LANES: An Inter-Domain Data-Oriented Routing Architecture

Kari Visala
kari.visala@hiit.fi

Dmitrij Lagutin
dmitrij.lagutin@hiit.fi

Sasu Tarkoma
sasu.tarkoma@tkk.fi

Helsinki Institute for Information Technology HIIT / Helsinki University of Technology TKK

ABSTRACT

Data-oriented networking has attracted research recently, but the efficiency of the state-of-the-art solutions can still be improved. Our work towards this goal is set in a clean-slate architecture consisting of modular rendezvous, routing, and forwarding functions. In this paper we present the inter-domain routing layer and its interplay with the other components of the system. The proposed system is built around two types of nodes: forwarding nodes and branching nodes. The forwarding nodes are optimized for throughput with no per-subscription state and no need to change passing packets, while branching nodes contain a large memory for caching and can make complex routing decisions. The amount of storage space and bandwidth can be independently scaled to suit the needs of each network. In the background, topology nodes perform load-balancing and configure routes in each domain using a two-dimensional addressing mechanism. The paths taken by packets adapt to the number of active subscribers to keep the amount of in-network state and latency low. A new data-oriented congestion control scheme is introduced, which takes into account the use of storage resources on-path and is fair to multicast flows.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*
; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design

Keywords

Data-oriented networking, Internet architecture, routing, caching, congestion control

ReArch'09, December 1, 2009, Rome, Italy.

1. INTRODUCTION

The Internet is about sending messages to endpoints but the bulk of the current traffic is generated by applications requesting and disseminating identified data objects. Such incompatibility causes inefficiency both in the network and the design of the applications. This has recently stimulated research [15, 10, 3] on redesign of the Internet based on data-oriented concepts. Meanwhile, the problem is partially addressed by peer-to-peer (P2P) overlays like BitTorrent and content delivery networks (CDN) often collocated with ISPs' networks.

The benefits of the data-oriented communication model include the possibility to trade storage space for lower average latency by caching content locally and support for multicast that removes the sender-side bottleneck for popular data thus scaling to flash crowds. It is also cost-effective for ISPs to cache a data object instead of using global transit to download it if the item is requested frequently enough.

When data-orientation is implemented on the network layer, caches can be placed hierarchically on-path, routing designed for data retrieval, and topology information with traffic measurements are available for optimization. Compared to the higher-level approaches, the additional layer of functionality required by an overlay is not needed increasing the efficiency and decreasing the complexity of the system. P2P overlays exhibit complex security and incentive problems as the users cannot trust each other and the high node churn causes problems for their efficiency. On the other hand, CDN-based solutions are generally not designed to interwork, traditionally have been operated as service for the content providers and therefore do not fully adapt to the demand generated by subscribers[1], and are currently limited by web application semantics. In addition, when the network layer is based on receiver-driven communication, it is possible to eliminate unwanted payload traffic which alleviates the effects of DDoS attacks.

In this paper we propose a new clean-slate routing architecture LANES (stands for *Logical Address space Network Extensions*) that aims to solve the data-oriented routing problem for the Internet efficiently. By this we mean using a near-optimal amount of state in the network, low stretch, and sharing of the link and storage resources with the goal of maximizing the aggregate utility. Inter-domain environment poses other main challenges: scalability, incentive compatibility, security, inter-domain policy compliance, heterogeneity, and evolvability. We briefly discuss our de-

[1]Although, this is changing as ISPs are building their own caches to improve the service for their customers.

sign choices against these challenges when they are relevant in the following sections.

We have adopted ideas from multiple sources: self-certifying names from [17, 15], our method of minimizing the subscription soft-state in the network resembles the path collapsing used in SCRIBE [5], the idea of having simple and more complex nodes mixed in the network is used in CDNs, two-dimensional addresses are used in source-specific multicast [2], the idea of separating routing from forwarding from [7], multicast trees were built from chained control loops in [24, 16], it was recognized in [14] that data requests may also require scheduling, and the congestion control algorithm was adapted from the multirate *layered multicast* [18] congestion control analyzed in [23].

