

A Introduction

For much of the Internet’s existence, researchers have sought to make data transfer efficient and robust. Till the late 1990s, a client-server architecture dominated data transfer in the Internet. As the inherent scalability and fault-tolerance limitations of server-based architectures became apparent, several alternatives emerged. These include unstructured peer-to-peer file sharing [62, 23, 39], content distribution networks [2, 21], structured distributed hash tables (DHTs) [89, 80, 82, 72], publish-subscribe [83], end-system multicast [32], and more recently, file swarming systems such as Bittorrent [7]. In different ways, these architectures enable content to be *location independent*, i.e., they allow content to be retrieved from any convenient location where it is available, and provide a suitable discovery mechanism to determine one such location. The goal is to use location independence to make data transfer scalable and fault-tolerant by eliminating a single point of congestion or failure.

Unfortunately, this transition has not happened. There are fundamental shortcomings to data transfer on the Internet today. For instance, consider the problem of building a robust system to disseminate critical data to a large number of users. Existing peer-to-peer systems either suffer from “free-riding” or introduce a central point of failure like in Bittorrent.

Nevertheless, swarming systems like Bittorrent have some unique attractive properties compared to their predecessors. First, the data transfer protocol is designed to align the incentives for improving an individual peer’s performance with overall system performance. Bittorrent clients adopt a tit-for-tat strategy that incents a peer to contribute more resources to the system in order to obtain better performance for itself. The simplicity and robustness of tit-for-tat proved to be an effective deterrent to free riding, a problem that previous widely deployed peer-to-peer systems did not address satisfactorily. Second, Bittorrent uses multipoint-to-point connections to download a file in a manner that jointly optimizes an individual peer’s performance as well as global performance. To our knowledge, Bittorrent is the first instance of systematically using multipoint-to-point connections at such a large scale. The tremendous and sustained success of Bittorrent — recent studies by Cachellogic [8] estimate that it accounts for more than a third of Internet traffic today — suggest that Bittorrent may hold valuable lessons for networking researchers to design data transfer architectures for a future Internet.

Inspired by the success of Bittorrent, we pursue the following, rather exploratory, question in this proposal: **can swarms form the basis of a universal data transfer architecture for the Internet and if so, what is an appropriate architecture?** We use the term *swarm* to refer to a set of loosely interconnected nodes that act in a selfish and highly decentralized manner and are always in a state of adaptation. Natural systems with swarm-like properties are known to be extremely robust both for an individual and for the system as a whole.

In the rest of this proposal, *we outline fundamental research aimed the design, analysis, and implementation of an architecture that systematically uses swarms for scalable, incentive-compatible, and fault-tolerant data transfer.* Our project, which we refer to as *uswarm* (short for a universal swarm), has the following specific thrusts:

1. **Data Transfer and Control Plane:** We develop a data transfer and control plane for *uswarm* based on the idea of one large swarm for all data that is at least a few hundred bytes long. We develop mechanisms to enable in-network caching to exploit locality of data accesses. We propose to analyze the potential benefits with respect to availability and response time of *uswarm* over isolated swarms, as well as how *uswarm* interacts with and improves network traffic engineering.
2. **Incentive Strategies:** We design *uswarm* from scratch to be robust to selfish behavior using novel incentive strategies. We leverage our recent research that suggests that Bittorrent’s tit-for-tat strategy can be easily manipulated by peers resulting in significantly diminished swarm capacity. We propose to analyze the interaction of selfish peer behavior in *uswarm* with network resource allocation using game-theoretic techniques.
3. **Deployment and Case Studies:** We propose to develop two case study systems using *uswarm*. The first is a robust system to disseminate critical information preventing the so-called denial-of-information attack. The second will test *uswarm* in a delay-tolerant vehicular network testbed [16] at UMass Amherst.

We believe that swarms have the potential to address fundamental shortcomings of data transfer in the current Internet. First, today it is difficult to build a robust system to disseminate critical data, such as emergency information during disaster recovery or operating system security patches, to a large number of users. Existing peer-to-peer systems either suffer from “free-riding” or introduce a central point of failure like in Bittorrent. In particular, DHT-based designs are not robust to selfish peer behavior. Second, multi-homing, wireless environments, and intermittent connectivity are quite common today, but are poorly served by a point-to-point transport layer; instead a multipoint-to-point transport layer that can tolerate long delays or outages in connectivity is required. Third, the Internet poorly supports information sharing in online social networks, a rapidly rising phenomenon and multi-billion dollar industry [61, 103]. Today, such sharing is done by aggregating content at a central location, a model that appear inherently difficult to scale in the long term. Finally, the organic growth of peer-to-peer data transfer systems today causes considerable headache to network administrators who go to great lengths to rate-limit or otherwise control these applications. On the other hand, these systems largely serve legitimate needs of users. We believe that the lessons learnt by studying these systems could lead to a more principled understanding of how to integrate them cleanly into the Internet’s data transfer architecture.

Our effort aligns well with FIND’s agenda in that we pursue a fundamentally different view of how a network should perform data transfer starting with a clean slate. Technically, our architecture differs from the current Internet in the following respects. First, multipoint-to-point connections are the norm, not a special case. Second, the architecture is designed to carefully account for incentives at every step. Third, the architecture is designed for fluid replication of data at any location where it is accessed.

An important aspect of our research is a rigorous combination of theory and real-world deployment. The PIs bring an eclectic set of skills that is well-suited to conduct the proposed research.

The rest of this proposal is organized as follows. Section B gives an overview of the uswarm architecture. Section C describes in detail the proposed research plan to develop the uswarm control and data plane, to quantify its benefits over data transfer today, and explore new traffic engineering knobs that it enables. Section D proposes novel incentive strategies that can form the basis of a universal swarming data transfer architecture, and evaluates its interaction with network resource allocation. Section E describes our case studies. The subsequent sections describe related work, a plan of work, synergy with other FIND efforts, broader impact and education plan, and results from prior support in that order.

B Overview of the uswarm Architecture

In this section, we describe the high-level architecture of uswarm. A helpful aid is to think of uswarm is one huge swarm for all data transfer as opposed to one swarm per file in Bittorrent like systems today.

In uswarm, the typical mode of transfer is a multipoint-to-point connection between peers. Peers in uswarm can be end-hosts or network intermediaries. End-hosts include typical user machines, servers, or content distribution intermediaries. Data in uswarm is location-independent and self-verifying, i.e, any peer can store and serve data and the proof of integrity of the data is in the data itself. The basic architecture is designed to transfer bulk data, i.e., named data that is several packets long and is re-used over time; we discuss how uswarm benefits dynamic data at the end of the section.

B.1 Naming and resolution

Figure 1 illustrates the key components of uswarm. An intent resolution service (IRS) is responsible for translating *intent* to *metadata*: the intent specifies what a peer is looking for and could be expressed as a set of keywords, a URL, an RSS feed, or other applications-specific means. The IRS could be a modern search engine, a web server that serves metadata similar to hyperlinks, or a peer-to-peer information retrieval plane similar to keyword searches supported by popular file sharing applications.

The metadata uniquely specifies the data. A metadata resolution service (MRS) dynamically returns a set of peer addresses that can potentially serve the data. This is the equivalent of a tracker in BitTorrent although, in uswarm, it is a distributed service. Any peer, including a network intermediary, that intercepts the request can resolve the metadata to peer addresses. Finally, the requesting peer makes a multipoint-to-point connection to a subset of the returned peer addresses to retrieve the data.

uswarm uses the hash of the content to identify content in a self-verifying manner [92, 57]. More concretely, a name consists of a two-tuple $[publisher, hash]$. The publisher field is optional and uniquely identifies a publisher, e.g., `www.umass.edu`, and the assignment of these names is managed by a global authority like ICANN [34]. The second field is mandatory and is a message digest [86, 58] of the content. If the publisher field is present, the digest is further encrypted by the publisher’s public key supported by a standard public key infrastructure.

The metadata consists of the self-verifying name followed by a list of self-verifying block identifiers. Each block identifier is a three-tuple $[offset, length, hash]$ where the offset and length identify the position of this block in the content and the hash is a message digest of the corresponding bytes. A similar naming scheme is used by DOT [92] and BitTorrent.

