDs of neighboring domains)
- **(A2): Node-dependent**, which uses similar metrics to the hierarchy generation algorithm, but with metrics collected only from one-hop neighbors
- **(A3): Domain-dependent**, which uses similar metrics to the hierarchy generation algorithm like A2, but relies on the characteristics of all the neighboring domains

The better the maintenance of the hierarchy quality, the more prolonged the time interval for reapplying the hierarchy generation mechanism to reconstruct the hierarchy quality. Although maintenance of the hierarchy quality over time is proportional to the information available, a random maintenance scheme is reasonably good. Thus, we can adapt the maintenance algorithm to utilize more or less information depending on the goals and characteristics of the network.

The use of priority and a configuration timer minimize the overhead due to transient effects of having parallel application of maintenance and generation mechanisms. The reapplication of the hierarchy generation mechanism is triggered from the quality level (cost) of the maintained hierarchy. Since the continuously applied maintenance happens locally, the nodes are not aware when the generation mechanism will reconfigure the hierarchy. Thus, there may be transient effects until the generation decisions reach the participating nodes. Simple transient effects are eliminated by assigning highest priority to the new configuration. A timer at the node, which has been configured from the ACA, also ensures local maintenance will not immediately override the ACA decision because some of the nodes' neighbors have not yet received the new domain decisions. The transient effects may be more complicated to handle if the ACA configuration decisions are not delivered reliably to all nodes. In this case the timer will expire, and local maintenance will take over. If the local maintenance results in a low-quality (high-cost) hierarchy, the generation mechanism will be retriggered, distributing new configuration decisions.

Results

This section presents large-scale simulation analysis results, and the implementation of the autoconfiguration algorithms and protocols in a small testbed.

Small Network Testbed

To study the feasibility of our integrated approach to network and domain configuration, we implemented the IPAS modules in a labora-

