Meta-Headers: Top-Down Networking Architecture with Application-Specific Constraints

Murat Yuksel
Computer Science and Engineering, University of Nevada - Reno,
1664 N. Virginia Street, MS 171, Reno, NV 89557, USA.
Email: yuksem@cse.unr.edu

***Abstract*—The direction of cross-layer wireless research has been prominently bottom-up, i.e., optimizing higher layer protocols based on the characteristics of the underlying wireless communication medium. Though this approach has been essential for efficient use of scarce wireless connectivity and bandwidth, there is a recent trend for massive availability of wireless resources through initiatives such as municipal WiFi and community wireless. The key metric to optimize is becoming application utility rather than the wireless resources. This paradigm shift calls for top-down cross-layer designs in wireless protocols, where application-specific requirements can be reconciled. We introduce "meta-headers" vertically traveling across the network stack and enabling top-down optimizations in the architecture.**

***Index Terms*—cross-layer wireless design; application-based networking; network architecture**

I. Introduction

As the reach of networked devices increases, the network infrastructure has to support *application-specific* designs due to the increasing variety of applications such as mobile data, wireless peer-to-peer, vehicular networks, and sensor networks. The growth in wireless requires the network to handle disruption-tolerance and large scale of nodes in the million-to-billion range. These requirements have pushed the wireless research to find cross-layer invariants and incorporate them into the networking stack via *vertical components*.

Mainly focusing on physical, network, and application layers of the stack, these vertical components have attempted to bridge the gaps between these three fundamental layers as shown in Figure 1d. For example, previous work showed that very flexible routing functions can be implemented [1] by using *network-specific* properties such as geographic routing capability, and that one can achieve routing at very large scales by abstracting *hardware-specific* properties like localization capability of nodes [2] spectrum characteristics [3] or environment characteristics [4]. *Application-specific* design in networking research has been performed in the sensor networks area, especially in routing functions customized for applications [5] [6]. It is a necessity to make such application-specific enablers a systematic practice of wireless protocol design and find architectural innovations to realize them.

As dynamic networks such as MANETs and peer-to-peer are getting integrated with the Internet, the network stack needs to have more cross-layer designs with systematic vertical components, as shown in Figure 1. The direction of cross-layer wireless research has been prominently *bottom-up*, i.e.,

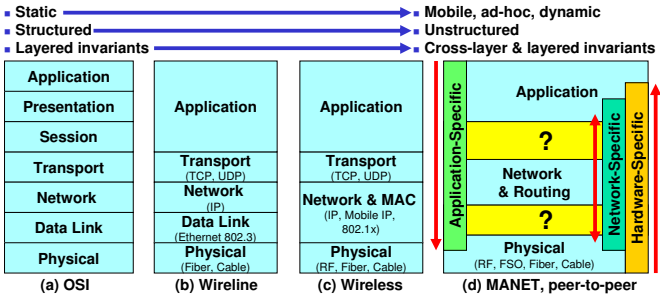

Fig. 1. Growth of the protocol stack as the networking environments become dynamic and unstructured.

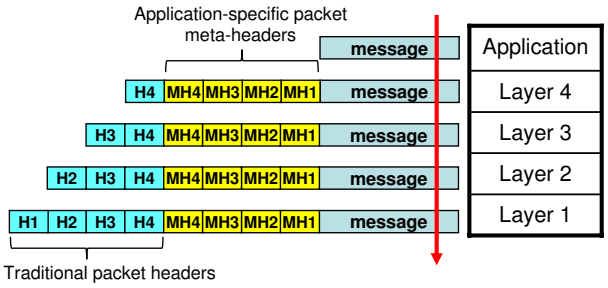

Fig. 2. One possible placement of packet meta-headers at the source node where the application is located.

optimizing higher layer protocols based on the characteristics of the underlying wireless communication medium. Though this approach has been essential for efficient use of scarce wireless connectivity and bandwidth, there is recent trend for massive availability of wireless resources, e.g., [7] [8] [9]. *The key metric to optimize is becoming application utility rather than the wireless resources*, and application-specific goals cannot be considered in the existing bottom-up designs. This paradigm shift calls for *top-down* cross-layer designs where application-specific requirements can be reconciled.