B.2 Central vs. distributed tracking

Peers in uswarm translate metadata to peer addresses using one of three options: (i) a logically centralized tracking service similar to DNS today, (ii) in-network support for tracking where a gateway router intercepts and processes a resolution request, and (iii) peer-to-peer tracking where peers help each other resolve metadata.

Why do publisher-supported trackers like in BitTorrent not suffice? First, central trackers create a single point of failure and congestion; content is not truly location-independent if it is tied to the publisher’s location. Second, publisher-supported trackers are well suited for pull-based data transfer when a user’s intent is known. But they are less useful when a user would like content of her interest, that she didn’t know even existed, pushed to her. We show how distributed trackers that leverage properties of social networks are better suited for the latter. Third, in wireless environments with intermittent connectivity or delay-tolerant environments, an unreachable remote tracker may cause data to be unavailable even if it exists in the user’s vicinity.

uswarm makes it convenient to exploit locality by interposing in-network caches and trackers. An organization or ISP often has a monetary incentive to serve the requested data from within its domain. Several studies have pointed to significant redundancy in network traffic, but there is little done to exploit this redundancy today. Although web proxy caching has seen over a decade of research, it is not widely used today for two reasons. First, Web traffic constitutes a small fraction of overall network traffic [8, 24], so the benefit of caching is not significant, particularly, with the attendant issues of ensuring consistency, handling dynamic content, or just administrative cost. More importantly, a cache needs to be developed and deployed for each new application (e.g., BitTorrent) that becomes wildly popular. Thus, there is a fundamental need to systematically support caching at the network layer in an application-agnostic manner, which uswarm enables.

B.3 Roles and application usage

Peers in the uswarm game have roles depending on their economic relationship with other peers and their network service provider. BitTorrent peers are referred to as seeds or leeches depending on whether they stick around after a download is complete. However, peers in uswarm are always ready to upload cached blocks of any file provided doing so is incentive-compatible. Publisher sites, CDN nodes, and in-network caches will always choose to upload

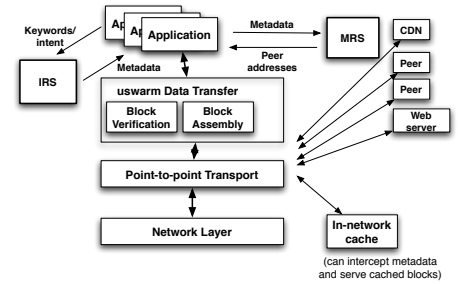


Figure 1: The uswarm data transfer architecture.

requested blocks given today's business models. However, in general, it costs peers to upload blocks (monetary bandwidth cost or interference to other applications), so they will behave selfishly. Peers may also belong to a social network, where they are partial to closer friends than more distant ones. Consequently, it is important to admit a general cost and payoff model in uswarm (see Section D).

Although the design of uswarm has been primarily influenced by bulk data transfer, the architecture is potentially useful for more sophisticated network services. Note that any re-usable content that is longer than a *few hundred bytes* can use uswarm as the overhead of metadata is minimal. Examples other than Web, CDNs and file transfer that can benefit from uswarm include:

- *Live streaming*: Several commercial and research efforts [44, 106] are aimed at developing peer-to-peer live streaming systems that are robust to selfish peer behavior, for which uswarm forms a natural basis.
- *Semi-autonomous peer systems*: Many Video-on-Demand (VoD) systems today are considering a push-based architecture where content is proactively pushed to set-top boxes that are under provider control, e.g., set-top boxes in a single DSLAM in a DSL network, or high-definition gaming servers and client software. The actual available bandwidth depends on other application traffic being independently generated by peers.
- *Human-centric applications*: The control plane of uswarm is designed to adapt to a topology based on content of interest to peers facilitating push-based applications. uswarm also naturally tolerates long delays or interrupted connectivity making it suitable for delay-tolerant networking applications (refer Section E).

C Data Transfer and Control Plane

In this section, we describe research necessary to design and implement a robust data transfer and control plane for uswarm. In particular, we investigate fundamental benefits of uswarm over isolated swarms with respect to performance, availability, locality and in-network caching, as well as its interaction with traffic engineering and management goals of network service providers.

The key benefits of one large tightly connected swarm over isolated swarms are as follows.

1. *Post-popularity*: Bittorrent is robust to flash crowds, but may have poor performance after the popularity wave has subsided. The reason is that there are few seeds in the system and the total upload capacity of seeds bottlenecks download performance. Although a publisher may be reasonably expected to support enough seed bandwidth for post-flash-crowd periods, this may lead to unfavorable interactions between swarms for different content supported by the same server or data center, e.g., if a publisher's website experiences a slashdot effect for file A, then a peer seeking a different and unpopular file B could experience a poor download rate even if B is sufficiently replicated in the network.
2. *Block availability*: In Bittorrent, two peers may be well-matched with each other with respect to upload capacities, but not have blocks of interest to each other. Allowing peers to exchange blocks across different content can improve block availability [25] and consequently download performance, even for small files consisting of a single block.
3. *Robust tracking*: Although Bittorrent's tracker-based architecture works reasonably well, it is inherently unscalable and cracks are already beginning to show. Our recent measurement study [63] indicates that some torrents have *fifty or more trackers for the same file* leading to disconnected or poorly connected swarms hurting performance. Support for distributed tracking coupled with tightly interconnected swarms can be more robust and improve performance due to improved opportunities for TFT exchanges, both for a single file and across different files.

C.1 Proposed approach

Our approach is to systematically analyze the claimed benefits of uswarm over isolated swarms and design mechanisms necessary to achieve them in practice.

C.1.1 Data plane: uswarm vs. isolated swarms

We consider a simple example consisting of two isolated swarms S_1 and S_2 as shown in Figure 2. Assume that both swarms have the same number n of peers that wish to download files f and g respectively. Let c denote the capacity of each peer as well as the available seed bandwidth for each file, say, supported by the publisher. Finally, assume that peers in S_1 already possess g and peers in S_2 already possess f . We compare the overall download performance when the two swarms operate independently versus when the two swarms assist each other.

Conjecture 1: The overall download capacity with interconnected swarms can be a $O(\log n)$ factor greater than isolated swarms.

The intuition behind the conjecture is as follows. Consider the isolated swarm S_1 ignoring incentive issues for the moment. It can be shown, similar to Yang et al. [99], that the best strategy is for the seed to use its entire upload capacity first to replicate a block of f to one peer, at which point, the total upload capacity increases to $2c$. Similarly, the capacity doubles in subsequent periods until, in $\log n$ rounds, the total upload bandwidth utilization equals total capacity $c.n$. On the other hand, if the two swarms cooperate, then each of them directly starts with a utilization of $c.n$, potentially, yielding a $O(\log n)$ improvement over the isolated swarm. Furthermore, we ignored block availability, i.e., whether two peers have blocks of interest to each other, and incentive issues above, which are likely to further benefit the interconnected swarm compared to the isolated swarms.

Similarly, the interconnected swarm can be shown to be more reliable compared to isolated swarms. Assuming a uniform mean time to failure for each node in the system, there is a higher probability of a missing block in the isolated case as the failure of the seed any time during the first $O(\log n)$ rounds can result in an unavailable block. On the other hand, all n peers holding a block of either file have to perish for a block to become unavailable. This is of course an over-estimate of the improvement in reliability as we did not precisely model the tracking process itself, which impacts reliability as a block may exist in the system but not be found by a peer. We turn our attention to the tracking process next. We refer the reader to our preliminary work on a fluid model of these swarms [20].

C.1.2 Control plane: distributed tracking

The tracker enables peers to find other peers holding blocks of their interest which, in Bittorrent, is implemented as a central server. The centralized tracker thus introduces a single point of failure: if the tracker fails or is unreachable, the system becomes unavailable to new peers, so they can neither obtain content nor contribute resources to the system. Recent measurement studies [74, 63] confirm that low tracker availability is a significant problem for Bittorrent users today.

To improve tracker availability, two recent trends have emerged. First, is the support for replicated trackers for each file. Second, is the integration of DHTs with BitTorrent clients that store information across the entire community of BitTorrent users; the DHT trend is more recent and began to appear a little over a year ago. To understand the practical impact of these mechanisms on tracker availability, we conducted an extensive measurement study spanning more than 26K torrents, 1700 trackers, and 25K DHT nodes over a period of several months. Our major findings include:

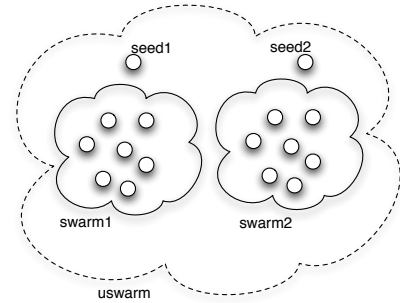


Figure 2: Two swarm example.