Our contribution is applying these techniques to the data-oriented routing. The system includes a novel, receiver-driven scheduling algorithm that takes into account the finite resources consumed by subscription soft state in the network in addition to *weighted proportionally fair* [12] link allocations and allows multicast to be used either synchronously or asynchronously. In contrast to layered multicast, subscribers can switch to parallel unicast segments in the delivery tree when there is shortage of cache space. More recently, erasure resilient codes were used to achieve multiple rates with small caches in [16]. This could be included in our system, but as our approach relies on large content caches to minimize average delay, it is uncertain whether the advantages exceed the added coding overhead. Caches also make the solution more flexible than simple *replicated multicast* from the source. The congestion control algorithm prevents the parallel unicast connections from reducing the total system utility by slowing down more efficient multicast connections too much. We also generalize source-based and shared tree multicast approaches (for example, [1]) into a single system where the source can choose the delivery tree shape on the uphill side of the *valley-free path* [9] so that the tree remains incentive compatible in the Internet-like inter-domain structure (the problem is explained in [19] for caches) while serving subscribers at the rate managed by the source.

Section 2 introduces the basic concepts, in Section 3, we describe the general architecture, the inter- and intra-domain structures are explained in Section 4, Section 5 shows how delivery trees are constructed, congestion control is covered in Section 6, and finally, we conclude by pointing out future work in Section 7.

2. NAMED DATA VALUES AND HOLES

The basic primitive of our architecture is a named, finite, immutable data object, or a *publication*. *Publishers* create new data objects and *subscribers* can request the values of the data objects by their name. A data object is conceptually almost the mirror-image of the session state of a dialog that expects input - a kind of "hole", which is straightforwardly implemented using message-passing. A similar symmetry exists in programming languages between values and *continuations* [8], which points to a possible direction for building a data-oriented network API. We argue based on this supposed symmetry that data-oriented and dialog-based communication models complement each other.

Data objects are identified with flat names similar to DONA. The name consists of <P,L> pair, where P is a public key trust anchor for the namespace of the name and L is an arbitrary binary string label possibly encoding text. The tussle [6] for human-readable global names is left for extra-network mechanisms by the namespaces. Labels can carry semantic information for applications, but the network should not interpret them as such mechanisms will be biased towards some applications and violate the end-to-end (E2E) principle [20]. We call the full, variable length name a *resource identifier* (RId). Data objects are signed with the publisher's public key certified by the namespace. It is also possible to sign data directly with the P key, but for dynamically created publications the extra indirection by a chain of temporary certificates allows to keep the namespace key offline and the identities can have a long life-time. Figure 1 depicts the relationships between the concepts.

The data object identified by a name may not exist at the time of subscription. For example, the label could be "snapshot of a web camera X at 12 o'clock" and it is not yet noon. In this case the network returns the object immediately after it has been published. The data values can also be dynamically generated, in which case the label could represent a nullary function, for example, "weather(Helsinki, Aug 11th 2009)". This is also the reason, why the full label information have to be communicated to the data source. Data values are also divided into smaller segments that can be individually requested.

All state is soft state in the LANES network and subscriptions need to be refreshed periodically and unused cached objects are automatically garbage collected. Separate entities at the end points of the network are responsible for storing the data persistently. This adheres to the principle of fate sharing [4] and makes the architecture more modular.

3. ARCHITECTURE

In Figure 1, the three layers of our architecture [21] are shown: rendezvous, routing, and forwarding. The rendezvous system consists of *rendezvous networks* that could be implemented as DONA systems and a hierarchical DHT-based *rendezvous interconnect* that joins the individual networks globally. *Data sources* function as the persistent storage for publications at the edge of the network. They use the rendezvous system to advertise their contents and subscribers can then request the data source locations for a given object. The subscriber uses the result of the renderzvous as an input to the routing function by sending a *subscription packet* which the network routes towards a nearby copy of the requested data. As the subscription packet travels in the network, it gathers a *forwarding identifier* (FId) that can be used by the data source or an intermediate cache to send the payload data via a simple and fast forwarding fabric back to the subscriber. We emphasize that this whole sequence of operations can be local for public data. For FIds, we use source-routed zFilters explained in [11]. Both the routing layer and the forwarding layer utilize statistical multiplexing of links and caches and only offer *best effort* semantics.

Because the named publications are abstract entities, we need a mechanism to tie them to concrete implementations. For this, we introduce the concept of *scope* that represents a *dissemination strategy*[2] for a set of objects. Scope controls, among others, access rights of data sources and subscribers, location, reachability, availability, replication, persistence, and upstream resources of a publication. Scope owners au-

[2]A term that we snatched from Van Jacobson's talks.