To facilitate top-down cross-layer and more integrated vertical optimizations without breaking the well-needed isolation properties of the layered stack architecture, we use the Application Layer Framing (ALF) principles [10]. The essence of our ALF-based approach is that network protocol services at lower layers can best be useful when applications' characteristics and intents are conveyed to the lower layers. Specifically, we propose to embed "packet meta-headers" (see

Figure 2) which may *vertically* travel across the network stack and establish a "vertical communication channel" among the traditional layers. These meta-headers will co-exist with the traditional per-layer packet headers which are installed or stripped off when crossing to the lower or the upper layer. Applications will be able to communicate their intent across all the protocol layers by attaching the meta-headers to data.

The vertical meta-headers in the top-down architecture will serve as the *Lagrange multipliers* [11] [12] between the application and the network. User applications can be viewed as entities trying to *maximize utility* from available network services while the network is trying to *minimize cost* of providing the services. Essentially, the meta-headers will work as a communication channel between these two concurrent optimizations by conveying the the constraints, i.e., network services as constraints to the user applications, and application-specific constraints to the network.

One particular viewpoint is that the user applications and the network dynamically *re-adjust* themselves to achieve these optimizations. However, time and space granularity of this re-adjustment is currently too coarse, not suitable for highly dynamic environments like MANETs. For instance, it has been many years since open WiFi points exist at several locations [13], but very few wireless applications [14] have actively leveraged them yet. A similar story can be told for applications placed at a multi-homed device but not leveraging multiple paths. Arguably the root cause of this *lack of vertical optimizations* is the fact that applications cannot (i) get informed about the available lower layer services in a timely manner, and (ii) scalably and reliably convey their intents to the lower layers. This position paper also illustrates example of such a vertical optimization via meta-headers, i.e., top-down value/QoS choices (layers 5, 3, and 2).

The rest of the paper first covers related work. Then, in Section III, we present our top-down architecture. Section IV shows examples of possible vertical optimizations in the proposed architecture. Finally, we summarize in Section V.

II. Related Work

Cross-Layer Signaling: There have been several proposals to communicate hints and notifications across layers of the protocol stack [15]. Though the performance improvement via vertical optimizations [16] has been the main focus, little has been investigated on how such cross-layer designs will be implemented. Proposals included upward or downward [17] [18] information flow across the layers. Mostly the implementation was based on a new interface between layers [19], an explicit cross-layer database serving all layers at once [20], or simply a joint design of the layers involved [21].

Similar to our work, some previous research considered addressing the cross-layer communication *architecturally*. The motivating force for exploring different architectures has been to make the layered protocol stack more flexible so that more integrated optimizations can be done. Proposals included a role-based architecture using a protocol heap rather than the traditional stack [22] or in general a stack with relaxed layering constraints [23] [24]. Dividing existing layers into micro-layers and establishing vertical signaling across them was recently proposed to improve transport performance [25]. Though integrated optimization of layers is of crucial interest, horizontal layering is likely to stay due to its unbeatable benefits in software implementation.

Application-Based Operation: Application-based networking has been studied in the context of sensor networks. Active [26] and cognitive [27] networking are efforts towards the same goal of customizing network behavior for various user applications. In the general context of routing, defining application-specific custom routes via user-defined choices [28] or a declarative language [29], have attracted interest. However, these efforts either require significant router computation or autonomous system level route descriptions, both are impractical in wireless routing. The very first application-based wireless routing work was an extreme scheme of Dynamic Source Routing (DSR) [30], which gives a full flexibility to the user application to define the routes. Since DSR could not scale to sufficient number of nodes, schemes like Trajectory-Based Routing (TBR) [1] and landmark selection [31] were proposed to reduce the packet header costs and yet still give reasonable flexibility for users to define their desired routes as trajectories. Recent wireless routing studies focused on customizing routing metric for applications [32] and centralized optimization of routing for application-based goals [33]. Our work aims to standardize these efforts through integration of "application-specific constraints" into service interfaces.

III. Function Placement and Meta-Headers

Realization of application-specific designs require architectural support from the network. The current network stack only allows a layer to talk to the layer above or below, and the majority of the cross-layer work has been "bottom-up" where characteristics of the hardware or the network environment being leveraged at the layers above. We propose a new top-down networking architecture using "meta-headers" in addition to the traditional packet headers, where not only bottom-up inter-layer designs are allowed but also the "top-down" ones. Recent discussions in the NSF's FIND initiative have stressed this issue and indicated that the network architecture should address application-specific requirements. Duties like trust management and end-to-end reliability are considered to be possible with a top-down architectural perspective. [34]