- Replicated trackers improve availability, but the improvement largely comes from a single highly available tracker. In particular, correlated tracker failures significantly hurt availability.
- Replicated trackers result in poorly connected swarms even though the constituent peers seek the same content.
- DHT-based trackers improve availability, but introduce a high response latency.

The last observation is illustrated in Figure 3 that compares the time required to obtain a peer with DHT-based and replicated trackers. Furthermore, a DHT-based tracker may not be incentive-aligned as it assigns tracking responsibility to a peer that actually has an disincentive to assist other peers, a topic we address in detail in the next section. Thus, a key challenge in uswarm is to enable support for robust distributed tracking.

We propose to achieve robust distributed tracking in uswarm through a combination of (i) massively replicated tracking, (ii) peer-to-peer gossip and (iii) in-network tracking (refer Figure 1).

The first option, massively replicated tracking, is similar in spirit to DNS — a hierarchically organized set of uswarm trackers assist peers by dynamically returning relevant peer addresses. Such an architecture would inherit the scalability and fault-tolerance properties of DNS, but is burdened with a more complex service. Unlike DNS that only performs simple lookups that change rather slowly, the uswarm tracking service must estimate which peers are likely to have blocks of a particular file. This problem can be alleviated in practice by: (i) tracking files clustered by publisher or semantics as opposed to tracking each file individually, (ii) assistance from domain-level trackers supported by publishers, and (iii) failover support provided from peer-to-peer and in-network tracking.

The second, peer-to-peer gossip works in two ways: (a) controlled flooding, and (b) active gossip. Controlled flooding is similar to Kazaa [39] or Gnutella [23] in which a peer asks its neighbors recursively for information. We chose controlled flooding over DHTs for a couple of reasons. The first is incentive compatibility — it is easier to design unstructured peer-to-peer networks that carefully account for incentives and peer heterogeneity, a point also made by Chawathe et al. [11]. The second is exploiting locality; designing practical locality-aware DHTs [28, 1, 46] is more complex. Finally, unstructured peer networks are well-suited to content sharing based on social networks as we describe in our case studies section. Active gossip works by re-using neighboring peers in the data plane to also learn about new peers. Just like peers in Bittorrent trade data blocks, they could also trade relevant peer addresses; we defer the precise incentive scheme to Section D.

The third, in-network tracking, couples a uswarm tracker beside in-network caches. In addition to caching data blocks, these routers also intercept and re-direct content requests to other peers in the same organization when a block is not found in its cache, or the router is under heavy load.

C.1.3 Implications for Traffic Engineering and Virtualized Architectures

The research described so far, except for in-network caching and tracking, largely resides above at and above the transport layer. While transport and higher layers are certainly elements of a “network architecture”, we believe that it is important to understand the interaction of a data transfer architecture on traffic engineering and management objectives of network service providers.

Today’s Internet constrains the routing choice available to both end users and ISPs. While the former is well understood [98, 81, 76], we note that lack of location independence, i.e., the freedom to choose the destination, constrains ISPs too. An AS can choose interdomain routes and exit policies, but has little freedom to alter its traffic matrix as, in most cases, a flow is uniquely mapped to an ingress-egress pair. On the other hand, network-layer

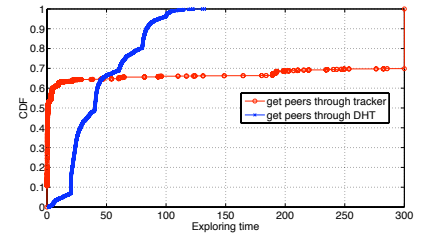


Figure 3: DHT vs. replicated trackers latency in BT.

tracking gives an ASes much more control over its customers' traffic. In particular, a tracker router can return a set of peers that results in the most load-balanced traffic assignment.

Flexible traffic engineering : We propose to investigate the power of this traffic engineering knob as follows. Consider how traffic engineering today is coupled with link-state routing. An AS takes as input a traffic matrix that is assumed to be unchanging over fine time scales. A set of OSPF link weights is computed such that, if the flows in the traffic matrix were routed along the shortest paths, then the overall cost, defined as say $\sum_{l \in L} \frac{C_l}{C_l - \rho_l}$ is minimized. Here the sum is over all links l and C_l and ρ_l denote the capacity and actual load on link l . Note that the cost thus achieved may not be the optimal, however traffic engineering is done this way largely because of the widely deployed and well-understood shortest path routing machinery. Furthermore, the actual traffic matrix itself may change over fine time scales, but link weights are recomputed only at coarse time scales. This mismatch between an AS's traffic engineering objectives and the knobs it has available is known to cause much frustration to network administrators [104, 3].

Now, suppose that an ingress tracker router had the freedom to dynamically return a set of peers that results in a traffic matrix that results in the least cost (either with the current set of link weights or with a re-computed set of weights). This gives an AS a powerful knob to achieve its traffic engineering objectives. The flexibility provided by this knob depends on the location of in-network caches as well as locality of content access by peers in an AS, which we propose to analyze via simulations over real backbone topologies and traffic matrices. We will leverage our previous experience in modeling such interaction — “Can an overlay compensate for a careless underlay” [31]. The relevant question to ask here is — “Can uswarm enable traffic engineering knobs that give both end-users and networks better routing choices”?

C.2 Research questions and plan

The projected performance and reliability benefits and the tracking mechanisms described above leave open several questions that we propose to address as follows.

1. The two-swarm example above does not tell us the performance or reliability benefits we can expect in practice. First, we must accommodate swarms of varying sizes and heterogeneous bandwidth capacities. Second, peers may arrive in the system staggered over time. An arrival process of particular interest is one where hordes of peers arrive as in a flash crowd over a short duration followed by a long tail of casually arriving peers. Finally, the amount of overlap in content of interest to different peers impacts the performance of the system (refer [20] for preliminary work). We will adopt a measurement-driven modeling approach to understand their impact uswarm's performance and reliability. We will begin by participating in popular Bittorrent swarms to measure peer arrivals and capacities, monitor popular torrent sites for popularity distribution of content, and estimate overlap in content interest. Subsequently, given that uswarm is intended as a universal data transfer architecture, we will extend the analytical and experimental evaluation of uswarm's benefits to more diverse content such as HTTP traffic, software updates, TCP multimedia traffic, RSS feeds etc. using a workload gathered at the UMass Amherst gateway leveraging existing infrastructure [33] to collect packet header information.
2. The three-pronged approach for distributed tracking described above may seem inelegant to an architectural purist. However, we believe that all three mechanisms — massively replicated global tracking, peer-to-peer gossip, and network layer tracking — are complementary and can provide levels of performance or reliability difficult to achieve otherwise. Given that our end goal is to build a practical system based on uswarm, our research will precisely quantify the relative benefits of the three tracking mechanisms and integrate them into a holistic policy architecture that enables seamless invocation of or failover to the best mechanism dynamically.

3. We implicitly assumed in the description of the approach that download rate is the performance metric of interest. However, for streaming applications based on buffering and playback, it is important to limit the latency of retrieving each block. More generally, a data transfer architecture must be capable of supporting sophisticated scheduling requests issued by users, e.g., “retrieve movie A within 2 hours and file B as quickly as possible”, or “retrieve as many relevant documents as possible to these keywords within 10 minutes” for running a local data mining application over the results. The latter may require careful selection of peer, CDN, or server nodes based on network characteristics. We will leverage our prior experience building *iPlane*, an Internet-wide information plane, to enable more sophisticated data transfer objectives integrating *iPlane* with uswarm’s distributed tracking plane.
4. In-network caching of blocks naturally raises the question of placement and replacement strategies. Although our architecture is based upon the premise of inexpensive storage, it is by no means sufficient to support an Internet-wide data transfer architecture. We will leverage our prior work on online hierarchical cooperative caching [45], which developed a lower bound and a matching constant-competitive algorithm, as well as bandwidth-constrained placement strategies [95, 96] for prefetched content.