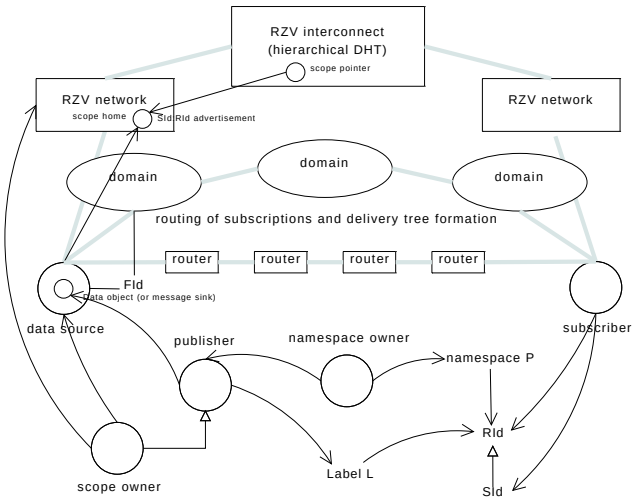

Figure 1: The architecture consists of rendezvous, routing, and forwarding layers. The abstract data values identified with RIds can be placed in scopes, that determine how they are distributed.

thorize data source(s) to advertise offered publications in the scope and a trusted rendezvous network is authorized to advertise the scope in the rendezvous interconnect. Scopes are identified using a special type of RId, a *scope identifier* (SId). When a subscriber wants to receive a publication, she has to provide the network with a <SId,RId> pair. Subscriber should request the publication from a scope that she trusts to deliver the data in the correct way. Data integrity, on the other hand, can be checked directly from the data object. Some scopes can encrypt the contents of the publications and/or require encryption of the subscribed labels. This causes some inefficiency as the network may have to cache the same data multiple times from different sources. The required capabilities to encrypt/decrypt can be delivered to subscribers in the response from the rendezvous system encrypted with the subscriber's public key delivered in the request message, but this requires the rendezvous to happen in a *home rendezvous network*[3] of the scope, which may be subject to DDoS attack and prevents caching responses. Rendezvous networks are modular entities that can provide different services for scopes and data sources. While the routing layer is not allowed to interpret RId labels, rendezvous networks can use them to select the data source. The scopes can mark the rendezvous responses uncacheable to force all rendezvous requests to reach one of the home rendezvous networks of the scope.

In a typical usage scenario, the subscriber wants to subscribe to many publications from the same scope in sequence. It would be wasteful to perform the rendezvous operation for every subscription and therefore the subscriber can first try to opportunistically request the publications directly from a known data source.

[3]Each scope can be replicated in multiple rendezvous networks.

4. DOMAIN STRUCTURE

We assume that the network is composed of independent domains, which typically have bilateral contracts with the adjacent domains for using their networks. The domains can be virtual and share physical routers with an underlying domain. The domains exchange *service offers* between their neighbours to signal which network services they provide. Each domain has a logically central *topology node* (TN) that stores accepted offers from the neighbours. The LANES network requires a BGP-like protocol to discover domain-level policy compliant paths to other domains as one such service. Such policies are complex and outside the scope of this paper. Inside a domain each node has an identity and they periodically update information about their immediate neighbours and send it towards the TN that manages the whole intra-domain topology.

Figure 2 shows a simplified topology of three domains, where domains A and B are customers of the one in the middle. In addition to TNs, each domain can have *forwarding nodes* (FN), *edge nodes* (EN), and *branching nodes* (BN). Links can be unidirectional and in general the subscription packets on the routing layer follow different path than the path they set up in the reverse direction on the forwarding layer. This makes it possible to use, for example, asymmetric satellite connections for payload data. It is only assumed that neighboring domains have at least one connection in both directions.

The only functions of the FNs is to route incoming packets with zFilter FIds by matching them to the outgoing link ids [11] and update the link traffic information to the relevant BNs. BNs are more complex and they perform the actual route selection for new subscriptions based on the information received from the TN and the traffic measurements, manage large caches, pass path price feedback packets from receivers to senders, and perform the branching of delivery trees when there are multiple simultaneous subscribers for the same data. This is achieved by switching the incoming packet zFilter for all downstream receivers with the next path segment FIds, which requires the BN to store subscription state for downstream branches. Unlike many IP multicast protocols, the state for active subscriptions is only stored in the nodes that currently branch, which reduces the in-network state as all state for single-receiver delivery trees is stored at the endpoints. BNs also check the integrity of the payload.

Each BN is responsible for only a subset of subscriptions. The load is balanced between them by assigning each BN a range of logical addresses derived from hashes of RIds. The TN of the domain performs the address space allocation based on load information received from BNs and the links. In order for the system to be scalable to extremely large groups of millions of simultaneous subscribers for the same data, domains can organize themselves hierarchically into subdomains.