Why not continue merging layers? The wireless research community has found more comfort with joint designs across the functions of the protocol stack, rather than using cross-layer signaling without breaking layers. However, this practice makes it very hard to standardize protocols, which is essential for wide deployment of new networking technologies. We believe that the efforts should continue on questing for both horizontal and vertical invariants in the protocol stack. Arguably, as we elaborated in Figure 1d, there are three levels of essential functions in the network stack: application, network, and physical. We believe that merging any two of these three levels will result in a non-standardizable technology. Such

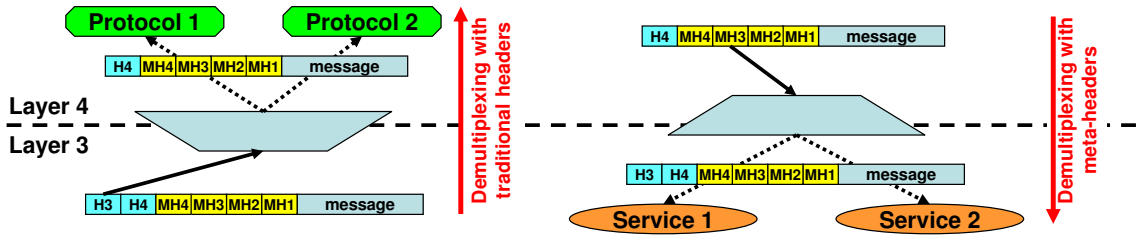

Fig. 3. Functioning of Meta-headers and Headers When Crossing a Layer: Meta-headers demultiplex the packet to the lower layer "customized" service primitives, while traditional headers demultiplex to the upper layer protocols.

highly integrated/merged designs will not be able to cope with ever-increasing diversity of the wireless applications. Further, integrating lower level functions with higher layer ones will not allow those lower level functions to be available as a substrate for other higher layer protocols. With the obvious trend of increasing low-level wireless resources, such highly integrated designs will be unnecessarily too costly. A well-known example of such overly integrated design is cellular phones being bundled with the wireless access provider (especially in the U.S.). Though such integration allows providers to optimize various wireless resources and components, it has become clear that open phone platforms allowing opportunistic usage of low-level substrates (e.g., WiFi when available) will be more successful in delivering what user applications desire. In the next subsections, we will describe a candidate architectural approach to the problem of effective cross-layer signaling without breaking highly-standardized horizontal layers.

A. An Application Talking to Lower Layers

In the layered network stack architecture, a layer composes its service primitives to the upper layer via service primitives of the lower layer. This conventional architecture makes each layer implement a discrete set of services and export a well-defined interface to be used by its immediate upper layer. The interface standardization provides isolation between functional modules of distinct layers. Such isolation is useful for allowing subsystems to be developed independently. Quoting Clark and Tennenhouse [10]: "*a major architectural benefit of such isolation is that it facilitates the implementation of subsystems whose scope is restricted to a small subset of the suite's layers.*" The layered architecture, however, may cause inefficiencies in end-to-end function implementation [10], and should not be considered as the only approach as more integrated and vertical designs might perform significantly better. This consideration establishes the basis for the concept of Integrated Layer Processing (ILP) [10].

We introduce the notion of "packet meta-header" to allow a systematic way of performing ILP while not hurting the nice isolation properties of the current layered network stack. This new network stack design will enable the application layer be capable of talking to the lower layers as a set of recommendations embedded into packet meta-headers, potentially on a packet-by-packet basis. Figure 2 illustrates *one possible* placement of meta-headers into the network stack. The application inserts its desired service action at layer i as a meta-header MHi along with the message data. We represent this application message as `<meta-header,message>`. As `<meta-header,message>` travels down the stack, protocols at the lower layers can treat it as a regular data message without caring about the meta-headers, or they can recompose the `<meta-header,message>` by updating its meta-headers. From layer k's perspective, the service action/primitive desired by the application to be applied on the packet is included in the meta-header MHk. Note that during the recomposition of `<meta-header,message>`, layers can update the specific meta-headers and envelope a desired service message to the lower layers. This rhymes well with the owner privilege issues since layers lower than k will most likely be owned by the owner of layer k. When a packet crosses into a new domain, the layer processing the meta-headers can recompose them such that local rules (e.g., routing policies) can be forced.