D Incentive Strategies

In this section, we address the following research questions: (i) what incentive mechanisms are necessary to make uswarm robust?, (ii) how can we model uswarm using game theory? and (iii) how do incentives interact with congestion control and routing?.

D.1 Proposed approach

D.1.1 Incentives in isolated swarms

Free-riding is a fundamental problem in peer-to-peer systems. Initial attempts to thwart free-riding in peer-to-peer systems proved unsatisfactory, e.g., “incentive priorities in Kazaa [39] could be spoofed, currency in MojoNation was cumbersome, and the AudioGalaxy Satellite model of “always-on” clients has not been taken up. In contrast, Bittorrent’s tit-for-tat incentive strategy has been found, in practice, to be successful at inducing contribution from rational peers. Combined with the bilateral nature of tit-for-tat, which allows for enforcement without a centralized trusted infrastructure, a popular belief has emerged that “incentives build robustness in Bittorrent” [7, 75, 6, 43].

Our recent research questions this widely held belief. Based on a simple model driven by real-world measurements, we discovered two artifacts. First, there is significant altruism in Bittorrent, leading to a minority of high capacity peers contributing to the bulk of aggregate swarm resources. Second, and more importantly, we discovered that this altruism is not a consequence of tit-for-tat. In fact, a selfish peer can significantly reduce its upload contribution and yet obtain a higher download rate. If many peers behaved selfishly, then honest low capacity peers can experience degraded performance due to the absence of altruism, suggesting that Bittorrent is not robust to selfish peer behavior.

On the positive side, we discovered that our selfish algorithm benefits every peer irrespective of how many other peers are using it. Our preliminary algorithm, called Bittyrant, is illustrated in Figure 4. Bittyrant incents peers to adopt it over Bittorrent, thereby creating a natural evolution path towards its widespread deployment.

We learned two valuable lessons through our measurement and modeling effort. First, peers observe a random selection of other peers in large swarms despite the optimistic unchoke process at work. This is in contrast to models that characterize steady-state behavior [75]. Second, previous models do not account for two forms of strategic peer behavior: (i) the number of active connections maintained by a peer, and (ii) how a peer splits its upload capacity across those connections. Our preliminary work shows that strategizing with these degrees of freedom can significantly impact the performance observed by a majority of peers.

The preliminary work leaves many research questions open such as : (i) estimating peer upload and download capacities, (ii) analyzing if the proposed mechanism is strategyproof, (iii) robustness to byzantine faulty peers.

D.1.2 Incentives in uswarm

The potential benefits of uswarm over isolated swarms described in Section C did not account for selfish behavior. Here, we outline incentive strategies that retain the performance and reliability benefits of uswarm while being robust to selfish behavior.

Data plane We propose to extend the Bittyrant algorithm to uswarm using the same basic principle, i.e., select peers so as to maximize return (download rate received) on investment (upload rate). However, there are two key differences. First, each peer may be uploading blocks from several different files in order to download a particular file. Second, a peer may wish to simultaneously download several files and associate different utilities with different files. This can be achieved by sorting all available peers according to their d_p/u_p ratios weighted by utility. More concretely, if a peer is interested in m files with normalized utilities of r_1, \dots, r_m , then d_p is computed as $\sum_{i \in [1, m]} r_i \cdot d_{pi}$ where d_{pi} is the estimated or observed download rate of file i from peer p .

Control plane A key design requirement in uswarm is an efficient and incentive compatible control plane. While the global and in-network trackers can be implemented as an infrastructure or commodity service, the peer-to-peer scheme demands a decentralized and robust incentive mechanism. Recall that the objective of the control plane is to translate metadata into a set of peer addresses. With controlled flooding like in Kazaa or DHT-based distributed tracking, a selfish peer may simply drop a request and, indeed, in some cases preventing other peers from accessing a resource may be to its advantage.

To address this problem, we employ two mechanisms for the control plane: (i) tit-for-tat, and (ii) dynamic topology adaptation based on exploiting properties of social networks. We assume that peers are organized into a control plane topology and use controlled flooding to resolve metadata as in Gnutella or Kazaa. The tit-for-tat strategy works by keeping track of the number of metadata requests a neighbor helps resolve. Peers choose to help neighbors that have been most useful in the past. The dynamic topology adaptation mechanism works in conjunction with tit-for-tat by choosing more useful peers as neighbors over less useful ones. Additionally, peers prefer other peers with similar content interest for neighbors, automatically leading to a topology that resembles a semantic social network, in which adjacent peers have highly overlapping content interest. The dynamic topology adaptation scheme is inspired by Gia [11] that is designed primarily for scalability; in comparison, incentives and overlap in content interest fundamentally adapt the control plane topology in uswarm.

A key research goal is to determine how the behavior of uswarm is controlled by three interacting processes. The first is peer selection in the data plane that operates at fine time scales on the order of tens of seconds. The

For each peer p , maintain estimates of expected download performance d_p and upload required for reciprocation u_p .

Initialize u_p and d_p assuming the bandwidth distribution in Figure 2.

d_p is initially the expected equal split capacity of p .

u_p is initially the rate just above the step in the reciprocation probability.

Each round, rank order peers by the ratio d_p/u_p and unchoke those of top rank until the upload capacity is reached.

$$\underbrace{\frac{d_0}{u_0}, \frac{d_1}{u_1}, \frac{d_2}{u_2}, \frac{d_3}{u_3}, \frac{d_4}{u_4}, \dots}_{\text{choose } k \mid \sum_{i=0}^k u_i \leq \text{cap}}$$

At the end of each round for each unchoked peer:

If peer p does not unchoke us: $u_p \leftarrow (1 + \delta)u_p$

If peer p unchokes us: $d_p \leftarrow$ observed rate.

If peer p has unchoked us for the last r rounds: $u_p \leftarrow (1 - \gamma)u_p$

Figure 4: Bittyrant unchoke algorithm

second is peer selection in the control plane that operates over slower time scales on the order of hours or days. The third is the movement of peers across interest clusters based on their content access pattern. We propose to model this interaction in order to design policies that lead to efficient and reliable performance in steady state.

D.1.3 Connection games in uswarm

Our focus so far has been on the impact of selfish behavior on uswarm’s data and control planes assuming no cost to opening a connection. In practice, opening a peer uses memory and computing resources. The interaction of a large number of TCP flows itself causes performance degradation in practice [59, 77]. A Future Internet may use improved, network-assisted congestion control, however, that will incur a cost for the network as a whole. A Future Internet may also reflect this cost back to the user by charging on a per-connection basis. Thus, it is important to take into account the cost of opening a connection in uswarm, an architecture fundamentally designed to use several point-to-point connections for a single file.

We begin by modeling a cost that is proportional to the number of open connections, i.e., a cost βn for n open connections and fixed β . Figure 5 illustrates a “swarm formation game”. Let the bigger nodes represent peers with capacity say $C_1 = 3$ and the smaller ones with a lower capacity $C_2 = 2$, and each peer be restricted to at most 3 connections. Assume each pair of connected peers reciprocate, and each peer uploads by equally splitting their capacity across its neighbors. Then, the four high capacity peers other than S each receive a download rate of $3(= 3C_1/3)$, while S , the peer connecting the two groups, receives a rate of only $8/3(= 2C_1/3 + C_2/3)$. Similarly, the four low capacity peers to the right receive a rate of 2, while the interconnecting one receives $7/3$. This is a stable configuration where no peer benefits by switching peers. Now suppose peers could choose the number of connections, e.g., peer S increases its number to 5. If all other peers remained at 3 connections, S improves its performance from $8/3$ to $10/3 = (5C_2/3)$. However, if each extra connection caused high interference resulting in a penalty of $1/3$ in the rate, then benefit of increasing connections for S is offset by the increased cost.

