Toward a More General Network Layer (extended abstract)

Kenneth L. Calvert
Leonid Poutievski and James Griffioen

October 2004

We consider the design of a very general network layer, providing a service that subsumes unicast, multicast, publish-subscribe, and mobility, all using a single framework. Here, "network layer" refers to a set of forwarding algorithms and routing algorithms that work together to provide a packet-forwarding service over a given, arbitrary, fixed topology (interconnection of nodes and links). A forwarding algorithm takes as inputs forwarding state and packets that carry control information (headers), and produces as output a list of interfaces on which the packet is to be forwarded, possibly with modified control information. A routing algorithm takes as input control information that is carried in packets, and updates the state used by the forwarding algorithm, in such a way that packets are forwarded efficiently toward their destinations. The network-layer service we consider is called *speccast*. Speccast is based on the use of predicates to specify packet destinations. That is, each packet carries a predicate, and the network layer is required to deliver the packet (best-effort) to all nodes that satisfy the packet, and to as few others as possible. The Internet is a (weak) special case of this service, in which each unicast address is a predicate satisfied by at most one node, and predicates are tied to location in the topology. The practical utility of such a service depends on the set of predicates and how they are encoded and handled by the network layer.

Our approach to solving the speccast problem is based on link-state data exchange. Each node's forwarding state is a graph or map of the network topology, in which nodes are labeled with predicates. Nodes exchange information about the predicates they satisfy, as well as their attached links. Each packets is *source routed*: the sending node's network layer determines which nodes in the graph satisfy the packet's destination predicate, places the tree of *links to be traversed* in the packet, and transmits the packet over the initial link(s). Each subsequent node forwards the packet over the next link(s) in the tree. This problem formulation and solution differ from prior work such as NIMROD, overlay networks, and publish-subscribe systems, in that no underlying routing/forwarding service is assumed; no predefined address assignment is required; and no resolution-discovery step is needed before forwarding.

As described, this approach is not scalable: every node must know the entire network topology graph. Therefore we abstract the forwarding state at each node by aggregation: subgraphs are collapsed into nodes, and the predicate labeling the aggregate node is formed by taking the disjunction of the predicates labeling the nodes in the subgraph. Each node

Dagstuhl Seminar Proceedings 04411
Service Management and Self-Organization in IP-based Networks
http://drops.dagstuhl.de/opus/volltexte/2005/84

on the border of an aggregated subgraph performs this abstraction operation before emitting its routing control information across any link leaving the subgraph. Border nodes also expand the source routes of packets on their way into the aggregated subgraph; the result is a hybrid source-routed/dynamic forwarding system. By themselves these measures are not sufficient to make the system scalable, because in the absence of other constraints, the aggregated predicates can grow linearly with the size of the aggregated subgraph. To improve scalability, we allow lossy compression of aggregated predicates. As a simple example, consider three nodes labeled with predicates $(a \wedge b \wedge c)$, $(a \wedge b \wedge \neg d)$, and $(a \wedge c \wedge d)$, respectively. We might label the subgraph containing these three nodes with the predicate a. This would result in some "overdelivery"—i.e., packets being delivered to the aggregated node that should not be. For example, a packet labeled with destination predicate $a \wedge \neg b \wedge \neg d$ would be delivered to the aggregate node, even though no node inside the aggregate satisfies that predicate.

The main subject of our ongoing work is the design of a suitable predicate set, and the relationship between its structure and the efficiency of compression. We are working toward a rigorous definition of *locality*, to quantify the efficiency of an assignment of predicates to nodes in a topology. Among other important issues are the cost of determining which nodes satisfy a destination predicate (a poorly-chosen predicate set could make this task NP-hard in the size of the destination predicate), and how return paths are specified.

The "self-organizing" aspect of speccast relates to the dynamic formation of aggregates in the network according to topology, the predicates satisfied by nodes in the area, and local policies. In speccast, the framework *isolates scalability issues* from the design of the routing and forwarding mechanism itself. This raises the possibility of a single framework that supports policy in various forms and provides multiple services, all with run-time overhead roughly comparable to the existing Internet.

(This work is being supported by the US National Science Foundation under grant CNS-0435272.)