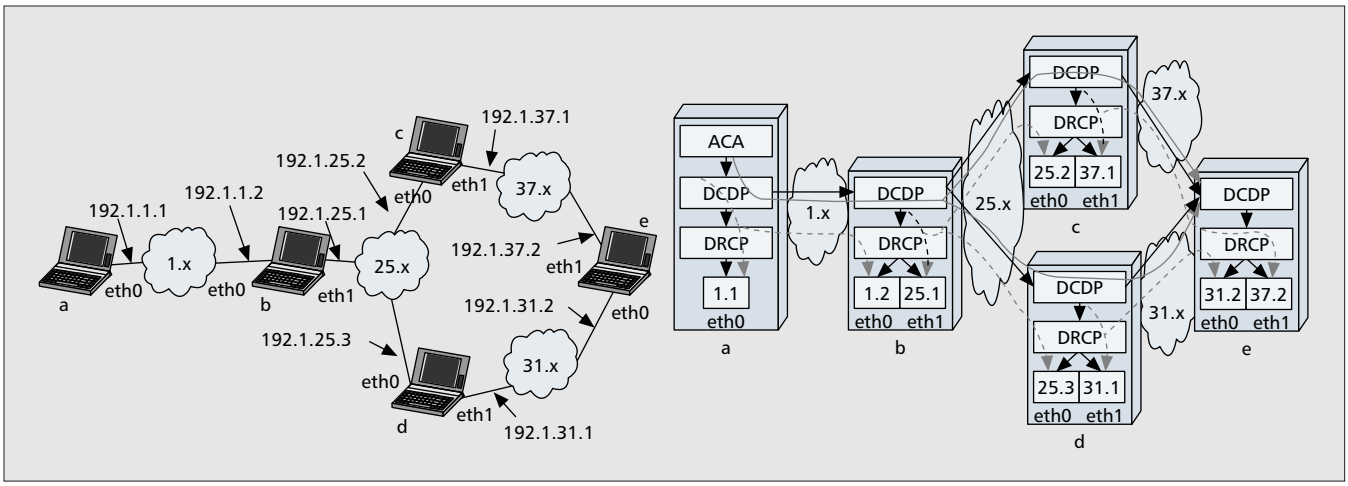


Figure 4. *The network autoconfiguration testbed and IPAS message flow.*

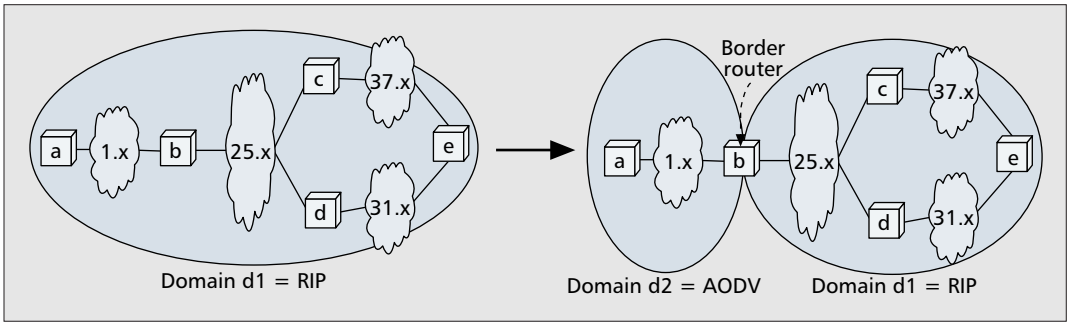


Figure 5. *Domain autoconfiguration for the generation of routing domains.*

tory testbed. We connected five Linux laptops, each equipped with two 802.11 cards, as shown in the left half of Fig. 4, and set up a scenario to validate the capabilities of IPAS. We demonstrated IP address configuration, dynamic routing protocol configuration, and dynamic hierarchy configuration. Node a performs the ACA function, initiating the distribution of IP addresses (initial pool was 192.1.1.1–192.1.38.255), domains, and other configuration information. The right half of Fig. 4 shows how the interfaces and subnets get configured, and the flows between the IPAS modules. The DCDP modules distribute the pool of addresses, and the DRCP modules utilize the pool of addresses obtained from DCDP to configure interfaces in their subnets.

Initially, when the configuration information is distributed, there is no routing protocol running. The ACA decides, based on the information collected from the network through YAP, on the hierarchies and which routing protocol to use in each domain. It is possible that the ACA decides to run different routing protocols in each domain. The left side of Fig. 5 shows that ACA initially configures the network into a single routing domain (d1) running Routing Information Protocol (RIP) (from the zebra package) because it sees stable links. After the routing stabilizes, we are able to transmit a video over the three hops from node a to node e.

We next emulated a degraded connection between node a to node b, which caused ACA to decide to split the original d1 domain into two domains, d1 and d2 (Fig. 5). ACA, through DCDP and DRCP modules, dynamically reconfigured the routing protocol running on nodes a and b from RIP to Ad Hoc On Demand Vector (AODV) routing (NIST kernel module). Node b dynamically detected it had two interfaces in two different domains, and thus ran a border router between those domains. The video transmission paused only for a few seconds during the time the kernel modules were inserted and the routing tables converged.

Based on extrapolation from experimental testbed results, Fig. 6 shows the IPAS configuration time and overhead in a single domain as a function of the number of nodes in the network. It shows the configuration time (primarily due to DCDP) for a typical "distributed network" configuration (with subnets connected in a mesh pattern) grows relatively slowly with the size of the network. The configuration distribution overhead is small (under 2 kbytes per link), since the information is essentially sent on a spanning tree, and the DCDP and DRCP headers and configuration information have been carefully optimized. The periodic overhead (primarily due to YAP and DRCP periodic advertisements) grows more rapidly, but can be contained to reasonable levels by limiting domains to have under 100 nodes and refreshing network metrics at most every 30 s.