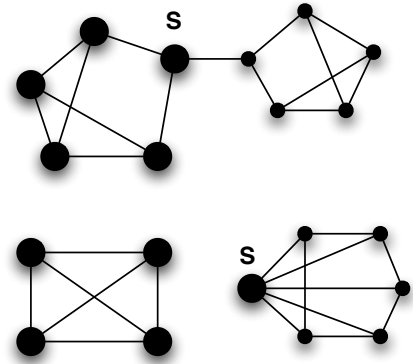


Figure 5: Tit-for-tat connection game

Next, suppose that peers followed a tit-for-tat strategy where a “link” is considered stable only when both peers choose to reciprocate. We can model this swarm formation game and analyze equilibrium behavior and the loss of efficiency due to selfish behavior, often termed as the “price of anarchy”. Our preliminary modeling effort [30] suggests that a *pairwise Nash equilibrium* exists for this game, i.e., a peer p can not benefit by unilaterally deviating from the equilibrium peer selection strategy and, furthermore, the benefit to p by adding a new peer q is offset by the loss to q . Unfortunately, the loss of efficiency in such an equilibrium is unbounded. This suggests that, in the presence of connection setup costs, we need to devise an enforceable incentive mechanism that restricts the set of available strategies to peers.

The above model abstracted away the effect of the underlying routing topology and congestion control. Ignoring selfish strategies like tit-for-tat, we have been able to model connection games where a peer is allowed to increase or decrease its number of open connections over a fixed set of available paths taking the underlying topology as well as congestion control into account. Assuming linear connection costs and TCP-like congestion control protocols, our model bodes pessimistic results: (i) a Nash equilibrium can be shown to exist for only simple network topologies, and (ii) the loss of efficiency at equilibrium can be arbitrarily large.

A key research agenda is to understand the impact of strategic peer selection on network resource allocation taking the underlying topology and connection costs into account; the models above analyze strategic peer selection or network resource allocation in isolation, but not their interaction. We will use these models to drive the design

of practical incentive mechanisms and peer selection strategies that are both efficient and stable.

D.2 Research questions and plan

In addition to the research questions outlined above, we will address several other practical research issues.

1. *Pricing*: The incentive strategies outlined above are designed for a market based on bandwidth as currency. The goal was to optimize download rate given a limited upload bandwidth. In practice, upload bandwidth may not be “free” or even linearly priced. ISPs often bill customers based a variety of bandwidth pricing schemes based on 95th percentile usage rate, peak rate, average rate or a minimum committed rate, and so on. A data transfer architecture must be neutral to the exact pricing scheme, so we will investigate support for different forms of bandwidth budgets or utilities based on both download rate benefit and upload cost.
2. *Indirect trading*: A tit-for-tat strategy may not be very useful for a mobile user with low upload capacity or power constraints. However, such a node may appoint, say, their desktop machine as a “home agent” to participate in uswarm on its behalf. The home agent may be used to either detour traffic (as in direct routing in Mobile IP), or be explicitly appointed as a delegate to which the mobile may direct peers. The mobile and the home agent can use standard symmetric key cryptographic techniques to authenticate the delegation. More generally, peer-to-peer detour routing itself may be implemented as a service on top of uswarm even for wireline networks.
3. *Workload and network conditions*: First, the models above do not account for the case when two peers are willing to upload blocks but do not have blocks of interest to the other. What is the impact of block availability on uswarm’s performance? Second, we assumed that a single point-to-point connection may suffice to utilize all or fair-share bandwidth along a path. In practice, TCP flows utilize only a small fraction of bandwidth on high capacity networks both due to TCP’s inefficiencies as well as traffic shaping on part of ISPs. Third, we assumed a rate-based strategy above, but one could consider a block-based tit-for-tat strategy where peers control the number of blocks uploaded to a peer based on the number of blocks downloaded from the peer; several other strategies are possible and their interaction needs more investigation. Finally, more global knowledge about other peers’ capacities helps a selfish peer make better strategic decisions. Although an information plane like *iPlane* may be employed, it is necessary to investigate the effect of noisy information on the efficiency or vulnerability to strategic peer behavior of uswarm.

E Deployment and Case studies

In this section, we describe research necessary to make uswarm practical. To this end, we propose to design and implement several case study systems on top of uswarm.

E.1 A DoI-resistant information dissemination service

Securing the Internet is widely considered as a fundamental architectural problem requiring a clean slate re-design, e.g., practically every network protocol is vulnerable to denial-of-service (DoS) attacks. However, a graver concern is that, as a result, the Internet does not even support a robust dissemination service for static or slow changing data, a problem we call as the denial-of-information (DoI) attack. For example, an attacker can simply DoS a web server or Bittorrent tracker to render the information unreachable. A DoI-resistant dissemination service would be valuable for disaster recovery and, more generally, for lookup services such as DNS or other bootstrapping services. An ideal DoI-resistant architecture would ensure the following property: if a user seeks a file at least one copy of which exists in the network, then she should be able to retrieve it unless the underlying network is physically partitioned by faulty or malicious nodes.

We propose to build a DoI-resistant information service on top of uswarm. The basic idea is simple: to use massive replication to ensure that a peer can reach at least one replica. To accomplish this, we need knowledge of the underlying routing topology, for which we will leverage our previous experience with *iPlane*. The knowledge of the underlying routing topology can be used by peers to select neighbors in uswarm’s control plane such that it is extremely difficult for an attacker to disconnect any peer from content it seeks. Finally, the content itself must be sufficiently replicated to begin with taking into account the network topology, so that DoS attacks on peers holding the replicas do not make the information unavailable.

Opportunistic measurement: On a complementary trail, we will also instrument a small number of Bittyrant clients to passively monitor transfer performance and contribute measurements to *iPlane*. Unlike other measurement projects that rely on altruism or sympathy for science [85], uswarm will make it incentive-compatible for peers to contribute measurements. As noted in Section D, peers with more information about the capacity of other peers (and network paths to them) can make better strategic choices. In return for such “shadow tracking” [51] services provided by *iPlane*, these peers will be expected to contribute to the information plane itself to develop a more complete topological map of the Internet. A research challenge here is to carefully choose the set of measurement responsibility assigned to peers based on their location to ensure coverage and reliability of reported information.

E.2 Delay-tolerant uswarm and new services

A key benefit of uswarm is that, by design, it is tolerant to delays or interrupted point-to-point connections. Thus, peers in uswarm can exchange blocks of interest with other peers within radio range. uswarm can also operate in delay-tolerant environments such as vehicular networks [16] or people networks [60]. We believe that uswarm can form the basis of a data transfer architecture whose performance gracefully degrades from well-connected wireline networks to mobile ad-hoc or delay-tolerant networks.

We will leverage our experience in developing and deploying a routing protocol, RAPID (Resource Allocation Protocol for Intentional DTN routing), that enables sophisticated “knobs” to intentionally optimize performance metrics of interest such as worst-case or average delivery delay, number of transfer that miss their deadlines etc. In contrast, several DTN routing protocols that incorporate a large number of design decisions to improve the likelihood of just finding a path, so they only have an incidental effect on specific performance metrics. A preliminary version of RAPID is already deployed on about 20 buses belonging to the DieselNet vehicular network testbed at UMass Amherst.

DTNs and MANETs also give us an opportunity to explore new incentive mechanisms for these environments. For example, appointing a delegate for indirect tit-for-tat, as described in Section D, and other access-point-assisted incentive mechanisms will be useful. Finally, we will also explore push-based applications where adjacent peers in uswarm’s control plane can *advertise* content of potential interest to their neighbor, and a software agent can choose to accept the trade subject to user-specified interests. Content pushing applications are particularly well suited to a control plane topology based on social networks of overlapping content interest and are applicable to both well-connected as well as wireless and delay-tolerant environments.

F Related Work

The architecture of uswarm borrows heavily from previous work in peer-to-peer systems, research on Bittorrent like systems, caching, and evolutionary game theory. Our primary contribution is to adapt these ideas to enable a holistic data transfer architecture based on swarms, which to our knowledge has not been done before.

Structured peer-to-peer systems based on DHTs [89, 80, 82, 107, 56, 53, 54, 28, 4, 46] have seen a large body of work in recent times. DHTs can massively scalable, fault-tolerant, and low-latency distributed services. However, DHTs primarily work by randomly hashing an identifier (a file, task, or information) to a node in the system. However, such an assignment may not be incentive-compatible to nodes. Furthermore, although DHTs are well

suited to distribute load across different files, they fall back on traditional replication techniques to handle flash crowds for a single file. More recent DHT based systems such as Beehive [79], CoDoNS [78] exploit caching and workload profiles to address this problem, but are still based on an underlying DHT infrastructure. In comparison, our goal is not to support generic distributed services; rather uswarm is a robust data transfer architecture that is designed from scratch with incentives in mind.

