

Designing and Implementing Cooperative Services

Harry C. Li

Cooperative services are an increasingly popular way to deploy applications. In these services, computers controlled by different entities are supposed to work together to achieve a common goal, such as file-sharing [6, 12], mesh routing [2], and peer-to-peer backup [1, 7]. The resulting systems can be more fault-tolerant, scalable, and less expensive than traditional client-server approaches. However, a cooperative service may never see these benefits if it does not tolerate Byzantine users who may disrupt the service or selfish ones who may try to use it without contributing their fair share.

One of the key challenges in designing a cooperative service is ensuring that participants actually cooperate. Several deployed applications [6, 12] and research prototypes [7, 17] include incentives and punishments to encourage cooperation. However, selfish participants still frequently find ways to cheat. For example, in the KaZaA [12] network almost half the users falsify their contributions by using a hacked binary [13]. In BitTorrent [6], researchers have found ways to free-ride [19]. These vulnerabilities are symptoms of a deeper problem: almost no cooperative service *rigorously* shows that its incentives and punishments are sufficient to induce cooperation.

In my research, I aim to build cooperative systems that tolerate the actions of Byzantine users while removing incentives for selfish users to cheat. In this adversarial environment, constructing systems that work well and are robust requires me to step back and forth between practice and theory, something that distinguishes my research from that of many others who design these systems. I believe that pursuing the practice-theory cycle—letting practice guide theory and vice versa—is necessary in building robust and practical systems. This approach requires me to be persistent, resourceful, and creative. Drawing upon these traits and my training so far, I plan to *i*) develop the theoretical framework necessary to reason about selfish and Byzantine behaviors in cooperative services, *ii*) design efficient and scalable mechanisms in this framework to address practical issues in building real services, and *iii*) implement large-scale prototypes that are competitive with if not better than existing less robust works.

1 Current Research

My efforts to construct cooperative services that are both practical and rigorous have focused on peer-to-peer (p2p) live streaming services. My colleagues and I have built two such systems, BAR Gossip [15] and FlightPath [14], that tolerate Byzantine and selfish peers. The key challenge we face in building a p2p live streaming system is in how to encourage cooperation. Our first approach draws from game theory concepts that require users to expect *no gain* from attempting to cheat. Our experience with BAR Gossip is that such a purist approach, while admirable, is too limiting in building real systems where performance is a key concern. Therefore, our second approach shifts to a weaker goal, seeking to prove that users expect, at most, a *small gain* from attempting to cheat. Our second system, FlightPath, requires less resources than its predecessor and improves stream quality by several orders of magnitude. Our contributions represent significant steps forward both in rigorously designing practical cooperative services and in implementing robust p2p live streaming systems. Moreover, BAR Gossip and FlightPath are exemplars of the practice-theory cycle. I discuss these systems in greater detail below to provide further insight into my research style.

BAR Gossip: A Rigorous Approach We base the design of BAR Gossip on gossip protocols [4, 8] in which peers periodically exchange recently received data with random neighbors. We construct a gossip protocol, Balanced Exchange, to be a Nash equilibrium [16]—a condition in which no selfish peer has an incentive to act alone in trying to cheat the system. We immediately face two practical hurdles. First, gossip’s robustness stems from its randomness, a trait that makes it difficult to enforce obedience since peers can mask suspicious behavior as the product of improbable though not impossible events. We overcome the randomness problem by creating a verifiable pseudo-random algorithm. Second, the fair exchange of data between two peers is provably impossible to achieve without a trusted arbiter [10], which can be a bottleneck at large scales. We sidestep the impossibility result by designing an exchange primitive that is provably *fair enough*, guaranteeing that selfish peers do not attempt to cheat others in trades.

So far, our desire to ensure that cheating is not beneficial results in a theoretically appealing Balanced Exchange protocol that performs poorly. To remedy this performance issue, we introduce a second protocol,

Optimistic Push. This protocol is similar to Balanced Exchange, but it allows peers to trade data with each other with the hope, rather than the certainty, that they will receive the same number of useful data packets in return. Together with Balanced Exchange, this protocol noticeably improves performance but has a theoretical drawback: We cannot prove that the Optimistic Push protocol is a Nash equilibrium, although our experiments suggest it is not easy to cheat the protocol.

FlightPath: A Rigorous *and* Practical Approach Whereas Balanced Exchange reflects how we let our theoretical goals influence our design, we now do the reverse. Our experience suggests that strict equilibria (e.g., Nash equilibria) may not let us reach our practical and theoretical goals. We therefore turn to approximate equilibria [5], and in particular, ϵ -Nash equilibria in which selfish peers expect to gain at most a factor of ϵ from unilateral deviations. We use these equilibria to trade resilience to cheating for practical concerns like performance, a tradeoff that the game theory community has never pursued.

FlightPath—our streaming system based on ϵ -Nash equilibria—significantly improves stream quality over Balanced Exchange and Optimistic Push. The strength of these equilibria is that they allow us to balance enforcing obedience against providing enough flexibility to adapt to challenging situations. An example of this balancing act is in how we craft FlightPath’s partner selection algorithm to both limit the number of partners that Byzantine peers can simultaneously contact while offering enough choices for other peers to select good trading partners.

In addition to their practical benefits, ϵ -Nash equilibria also have theoretical ones. We can prove that FlightPath is an ϵ -Nash equilibrium without relying on experiments (as we did with Optimistic Push) that only suggest this condition.

2 Future Work

Our efforts with BAR Gossip and FlightPath reveal a wealth of open problems in constructing cooperative services. Some of these problems are short-term hurdles that we will resolve as we deploy FlightPath in the coming months. Others, highlighted below, are more fundamental. I plan to tackle these issues as part of my long-term goal of developing the theory and practice behind building cooperative systems.