1) Demultiplexing with Meta-Headers: The traditional layered network stack only allows bottom-up demultiplexing of lower layer data units onto different "protocols" at the higher layer. The meta-header stack architecture allows top-down demultiplexing onto various "service primitives". Figure 3 shows two kinds of demultiplexing. As shown in Figure 3, layer 4 specifies a set of meta-headers with semantic hints to the lower layers for packet processing. In layer 3 (and every lower layer), these meta-headers are mapped to the set of service primitives that are implemented at this layer. By specifying an appropriate combination of hints, the data packet will be processed with the desired service primitives at lower layers. In the example, layer 3 specifies two service primitives, selection of which is based on the content of MH3. This architecture allows each layer to independently apply the appropriate service primitive for the packet as using hints provided by the originator of the data.

2) Talking to Intermediate Routers and End-to-End Coordination: A design issue with our meta-header stack architecture is the processing of these meta-headers at intermediate routers. A router capable of processing meta-headers will ideally not remove them when taking the datagram up through the stack, and then interpret the meta-header message pertaining to it (i.e., MH3) and act upon the content of the meta-header. Alternatively, as shown in Figure 4, meta-headers may get converted/embedded into traditional headers and the receiving side can decode them from the headers.

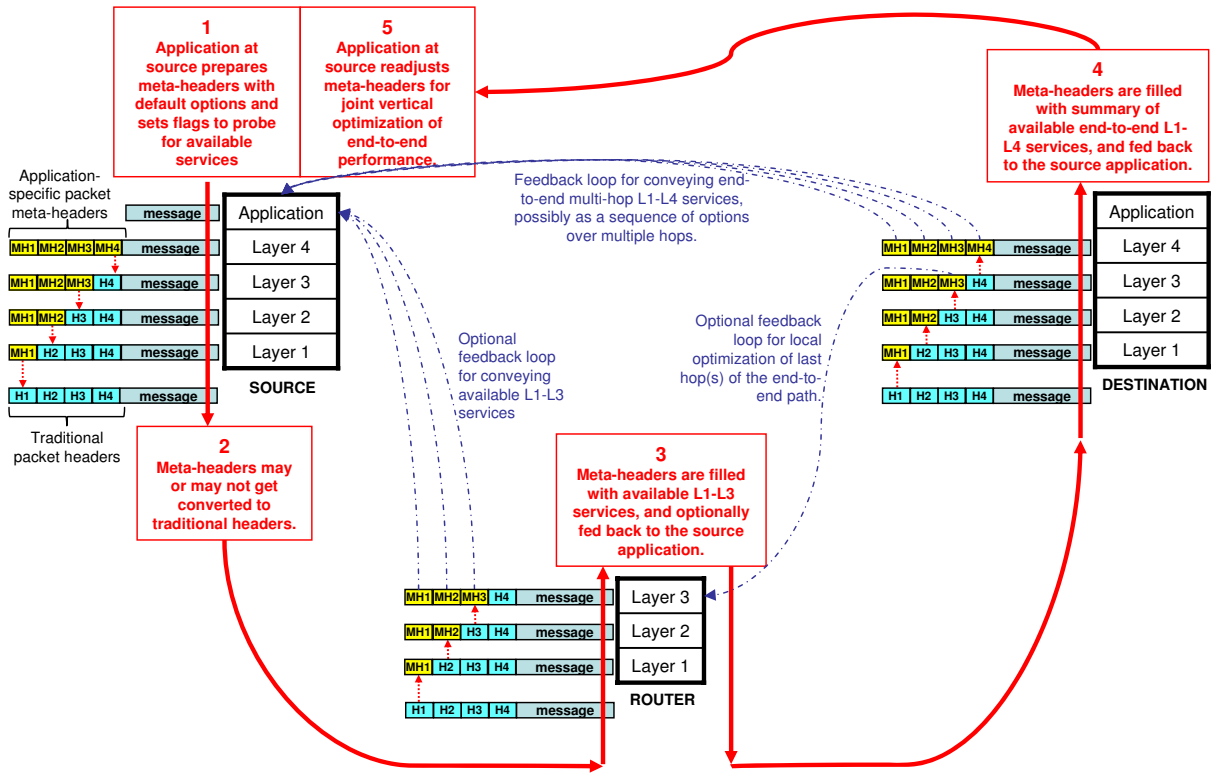

Fig. 4. End-to-End Coordination with Meta-Headers: Traditional packet headers are stripped off while going to an upper layer at an intermediate router. However, packet meta-headers *may* be kept within the datagram when going to an upper layer, allowing the application at the source to talk to the layers of an intermediate router. Alternatively, to reduce packet size, the meta-headers can be *converted* to headers.