The benefits of using multipoint-to-point connections [91, 13, 87, 70, 92, 38], and mesh-based content or peer-to-peer content distribution [42, 9, 22, 87, 21, 88] have been well noted, however these proposals are either for managed infrastructure or cooperative peer-to-peer environments. Some of the proposed techniques for dynamic topology adaptation are similar to Gia [11]; however, their primary focus is to make file sharing scalable, but not necessarily robust to selfish peer behavior.

Incentives in peer-to-peer systems have seen a large body of work in recent times (see [66, 67, 68] for a partial list). These proposals use one or more of the following techniques: (i) virtual currency, (ii) reputation-based schemes, (iii) secure identities, and (iv) bilateral reciprocation-based strategies. In comparison, uswarm relies only on reciprocation-based strategies as they are simple, low-overhead, and completely decentralized. Feldman et al. [17] model incentive techniques based partly on reciprocation and suggest it they can drive a system of strategic users to nearly optimal levels of cooperation; peer selection in uswarm’s control plane uses techniques similar to “discriminating server selection” and “adaptive stranger policies” in their work.

We freely admit to borrowing from these and other existing work on incentives in peer-to-peer systems. What sets us apart from prior work is our goals and research methodology. First, we seek to augment data transfer based on servers, CDNs, and other infrastructure, not replace them. Second, we seek to develop an incentive mechanism that limits free-riding, not eliminate it, while maintaining reasonable performance. Third, in a data transfer architecture, we seek to exploit locality of accesses, overlapping content interest, and inherent altruistic or sharing tendencies of people when appropriate. To this end, our research seeks to integrate social networks, an important growing phenomenon, systematically into a data transfer architecture.

Our research methodology is based on a rigorous combination of practice and theory. Prototype systems and measurements “in the wild” will drive both our design and analysis. On the modeling front, our focus is on modeling transient behavior compared to existing simplistic analyses of swarms that assume steady state behavior that is rarely reached. It is noteworthy that uswarm by design is always expected to be in a state of flux.

G Plan of Work

The proposed work will involve three graduate students who will work in coordination. One student will focus on modeling and analysis of uswarm, while the the other two will focus on the design and implementation of the core uswarm architecture, develop case study systems, develop a measurement infrastructure for gathering data from real swarms, and experimentally analyze the interaction of uswarm with existing network-layer mechanisms as well as new ones being developed by other FIND researchers. The project will span three years with the following schedule and milestones.

- Year 1: Develop models to analyze transient single-swarm behavior accounting for strategic peer behavior. Collect real swarm data with an instrumented Bittyrant client. Refine model to include multiple swarms. Design and implement different strategies and experimentally analyze them using Emulab first, then Planetlab, and then over real swarms.
- Year 2: Investigate social-network based sophisticated topology adaptation strategies. Design and implement DTN case study. Integrate iPlane support into uswarm. Start design and implementation of DoI-resistant dissemination service. Design and implement in-network block cache using Click [41]. Refine topology adaptation models based on real measured data.

- Year 3: Design and implement a modified socket API to enable application use. Finish DoI case study. Investigate interaction with traffic engineering and virtualized architectures. Integrate support for legacy applications using the Oasis [52] toolkit. Integrate in-network caches into VINI. Test for security or privacy holes. Release all software.

H Synergy with Other FIND Efforts

The PIs will attend FIND meetings regularly to disseminate research findings as well as include network mechanisms being developed concurrently by other FIND researchers into uswarm’s design and implementation.

The proposed research maps well to the goals of FIND. Our data transfer architecture can co-exist with different network-layer and link-layer mechanisms including a connection-oriented network layer. uswarm is agnostic to network address structure, but relies on the network to correctly route packets to specified addresses as it relies on three-way handshakes to confirm a peer’s identity. uswarm goes beyond traditional client-server, peer-to-peer, and largely point-to-point connections and, instead has a more fluid notion of a data transfer architecture, namely, loose coordination, loose organization, and location-independent content. Finally, uswarm proposes in-network caching and tracking, and our research includes investigating the interaction with network management and virtualized architectures. (see Section C).

The uswarm project synergizes well with ongoing FIND projects. For example, in-network caching and tracking is one form of service in the ITDS project [35] and more network support for tracking can further benefit uswarm. *The Postcards from the Edge* [73] shares our goals of a delay-tolerant block transfer service assisted by in-network caches, but focuses on wireless environments. We expect the design of their network stack to influence ours. NIRA [64] investigates support for interdomain and intradomain source routing, which is complementary to location-independence offered by uswarm. Similar in spirit to *Markets for Service and Security* [55], uswarm can isolate bulk transfers from urgent browsing or VoIP and our incentive strategies support heterogeneous pricing and utilities (Section D). We will work with the PIs of the CONMAN [14] project to understand the requirements of a management plane that uswarm should support. The naming architecture in UIA [93] is synergistic and complementary to our DTN case study and push-based social networking applications in public spaces. Advances in network-layer security mechanisms such as capabilities [100] or SANE [84] will also benefit uswarm through the support for stronger identities, but requires careful integration as uswarm allows routers to “move the destination”. In summary, we believe that designing a scalable, fault-tolerant, incentive-compatible data transfer architecture will continue to be important for Internet services, no matter what mechanisms the network- and link-layer technologies support.

I Broader Impact and Education Plan

The proposed work if successful could fundamentally improve the way data is sent over the Internet. In particular, it can (i) enable a robust dissemination service resistant to denial-of-information attacks, which can prove crucial for emergency information services and national security; (ii) foster the growth of novel human-centric applications that are difficult to build today, and (iii) enable delay-tolerant data transfer for poorly connected environments, particularly in developing nations, for which the current networking stack is poorly suited.

Integrating Research and Teaching: The PIs at UMass-Amherst teach several courses and laboratories on communication networks, both at the undergraduate and graduate levels. In addition, they regularly offer seminars on advanced networking topics. These courses have enrollment of students from both the departments of Electrical and Computer Engineering, and Computer Science, and sometimes students from other departments such as mathematics and industrial engineering, and other Colleges such as Smith College. We plan to incorporate the results of the proposed project in graduate level seminars and in a Senior-level projects course. courses as well. Thus, we hope to disseminate the results of our research to a broad cross-section of students at UMass-Amherst.

International Outreach: Over the past five years, Towsley at UMass has conducted graduate level courses and seminars over the Internet to several Brazilian Universities. More recently Venkataramani and Towsley both ran a seminar that include not only Universities in Brazil but two universities in Italy as well. The experiences derived from these courses have led to the development and refinement of software useful for educational and collaborative efforts. Furthermore, they have led to interesting network performance measurements that have been published [12]. We will continue these efforts, disseminating the results of our research as well as using the teaching infrastructure as a testbed for exercising different swarm-based algorithms.

Undergraduates in Computer Science: The PIs manage an Internet research lab consisting of Cisco routers, and linux machines converted to routers. In addition the PIs have access to Planet Lab. In the past, several undergraduate students have been employed through REUs and other grants to work in these labs and or on Planet Lab. These students have moved on to either have successful careers in networking companies or to attain post-graduate degrees in networking. If this proposal is successful, we plan to apply for REU grants to involve undergraduate students in both the theoretical aspects of the research as well as to implement some of the algorithms.

All software resulting out of this research will be available for use by the community. We will also regularly disseminate our research via scholarly papers and IETF drafts where appropriate.

J Team and Prior Support

The PIs bring an eclectic set of skills that is particularly well suited to conduct the proposed research. PI Venkataramani has over five years experience in networked and distributed systems while Co-PI Towsley brings in over 25 years of experience in network performance modeling and analysis. Their combination of applied and analytical skills is valuable to the project. The PIs recently started collaborating and an upcoming Infocom'07 paper [63] together analyzing large swarms is directly related to the project.

PI Venkataramani has a track record of combining strengths in networking and distributed systems as well as theory and practice. He has co-developed TCP Nice [94] - a background transport protocol, *iPlane* [51] - an information plane for the Internet, and Bittyrant [71] - a selfish Bittorrent client. Other highlights include PRACTI replication [5], byzantine fault-tolerance with privacy [101], safe speculative replication [15], and algorithms for space-constrained [45] and bandwidth-constrained caching [95].

Co-PI Towsley's highlights include PFTK [69] - a widely used model of TCP, one bit congestion pricing for reliable multicast [65] - a William Bennett Prize paper, the first provably stable multipath congestion control and routing algorithm [27] among others. Recent relevant work includes game-theoretic models of overlay-underlay interaction [31, 102], and connection games [29, 30].