ENs perform filtering of messages at the boundary of the domain for security. They also store an intra-domain routing table that is used to route the subscription packets to the correct BNs. This routing table implements a function of type *(destination domain,set of possible rendezvous domains,RId)* $\rightarrow$ *zFilter*. The routing information is computed by the TN and distributed to ENs. If the same BNs are shared for all egress links of the domain, this routing table is very small, but it is possible to also position the BNs

near the egress points which trades larger routing tables and worse cache utilization to better new subscription latency and less feedback traffic from FNs to BNs. TN also sends zfilters for each FN to send link and traffic information to BNs.

5. DELIVERY TREE FORMATION

Figure 2 depicts a new subscription operation starting from domain B. A subscription packet is sent along the path marked by arrows and it traverses recursively via the BNs responsible for the RId in every domain on the path towards the location of the data source until it finds an active subscription or cached copy of the data in some BN (BN2 in the figure). Each BN on the path computes a zFilter for the most efficient[4] new payload route from ingress to egress link back towards the domain where the subscription message was received from and concatenates the new path segment to the route by *OR*ing the added link ids with the existing zFilter carried in the subscription packet. The BN also computes the best local bypath from the computed route towards itself and *ORs* it to the zFilter to receive cache updates while not adding stretch to the payload path. In addition, the BNs accumulate the current explicit path cost and delay of the route in the subscription packet, which can be used to set the congestion window immediately to a good starting value in the sending node instead of using slow-start type of algorithm of TCP. In the example, BN2 can then start sending the content packets with the FId built from the zFilter. Payload packets identify the data with a short, statistically unique hash of the RId. If the cache contains the subscription only partially, the BN removes these segments from the subscription before passing it upstream. BNs also gather data about inter-domain connection costs from cache update rates and can perform traffic engineering based on this data for future subscriptions if multi-homed. The costs implicitly take into account the cache sizes of other domains. Intra-domain link failures are detected from missing FN updates.

Receivers can use the zFilter for identifying existing subscriptions in following update packets that are routed similarly to subscription packets to signal path price upstream. This method has two important advantages when combined with the zFilter forwarding: First, there is no need to modify the fast path packets with congestion information, and secondly, the false positive links and cache bypaths can be taken into account in the path price and cannot therefore cause congestion in off-path links. A subscription initiated by receiver r also contains two additional parameters: a *weight* w_r and the *duration* of the subscription. The duration tells the network how long the subscription should be stored as a soft-state and w_r signals how much the subscriber is willing to pay per unit time for receiving the data. This information can then be used to allocate a share of link and storage capacities for each subscription as explained in section 6. Each domain can then directly charge the neighboring domain where the subscription was received from the weight minus the current accumulated path cost to encourage truthful signaling of the utility.

Because the subscription packets are routed based on both the RId and the data source location information, we call this two-dimensional addressing. The location information

[4]based on link costs, path length or some similar metric

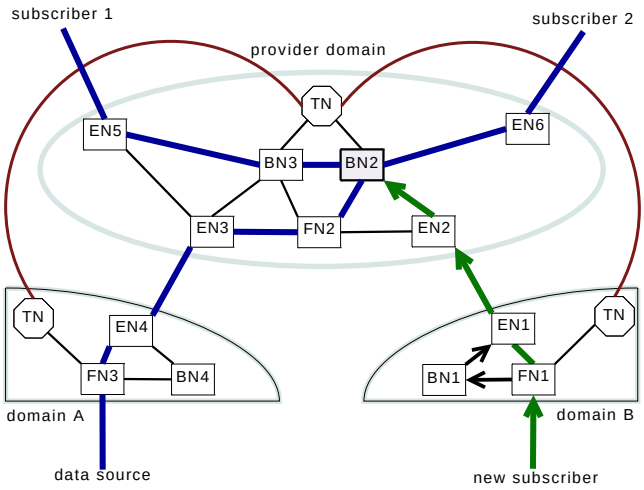

Figure 2: A new subscriber joins an existing delivery tree that branches at BN2.

contained in the subscription message is a <domain, intra-domain address> tuple, where the intra-domain addressing mechanism is specific to each domain. Based on this information, the BNs can choose the next AS towards the source. Multi-homing is possible and one possible way to optimize cache space over domain boundaries would be to use the RId to affect which provider is chosen.