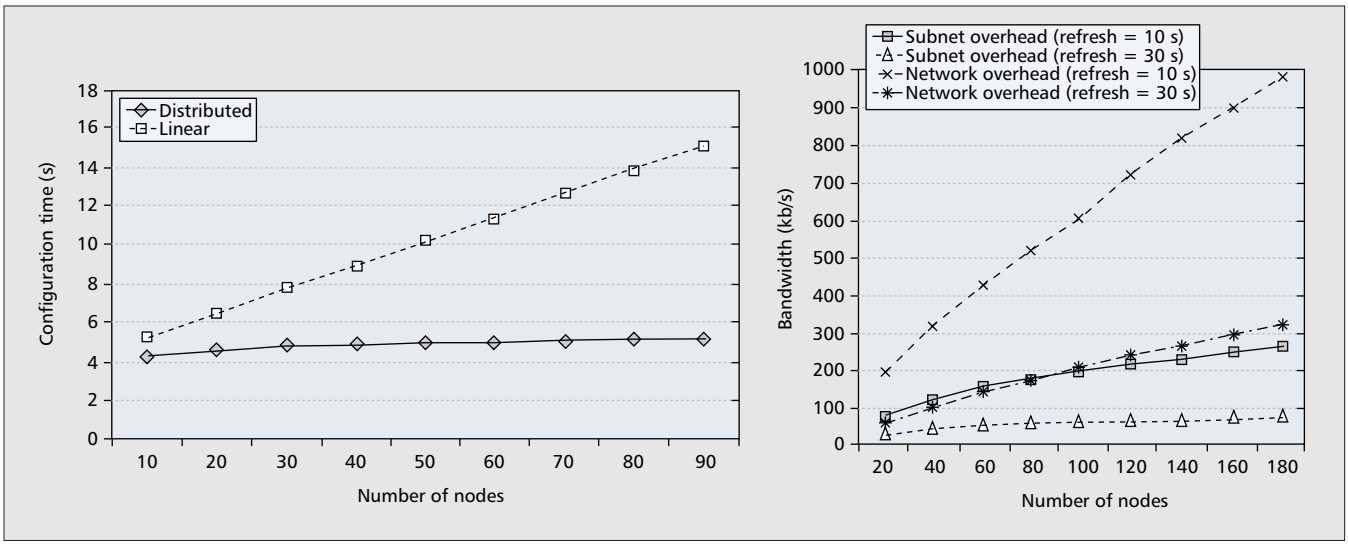


Figure 6. *IPAS configuration time and aggregate overhead.*

(*n*,*K*)	(100,2)	(200,2)	(100,4)	(200,4)	(100,5)	(200,5)	(100,10)	(200,10)
Initial cost	2880000	6.2E+08	310243	713621	670964	7362245	9811	235088
Solution cost	0	0	0	0	13	64	146	228

Table 2. *Improvement on the initial cost provided from the modified SA.*

Simulations of the Enhanced Simulated Annealing Optimization

Running the SA optimization algorithm on a Pentium 4 machine with a 1 GHz processor and 512 Mbytes RAM on a large emulated network, we investigated the SA performance. Compared to the original SA, our enhanced version decreased the convergence time from 30 min down to 19 s for networks of 1000 nodes. A 100-node network could be optimized in 100 ms. The quantitative results given above are for average node degree of 5. When the node degree increases, the convergence time decreases as the SA generation mechanism becomes faster. For average node degree 10, the convergence time for 1000 nodes drops to 7 s and for 100 nodes drops to 45 ms.

The effectiveness of the enhanced SA depended mostly on the number of generated domains. The smaller the number of generated domains, the higher the probability of obtaining the globally optimal solution (e.g., for generating four domains the algorithm always obtains the global optimal; for 10 domains the percentage is 64 percent). This is due to the size of the solution space, which grows as the number of generated domains gets larger. An important observation, however, is that even in cases where the algorithm does not obtain the global optimal, the cost of the solution is close to the global optimal cost. Table 2 shows the goodness (low cost) of the enhanced SA solutions in eight example networks, with K generated clusters and n nodes. In all cases high-quality (low-cost) hierarchical solutions (global optimal cost is normalized to 0) are obtained, compared to the initial random (but feasible) solutions, which are generally of very low quality (high cost).

Conclusions

This article presents a unified framework for completely autoconfiguring hosts, routers, and servers. Based on the IP Autoconfiguration Suite, it is a powerful configuration tool for future networks, especially when applied in emergency situations or dynamic large-scale networks. IPAS performs robust and conflict-free address allocation, based on the distribution of address pools into automatically generated domains. Address management ensures that the routing protocol can take advantage of aggregation at domain boundaries. The entire configuration does not merely configure the network based on some simple heuristics; rather, it optimizes network performance. Maintenance of configuration is performed in a distributed manner, so it can reach fast to topology changes and be robust to loss of nodes (like the ACA). Our testbed shows the first combination of dynamically configuring and reconfiguring addresses, routing protocols, and routing hierarchies.