J.1 Prior Support

Venkataramani started his academic appointment at UMass Amherst in January 2005 and has not received NSF support.

Don Towsley was funded by NSF Grant ANI0085848 *Scalable Quality-of-Service Control for the Next Generation Internet: Fundamental Challenges and Effective Solutions* in collaboration with Lixin Gao and Jim Kurose. The grant was for the period 9/2000-8/2006. The project focused on problems in the areas of wireless networks, [49, 37, 48], quality of service [19, 40, 10], network performance modeling [97, 50, 18, 47, 27], protocols [36], peer-to-peer applications [26], network monitor placement [90], and network security [108, 105].

References

- [1] Ittai Abraham, Dahlia Malkhi, and Oren Dobzinski. LAND: stretch $(1 + \epsilon)$ locality-aware networks for DHTs. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 550–559, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [2] Akamai, Inc. Home Page. <http://www.akamai.com>.
- [3] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 313–324, New York, NY, USA, 2003. ACM Press.
- [4] James Aspnes and Gauri Shah. Skip graphs. In *Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, January 2003.
- [5] Nalini Belaramani, Mike Dahlin, Lei Gao, Amol Nayate, Arun Venkataramani, Praveen Yalagandula, and Jiandan Zheng. Practi replication. In *USENIX NSDI*, 2006.
- [6] Ashwin Bharambe, Cormac Herley, and Venkat Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *IEEE INFOCOM*, 2006.
- [7] Bittorrent home page. <http://www.bittorrent.com>.
- [8] CacheLogic Home Page. Advanced Solutions for P2P Networks Home Page. <http://www.cachelogic.com>.
- [9] M. Castro et al. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *SOSP*, 2003.
- [10] Yossi Chait, C. V. Hollot, Vishal Misra, Don Towsley, Honggang Zhang, and Yong Cui. Throughput differentiation using coloring at the network edge and preferential marking at the core. *IEEE/ACM Trans. Netw.*, 13(4):743–754, 2005.
- [11] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, 2003.
- [12] Weifeng Chen, Yong Huang, Bruno F. Ribeiro, Kyoungwon Suh, Honggang Zhang, Edmundo de Souza e Silva, Jim Kurose, and Don Towsley. Exploiting the IPID field to infer network path and end-system characteristics. In *Proceeding of the 2005 Passive and Active Measurement (PAM'05) Workshop*, March 2005.
- [13] L. Cherkasova and J. Lee. FastReplica: Efficient Large File Distribution within Content Delivery Networks. In *Proc. of USITS*, 2003.
- [14] NeTS-FIND: Towards Complexity-Oblivious Network Management PIs: Paul Francis and Jay Lepreau Institution(s): Cornell University and Univ. of Utah.
- [15] M. Dahlin, A. Iyengar, R. Kokky, A. Nayate, A. Venkataramani, and P. Yalagandula. SSR: Safe speculative replication. In *review: ACM Transactions on Computing Systems.*, January 2007.
- [16] DOME: Diverse Outdoor Mobile Environment. <http://prisms.cs.umass.edu/dome/>.
- [17] Michal Feldman, Kevin Lai, John Chuang, and Ion Stoica. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proceedings of the ACM Conference on Electronic Commerce*, June 2004.

- [18] Daniel R. Figueiredo, Benyuan Liu, Anja Feldmann, Vishal Misra, Don Towsley, and Walter Willinger. On tcp and self-similar traffic. *Perform. Eval.*, 61(2-3):129–141, 2005.
- [19] V. Firoiu, J.-Y. LeBoudec, D. Towsley, and Z.-L. Zhang. Theories and Models for Internet Quality of Service. *Proceedings of the IEEE*, 90(9):1565–1591, September 2002.
- [20] A Fluid Model of a Universal Swarm with Social Interest Classes, Working draft. <http://www.cs.umass.edu/arun/fluid.pdf>.
- [21] Michael J. Freedman, Eric Freudenthal, and David Mazières. Democratizing content publication with Coral. In *NSDI*, 2004.
- [22] S. Ganguly, A. Saxena, S. Bhatnagar, S. Banerjee, and R. Izmailov. Fast replication in content distribution overlays. In *Proc. of Infocom*, 2005.
- [23] Gnutella. <http://www.gnutella.com/>.
- [24] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of 19-th ACM Symposium on Operating Systems Principles*, October 2003. Bolton Landing, NY, USA.
- [25] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurement, analysis, and modeling of bittorrent-like systems. In *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'05), New Orleans, LA*, October 2005.
- [26] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2cast: peer-to-peer patching scheme for vod service. In *WWW '03. To appear in Journal of Multimedia Tools and Applications 2007*, pages 301–309. ACM Press, 2003.
- [27] H. Han, S. Shakkottai, C.V. Hollot, R. Srikant, and D. Towsley. Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the Internet. *IEEE/ACM Transactions on Networking*, 14(6), December 2006.
- [28] Nicholas Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *In proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, Seattle, WA, March 2003.
- [29] Honggang Zhang, Don Towsley, and Weibo Gong. TCP Connection Game: A Study on the Selfish Behavior of TCP Users. In *13th IEEE International Conference on Network Protocols (ICNP)*, November 2005.
- [30] Honggang Zhang, Giovanni Neglia, Don Towsley, and Giuseppe Lo Presti. On Unstructured File Sharing Networks. In *IEEE Infocom*, May 2007.
- [31] Honggang Zhang, Jim Kurose, and Don Towsley. Can an overlay compensate for a careless underlay? In *IEEE INFOCOM*, April 2006.
- [32] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast (keynote address). In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM Press.
- [33] NSF-ITR Award 0325868: Hyperion: Next-Generation Measurement Infrastructure and Application Use. <http://www.cs.umass.edu/csinfo/announce/itr2003.html>.

- [34] ICANN, Internet Corporation for Assigned Names and Numbers. <http://www.icann.org>.
- [35] NeTS-FIND: Service-Centric End-to-End Abstractions for Network Architecture PIs: Tilman Wolf Institution(s): UMass Amherst.
- [36] Ping Ji, Zihui Ge, Jim Kurose, and Don Towsley. A comparison of hard-state and soft-state signaling protocols. In *ACM SIGCOMM'03. (To appear in IEEE/ACM Transactions on Networking 2007)*, pages 251–262, 2003.
- [37] Ping Ji, Benyuan Liu, Don Towsley, Zihui Ge, and Jim Kurose. Modeling frame-level errors in gsm wireless channels. *Perform. Eval.*, 55(1-2):165–181, 2004.
- [38] P. Karbhari, E. Zegura, and M. Ammar. Multipoint-to-point session fairness in the internet, 2003.
- [39] KaZaA. <http://www.kazaa.com/>.
- [40] P. Key, L. Massoulie, and J. Shapiro. Service differentiation for delaysensitive applications: An optimisation-based approach. *Performance Evaluation*, 2001.
- [41] Eddie Kohler et al. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3), August 2000.
- [42] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *SOSP*, 2003.
- [43] Arnaud Legout, Guillaume Urvoy-Keller, and Pietro Michiardi. Rarest First and Choke Algorithms are Enough. In *Proc. of IMC*, 2006.
- [44] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. In *Proceedings of the 2006 USENIX Operating Systems Design and Implementation (OSDI)*, November 2006.
- [45] Xiaozhou Li, C. Greg Plaxton, Mitul Tiwari, and Arun Venkataramani. Online hierarchical cooperative caching. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 74–83, New York, NY, USA, 2004. ACM Press.
- [46] Xiaozhou Li and Greg Plaxton. On name resolution in peer-to-peer networks. In *Workshop on Principles of Mobile Computing*, 2002.
- [47] B. Liu, D.R. Figueiredo, Y. Guo, and J. Kurose and D. Towsley. On the efficiency of fluid simulation versus packet-level simulation. *Computer Networks*, 50(12):1974–1994, August 2006.
- [48] B. Liu, D. Goeckel, and D. Towsley. TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks. In *Proceedings of Global Internet Symposium*, November 2002.
- [49] B. Liu, Z. Liu, and D. Towsley. the capacity of hybrid wireless networks, March 2003.
- [50] Yong Liu, Francesco L. Presti, Vishal Misra, Donald F. Towsley, and Yu Gu. Scalable fluid models and simulations for large-scale ip networks. *ACM Trans. Model. Comput. Simul.*, 14(3):305–324, 2004.
- [51] Harsha Madhyastha, Tomas Isdal, Mike Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane: An information plane for distributed services. In *USENIX Operating System Design and Implementation (OSDI) (to appear)*, December 2006.