Altruism Many p2p systems rely on the existence and participation of altruistic users who contribute more than is required. Taking advantage of user altruism can increase the robustness and performance of a system and is a desirable goal. Yet, we need to be careful how we hand out such generosity so that it is not abused by selfish peers seeking to contribute less. I plan to explore how to take advantage of altruism while still limiting the possible gains from cheating to a small value.

Network Heterogeneity Although users’ network connections can vary greatly in practice, we currently have no mechanisms that leverage additional bandwidth users may have so as to help those who have little bandwidth to offer. One of the main difficulties is in incentivizing peers to use their extra bandwidth to help the system in a way that cannot be manipulated.

To provide these incentives in live streaming, one approach is to leverage the layered coding inherent to many video codecs. I would like to split video content into multiple layers and embed a key into each layer such that access to a given layer requires sufficient participation at all lower layers [11]. Furthermore, users who can gain access to higher layers can be required to trade data packets at lower layers using less favorable ratios, so as to aid those users without as much bandwidth. Assuming that users’ first priority is to gain access to the highest possible layer, such a restriction can force those with more to help those with less.

Content Distribution Networks I plan to explore how to apply game-theory concepts to content distribution networks (CDNs) [9, 21]. The main challenge in incentivizing cooperation in a CDN setting is that users may not be interested in the same content at the same time. Therefore, the bartering mechanism we use for live streaming may be inapplicable. An initial solution is to introduce credits into a system to make exchanges more flexible.

Two common critiques of credit-based systems are that they often offer loopholes that participants can abuse or are overly complicated [18, 20]. I plan to explore how feasible it is to design simple, light-weight, and theoretically sound credit mechanisms while building a large-scale content distribution network.

Fair Exchange Service I would also like to design and implement a practical large-scale fair exchange service. Since the exchange of goods and resources is an important part of p2p systems, a fair exchange service would be a very useful building block for novel cooperative services, such as CDNs, file-sharing applications, and video-on-demand services.

To build this service, my plan is to combine game theory with work in *optimistic fair exchange* protocols [3, 20]. Like all fair exchange protocols, optimistic ones use a trusted arbiter to guarantee fairness. What makes these protocols optimistic is that the arbiter is never involved in problem-free trades; it is only used to resolve disputes. However, if selfish parties seek to conserve bandwidth, their actions can cause every exchange to require arbitration and make the arbiter the bottleneck of the system.

We may be able to borrow techniques from BAR Gossip and FlightPath to limit the bandwidth savings possible from cheating in each exchange, thereby inducing cooperation and avoiding arbitration in most exchanges. I would like to explore the practical limits of such a fair exchange service and understand the kinds of applications that can benefit most from it.

Designing cooperative services requires approaches that take into account users who may be selfish or Byzantine. Our research has taken significant strides in this space. My goal is to continue to develop the theories, tools, and mechanisms for building systems in this challenging area.

References

- [1] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR fault tolerance for cooperative services. In *SOSP '05*, pages 45–58, Oct. 2005.
- [2] I. F. Akyildiz and X. Wang. A survey on wireless mesh networks. *Communications Magazine, IEEE*, 43(9):S23–S30, 2005.
- [3] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *CCS '97*, pages 7–17, New York, NY, USA, 1997. ACM.
- [4] K. P. Birman, M. Hayden, O. Oskasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM TOCS*, 17(2):41–88, May 1999.
- [5] S. Chien and A. Sinclair. Convergence to approximate nash equilibria in congestion games. In *SODA '07*, pages 169–178, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.
- [6] B. Cohen. Incentives build robustness in BitTorrent. In *P2PECON '03*, June 2003.
- [7] L. P. Cox and B. D. Noble. Samsara: honor among thieves in peer-to-peer storage. In *SOSP '03*, pages 120–132, 2003.
- [8] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *SOSP '87*, Aug. 1987.
- [9] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *NSDI '04*, Berkeley, CA, USA, 2004. USENIX Association.
- [10] B. Garbinato and I. Rieckbusch. Impossibility results on fair exchange. Technical Report DOP-20051122, Université de Lausanne, Distributed Object Programming Lab.
- [11] S. Gorinsky and H. Vin. The utility of feedback in layered multicast congestion control. In *NOSSDAV '01*, June 2001.
- [12] Kazaa. <http://www.kazaa.com>.
- [13] Kazaa Lite. http://en.wikipedia.org/wiki/Kazaa_Lite.
- [14] H. C. Li, A. Clement, M. Marchetti, Kapristsos, L. Robison, L. Alvisi, and M. Dahlin. Flighpath: Obedience vs. Choice in cooperative services. In *Proceedings of OSDI '08*, Dec. 2008.
- [15] H. C. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR Gossip. In *Proceedings of OSDI '06*, pages 191–204, Nov. 2006.
- [16] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54:286–295, Sept 1951.
- [17] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. Incentives-compatible peer-to-peer multicast. In *P2PECON '04*, June 2004.
- [18] J. Shneidman, C. Ng, D. C. Parkes, A. AuYoung, A. C. Snoeren, A. Vahdat, and B. Chun. Why markets could (but don't currently) solve resource allocation problems in systems. In *HOTOS '05*, Berkeley, CA, USA, 2005. USENIX Association.
- [19] M. Sirivianos, J. H. Park, R. Chen, and X. Yang. Free-riding in bittorrent networks with the large view exploit. In *IPTPS '07*, February 2007.
- [20] M. Sirivianos, J. H. Park, X. Yang, and S. Jarecki. Dandelion: cooperative content distribution with robust incentives. In *USENIX*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.
- [21] L. Wang, V. Pai, and L. Peterson. The effectiveness of request redirection on cdn robustness. *SIGOPS Oper. Syst. Rev.*, 36(SI):345–360, 2002.