Acknowledgments

The authors would like to acknowledge the following for their significant contributions in the development of the ideas and implementation of IPAS: Dana Chee, Jason Chiang, Subir Das, Sunil Madhani, Archan Misra, Sunil Samtani, Larry Wong, and Ken Young. Also, we would like to acknowledge the anonymous reviewers for their useful comments and suggestions.

REFERENCES

[1] C. E. Perkins *et al.*, "IP Address Autoconfiguration for Ad Hoc Networks," Internet draft, Nov. 2001.
[2] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration," IETF RFC 2462, Dec. 1998.
[3] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," *Proc. INFOCOM '02*, pp. 1059–68.
[4] K. Weniger, "PACMAN: Passive Autoconfiguration for Mobile Ad Hoc Networks," *IEEE JSAC*, Special Issue on Wireless Ad Hoc Networks, Mar. 2005.
[5] H. Zhou, L. Ni, and M. Mutka, "Prophet Address Allocation for Large Scale MANETs," *Proc. INFOCOM '03*.
[6] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proc. IEEE*, vol. 75, no. 1, Jan. 1987, pp. 56–73.
[7] M. Gerla, J. T-C. Tsai, "Multicluster, Mobile Multimedia Radio Networks," *Wireless Networks 1*, 1995, pp. 255–65.
[8] B. McDonald and T. F. Znati, "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks," *IEEE JSAC*, vol. 17, no. 8, Aug. 1999.
[9] A. McAuley *et al.*, "Experience with Autoconfiguring a Network with IP Addresses," *IEEE MILCOM*, Oct. 2001.
[10] R. Morera, A. McAuley, and L. Wong, "Robust Router Reconfiguration in Large Dynamic Networks," *MILCOM*, Boston, MA, Oct. 2003.
[11] K. Manousakis, A. McAuley, and R. Morera, "Applying Simulated Annealing for Domain Generation in Ad Hoc Networks," *ICC '04*, Paris, France, 2004.

BIOGRAPHIES

KYRIAKOS MANOUSAKIS [StM] is currently a Ph.D. candidate in electrical and computer engineering at the University of Maryland, College Park. He received with honors his Diploma in electronics and computer engineering from the Technical University of Crete (1998) and his M.S. in electrical and computer eengineering from the University of Maryland, College Park (2002). He designs, evaluates, and implements protocols for wireless communications, reliable multicasting, network autoconfiguration, and security.

ANTHONY MCAULEY received his Ph.D. from Hull University, England, in 1985. He was a research fellow at California Institute of Technology from 1985 to 1987. Since 1987 he has been at Telcordia and is currently a chief scientist in the Wireless Research group. He helped create and works on projects related to wireless and ad hoc networking, particularly autoconfiguration, routing, mobility, security, and QoS. He has built several mobile internetworking systems on Linux, and designed efficient error codes and VLSI chips.

RAQUEL MORERA received her Ph.D. (2001) in mobile communications from Strathclyde University, Scotland, and her M.S. (1996) in telecommunications engineering from Universidad Politecnica de Valencia, Spain. In August 2001 she joined Telcordia as a research scientist in the Wireless and Mobile Networking group. Since then she has worked on the design of autoconfigured networking solutions for highly dynamic ad hoc networks. Other research interests include analytical analysis of ad hoc networks, security, and QoS.

JOHN S. BARAS [F] received a B.S. in electrical engineering from the National Technical University of Athens, Greece, in 1970, and M.S. and Ph.D. degrees in applied mathematics from Harvard University in 1971 and 1973. He was the founding director of the Institute for Systems Research (one of the first six NSF Engineering Research Centers) from 1985 to 1991. Since August 1973 he has been with the Electrical and Computer Engineering Department and the Applied Mathematics Faculty at the University of Maryland, College Park. In February 1990 he was appointed to the Lockheed Martin Chair in Systems Engineering. Since 1991 he has been director of the Center for Hybrid and Satellite Communication Networks (a NASA Commercial Space Center). He has received several awards for his research. He has published numerous technical articles and edited one book. He has consulted extensively with industry and government on various automation and telecommunication problems. He has served on several editorial boards. He holds two patents and has four pending. His research interests focus on control, communication, and computing systems.

IPAS performs robust and conflict free address allocation, based on the distribution of address pools into automatically generated domains. Address management ensures that the routing protocol can take advantage of aggregation at domain boundaries.