- [52] Harsha Madhyastha, Arun Venkataramani, Thomas Anderson, and Arvind Krishnamurthy. Oasis: An Overlay-aware Network Stack. *ACM SIGOPS Operating Systems Review, Planetlab Special Issue*, Vol. 40(1), Januray 2006.
- [53] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proceedings of the 21st annual ACM symposium on Principles of distributed computing*. ACM Press, 2002.
- [54] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *USITS*, 2003.
- [55] NeTS-FIND: Title: Market-Enabling Network Architecture PIs: Jean Walrand, Venkat Anantharam, John Musachio and Shyam Parekh Institution(s): University of California, Berkeley; University of California, Santa Cruz and Bell Laboratories, Lucent Technologies.
- [56] P. Maymounkov and D. Mazieres. Kademia: A peer-to-peer information system based on the xor metric, 2002.
- [57] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating key management from file system security. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP '99)*, pages 124–139, Kiawah Island, South Carolina, December 1999.
- [58] IETF RFC 1321: The MD5 Message Digest Algorithm. <http://www.ietf.org/rfc/rfc1321.txt>.
- [59] R. Morris. TCP behavior with many flows. In *Proceedings of ICNP*, 1997.
- [60] Mehul Motani, Vikram Srinivasan, and Pavan Nuggehalli. Peoplenet: engineering a wireless virtual social network. In *ACM MOBICOM*, pages 243–257, 2005.
- [61] Myspace Website. <http://www.myspace.com>.
- [62] Napster website. <http://www.napster.com/>.
- [63] Giovanni Neglia, Giuseppe Reina, Honggang Zhang, Donald Towsley, Arun Venkataramani, and John Danaher. Availability in bittorrent systems. In *IEEE Infocom'07*, May 2007.
- [64] NeTS-FIND: An Internet Architecture for User-Controlled Routes PI: Xiaowei Yang Institution(s): UC Irvine.
- [65] Jorg Nonnenmacher, Ernst W. Biersack, and Don Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Trans. Netw.*, 6(4):349–361, 1998.
- [66] Workshop on economics of peer-to-peer systems, 2003, berkeley, ca. <http://www.sims.berkeley.edu/p2pecon/>.
- [67] Workshop on economics of peer-to-peer systems, 2005, philadelphia, pa. <http://p2pecon.cs.cornell.edu/>.
- [68] Workshop on economics of networked systems, 2006, ann arbor, mi. <http://www.cs.duke.edu/nicl/netecon06/>.
- [69] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: A simple model and its empirical validation. *CCR*, 28(4), 1998.
- [70] KyoungSoo Park and Vivek S. Pai. Scale and performance in the CoBlitz large-file distribution service. In *Proceedings of the Third Symposium on Networked Systems Design and Implementation (NSDI 2006)*, San Jose, CA, May 2006.

- [71] Mike Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bittorrent? In *USENIX Networked System Design and Implementation (NSDI) (to appear)*, April 2007.
- [72] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA '97: Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pages 311–320, 1997.
- [73] NeTS-FIND: Postcards from the Edge: A Cache-and-Forward Architecture for the Future Internet PIs: Roy Yates, Dipankar Raychaudhuri, Sanjoy Paul, James Kurose Institution(s): Rutgers WINLAB, UMass.
- [74] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, February 2005.
- [75] Dongyu Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of Sigcomm*, 2004.
- [76] Lili Qiu, Yang Richard Yang, Yin Zhang, and Scott Shenker. On selfish routing in internet-like environments. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 151–162, New York, NY, USA, 2003. ACM Press.
- [77] Lili Qiu, Yin Zhang, and Srinivasan Keshav. Understanding the performance of many TCP flows. *Computer Networks (Amsterdam, Netherlands: 1999)*, 37(3–4):277–306, 2001.
- [78] V. Ramasubramanian and E. Sirer. The design and implementation of a next generation name service for the internet. In *ACM SIGCOMM*, 2004.
- [79] Venugopalan Ramasubramanian and Emin Gn Sirer. Beehive: Exploiting power law query distributions for o(1) lookup performance in peer to peer overlays. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04)*, pages 331–342, San Francisco, CA, USA, March 2004.
- [80] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 31(4), pages 161–172, October 2001.
- [81] Tim Roughgarden and Éva Tardos. How bad is selfish routing? In *IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2000.
- [82] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.
- [83] RSS Advisory Board website. <http://www.rssboard.org/>.
- [84] NeTs-FIND: Designing Secure Networks from the Ground-Up, PIs: Dan Boneh, David Mazieres, Mendel Rosenblum, Aditya Akella, Nick McKeown Institution(s): Stanford University, University of Wisconsin-Madison.
- [85] SETI@Home. <http://setiathome.berkeley.edu>.
- [86] IETF RFC 3174: US Secure Hash Algorithm 1 (SHA1). <http://www.ietf.org/rfc/rfc3174.txt>.

- [87] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A Cooperative Bulk Data Transfer Protocol . In *Proc of IEEE Infocom*, Mar 2004.
- [88] Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Replication for web hosting systems. *ACM Comput. Surv.*, 36(3):291–334, 2004.
- [89] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.
- [90] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: complexity, heuristics, and coverage. *Journal of Computer Communications*, 29(10):1564–1577, June 2006.
- [91] The Globus Project. GridFTP: Universal Data Transfer for the Grid, 2000.
- [92] Niraj Tolia, Michael Kaminsky, David G. Andersen, and Swapnil Patil. An architecture for internet data transfer. In *USENIX NSDI*, May 2006.
- [93] NeTS-FIND: User Information Architecture PIs: Robert Morris and Frans Kaashoek Institution(s): MIT CSAIL.
- [94] A. Venkataramani, R. Kokku, and M. Dahlin. TCP-Nice: A mechanism for background transfers. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, December 2002.
- [95] A. Venkataramani, P. Weidmann, and M. Dahlin. Bandwidth constrained placement in a wan. In *Proceedings of the 20th Symposium on the Principles of Distributed Computing*, August 2001.
- [96] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin. Potential costs and benefits of long-term prefetching for content-distribution. *Elsevier Computer Communications*, 25(4):367–375, March 2002.
- [97] W. Wei, B. Wang, and D. Towsley. Continuous-time hidden Markov models for performance evaluation. *Performance Evaluation*, 49(1-4):129–146, September 2002.
- [98] X. Yang. NIRA: A New Internet Routing Architecture. In *ACM SIGCOMM FDNA 2003 Workshop, Karlsruhe*, August 2003.
- [99] X. Yang and G. de Veciana. Service capacity of peer to peer networks, 2004.
- [100] Xiaowei Yang, David Wetherall, and Thomas Anderson. A dos-limiting network architecture. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 241–252, New York, NY, USA, 2005. ACM Press.
- [101] Jian Yin, Jean-Philippe Martin, Arun Venkataramani, Lorenzo Alvisi, and Mike Dahlin. Separating agreement from execution for byzantine fault tolerant services. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 253–267, New York, NY, USA, 2003. ACM Press.
- [102] Yong Liu, Honggang Zhang, Weibo Gong, and Don Towsley. On the Interaction Between Overlay Routing and Underlay Routing. In *24th IEEE Conference on Computer Communications (INFOCOM 2005)*, March 2005.
- [103] YouTube Website. <http://www.youtube.com>.

- [104] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Towsley. On optimal routing with multiple traffic matrices. In *IEEE INFOCOM*, 2004.
- [105] X. Zhang, G. Neglia, J. Kurose, and D. Towsley. Performance modeling of epidemic routing. In *IFIP'06. To appear in Journal of Computer Networks 2007*, 2007.
- [106] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming, 2005.
- [107] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. In *IEEE Journal on Selected Areas in Communications*, 2003.
- [108] C.C. Zou, N. Duffield, D. Towsley, and W. Gong. Adaptive defense against various network attacks. *IEEE Journal on Selected Areas in Communications*, 24(10):1877–1888, October 2006.