Using a separate packet for each RId subscription has a high overhead, which is why we allow also to use the data source's identity as the hash to choose BNs on the path. This allows the bundling of multiple RId subscriptions in a single packet. However, if the same object is cached locally from some other data source, the network may not utilize it.

If the data object does not exist at the time of the subscription, the data source or upstream BN sends a positive ACK packet along the payload path to signal an advance booking of caches. This booking is cached in the BNs with a similar policy to other data. If another (re-)subscription for the same data arrives at the BN, it starts an active branching by storing the subscription locally as if it had the data to be sent. This logic makes stream type multicast possible. We note that using zFilter for the global multicast would not be feasible because of the limited group size and its single-rate operation. For popular static data, the simple caching at every level causes the implicit branching happen closer to the receivers and we could call it asynchronous multicast. As we move recursively closer to the data source, the probability of finding the data in the cache increases as the requests for the data converge.

Subscription packets optionally contain a set of domains accepted by the data source for upstream delivery as potential points of rendezvous[5] and the subscription packet must then be routed to the preferred rendezvous domain that can be reached by only the subscriber side compensation to the network. We had to add this "limitation" to the system as it is likely that the larger ISPs charge their customers also on the sender side for every branch and cache copy in its *upgraph* [25].

[5]This has nothing to do with the "slow path" rendezvous on the rendezvous layer.

6. CONGESTION CONTROL

Our point of view on congestion control is that it is a mechanism for resource allocation in a network. We assume that the subscribers receive utility from the network as a non-decreasing, strictly concave, and continuously differentiable function of their receiving rate producing *elastic* [12, 22] flows. An economically justifiable goal of the network is to maximize the sum of utilities of the subscribers while taking care that the ISPs have the incentives to participate in the protocol.

Each subscription uses multiple types of resources in the network: Both the subscription upstream and the payload downstream share links with other connections and at each active BN they consume storage space for storing active subscriptions and prevent some data from being removed from a cache. We assign a cost to each resource as a function of its current demand and the system tries to approach a balance of the utility generated by the active subscription and the sum of costs of the resources it is using. The links use the following formula to calculate their price per unit flow at time t:

$$p_l(t) = f_l(y_l(t)) \tag{1}$$

where p_l is the price to use link l, f_l is a *penalty function* that is a non-decreasing, continuous function and non-zero for large rates, and $y_l(t)$ is the aggregate rate of traffic on link l at time t. It is possible to choose the penalty functions such that system converges toward maximum aggregate utility in the unicast case [23]. We use a price function for the unit storage space per unit time at BNs that is a function of the current demand for the storage.

If multiple subscriptions are joined in a BN, the associated BN becomes an intermediate congestion controller between the upstream and the downstream segments of the tree and must divide the weights of the associated subscriptions to the different path segments. We base the algorithm on the multi-rate multicast case analyzed in [23] and the *primal algorithm* [13] to control the sending rate [6] at each BN or the data source. The differential equation for the sending rate on the downhill side of the valley-free path is given in (2).

$$x_r(t)' = k_r(x_r)(w_r - x_r(t)q_r(t) - c_r(t)) \tag{2}$$

where x_r is the controlled rate, $q_r(t)$ is the total unit price of the route segment r used as a function of time, $c_r(t)$ is the price of the lease of storage space required by the subscription at BN, and $k_r(x)$ is any non-decreasing, continuous scaling function such that $k_r(x) > 0$ for any $x_r > 0$. Without branching this corresponds to utility functions of the form $U_r(x_r) = w_r \log x_r$ and produces weighted proportionally fair allocation of links when penalty functions are properly chosen [13]. The active BNs try to keep the split of weights near an operating point where each subscription increases its weight only for the (recursive) upstream incoming flow if it also increases the downstream rate of that branch. Another requirement is that increase of the downstream weight should always increase the rate of that branch. If a subscription is split because of a partial cache hit, weights are first split equally and then adjusted for each sender separately by the receiver using the path price update messages. Because BNs have large caches for storing pending data for active subscriptions, it is not necessary to tightly couple upstream and downstream control loops which helps to stabilize the system compared to hop-by-hop congestion control.

When multiple subscriptions of a branching delivery tree are going to use the same cached data, the price of the storage space is added to the upstream segment cost of the delivery tree at the BN. The cost of storage space used for storing the subscription state is added to the downstream cost of each branch. The removal policy for inactive cached data is based on expected utility based on estimates of popularity and retransmission cost. This will also determine the minimum cost of storage space for active subscriptions. If the cost of the upstream segment of the delivery tree is greater than its combined weight at an intermediate BN, all downstream subscriptions of the tree are passed toward the data source. On the other hand, if the storage for the subscription state of a single branch costs more than the weight of that branch, the subscription is either moved toward the data source or removed completely if this happens in the data source. This algorithm does not necessarily minimize the total cost of the delivery tree but tries to push down the branching points in the delivery node as close to the subscribers as possible which should be a good approximation as the link costs are expected to be much higher than the cost of the storage space. Network nodes without cache can be seen as BNs with infinite price for storage which means that it is not necessary for all transit domains to deploy large caches or branch delivery trees if they still route subscription messages correctly and form forwarding paths through them.

On the sender side of the valley-free path it may be the case that compensation for requested data does not flow freely downwards. In this case, the separate weights used for link allocations to reach all selected rendezvous domains are determined by the source. It should be noted that because of this, the assumption about the elasticity of the flows does not hold any more. We will cover this and the congestion control in more detail in a future paper.

[6] by adjusting a congestion window size

7. FUTURE WORK

In this paper we have reported our results so far in solving the data-oriented inter-domain routing problem. The next steps in our work are to perform detailed simulations of the system described here, validate the stability and security of the congestion control mechanism, and integrate bursty traffic into it. Other interesting questions are how could we combine multiple forwarding methods and support multipath forwarding and how should the caches be located in the network.

8. REFERENCES

[1] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing. *SIGCOMM'93*, 23(4):85–95, 1993.

[2] S. Bhattacharyya. rfc3569: An Overview of Source-Specific Multicast (SSM). RFC 3569, IETF, July 2003.

[3] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker.

ROFL: Routing on Flat Labels. In *ACM SIGCOMM'06*, September 2006.

[4] B. Carpenter. rfc1958: Architectural Principles of the Internet. Technical report, IETF, June 1996.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, 2002.

[6] D. Clark, J. Wroclawski, K. Sollins, and R. Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. *IEEE/ACM Transactions on Networking*, 13(3):462–475, June 2005.

[7] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The Case for Separating Routing from Routers. In *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 5–12, New York, NY, USA, 2004. ACM.

[8] A. Filinski. Declarative Continuations and Categorical Duality. Master's thesis, University of Copenhagen, Aug. 1989.

[9] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, Dec. 2001.

[10] M. Gritter and D. Cheriton. An Architecture for Content Routing Support in the Internet. In *USENIX USITS'01*, 2001.

[11] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, and P. Nikander. LIPSIN: Line speed Publish/Subscribe Inter-Networking. In *SIGCOMM'09*, 2009.

[12] F. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[13] F. Kelly, A. Maulloo, and D. Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *The Journal of the Operational Research Society*, 49(3):237–252, Mar. 1998.

[14] F. Klemm, J.-Y. L. Boudec, and K. Aberer. Congestion Control for Distributed Hash Tables. In *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, pages 189–195, July 2006.

[15] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and Beyond) Network Architecture. In *SIGCOMM: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, volume 37, pages 181–192, 2007.

[16] G.-i. Kwon and J. Byers. ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections. In *IEEE INFOCOM 2004*, 2004.

[17] D. Mazières, M. Kaminsky, F. M. Kaashoek, and E. Witchel. Separating key management from file system security. *ACM SOSP'99*, Dec. 1999.

[18] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *ACM SIGCOMM '96*, pages 117–130, 1996.

[19] J. Rajahalme, M. Särelä, P. Nikander, and S. Tarkoma. Incentive-Compatible Caching and Peering in Data-Oriented Networks. In *ReArch'08*. ACM, 2008.

[20] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[21] M. Särelä, T. Rinta-aho, and S. Tarkoma. RTFM: Publish/Subscribe Internetworking Architecture. In *ICT-MobileSummit 2008 Conference Proceedings*, 2008.

[22] S. Shenker. Fundamental Design Issues for the Future Internet. *IEEE J. Sel. Areas Commun.*, 13:1176–1188, 1995.

[23] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhäuser, 2003.

[24] G. Urvoy-Keller and E. Biersack. A Congestion Control Model for Multicast Overlay Networks and its Performance. In *4th international workshop on network group communication*, Oct. 2002.

[25] X. Yang, D. Clark, and A. W. Berger. NIRA: A New Inter-Domain Routing Architecture. *IEEE/ACM Trans. Netw.*, 15(4):775–788, 2007.