Multi-provider and multi-domain nature of the Internet is key in its success to reaching billions of nodes. However, the benefits of our meta-headers can be maximized when the knowledge about service primitives of all routers along the path towards destination is available. Naturally, exposure of such information is not welcomed by ISPs and domain owners. ISPs, on the other hand, will have business incentives to provide a summary of service primitives provided by their inner routers, rather than specific router setup information. Yet another possibility is to include the specific service primitive in the meta-header itself, and ask the intermediate routers to execute the primitive included in the meta-headers. This kind of interaction with intermediate routers is similar to active networking [26] where packets include the processing desired action from the router. However, in our top-down architecture, (i) lower layers are not required to "execute" a function included in a meta-header, but rather we only allow meta-headers to select one of the available lower layer services; and (ii) the meta-headers can be used to communicate across all layers, while active networking involved interaction between layers 3 and 4 only. Thus, our *top-down networking architecture is a generalization of several possible designs between end systems and multi-domain core systems.*

3) Meta-Header Security: To make packet meta-headers robust to potential security threats, it is possible to add appropriate cryptographic mechanisms to the meta-headers themselves. This will allow only authorized entities to access the meta-headers in a meaningful way. In particular, there is a clear tradeoff between the extent of trust in the system and data processing efficiency. In a completely trusted MANET, no cryptographic operations are needed on meta-headers. This approach exposes the semantics of data packets through meta-headers to all forwarders on a network path. In contrast, if cryptographic mechanisms are imposed on both data and meta-headers, then appropriate processing overheads are unavoidable. However, the additional overhead of meta-header protection will be a negligible fraction of that incurred to protect the data itself.

B. Informing Applications about Lower Layer Services

A key challenge of top-down networking is *how upper layers will know about the service primitives of the layers lower than the one below?* That is, how will layer k know about the existence and variety of service primitives at layers lower than $k-1$. This problem is emphasized when these service primitives are present at intermediate routers or distant points of the network, which is customary in MANETs.

1) Reactive Approach – Meta-Headers in Reverse Direction: A promising approach is to detect lower layer services in an on-demand manner. As connections arise, meta-headers can be used by letting lower layers *rewrite meta-headers in reverse direction* at the destination or at an intermediate router. To convey the available services to the application at a source node, the rewritten meta-headers will travel all the way up in the protocol stack, which can conveniently be done in a

closed-loop manner, as shown in Figure 4. This approach, in essence, uses the vertical communication channel in reverse direction. Such usage of meta-headers can be performed during a signaling phase before data packets are sent in a connection.

However, connectionless or multi-destination services (e.g., anycast, multicast) may not be realizable through this method. For such situations, broadcast-based service discovery protocols can be employed. However, closed-loops are still needed for conveying lower layer services to the application at the source. An interesting and plausible property of meta-header usage in reverse direction is its conformance to the domain ownership. Domain owners will still be able to control what lower layer services are exposed by simply filtering meta-headers at their egress points.

2) Proactive Approach – Pre-informed Designer: A brute-force approach is to let designers of layer k know about the primitives of layers $k-2$ and below; however, this can become complex as the number of lower layer service primitives increase. Indeed, the very basic motivation of the horizontal layered architecture has been to reduce the mapping complexity between various functions in the protocol stack. Further, exposing full semantics of lower layer primitives may not be desirable by ISPs or domain owners due to potential security threats and business conflicts. It is possible to pre-inform higher layer designer with a rank ordering of service primitives rather than the full semantics of them. This approach reduces the amount of mapping complexity from higher layers and should appeal to domain owners more as less internal information will be exposed.

As an example of "rank ordering", consider three routing service primitives at layer 3: Let (i) service A be shortest-path routing, (ii) service B be shortest-path routing with a minimum end-to-end quality guarantee, and (iii) service C be source routing. A is apparently a "baseline" routing service, while B and C are stronger and more flexible, respectively. So, A$<$B and A$<$C show the right rank orders. However it is unclear which one is "higher" in the rank order, B or C? B$>$C is true in terms of *strength* of the service, whereas C$>$B is true in terms of *flexibility* of the service. Further, while B is more valuable for one application, C might be more valuable for another. This illustrates the need for standardization of rank ordering of services. Classification of services has been a common practice in networking, and the standard dimensions are quality (e.g., delay, loss probability), reliability (i.e., variance in quality), and flexibility (i.e., number of options).

IV. Vertical Optimizations

The current network architecture does not allow users to express their value choices at sufficient granularity, e.g., a user willing to pay more for a service critical to itself cannot express this willingness-to-pay to the network. Recent work [28] focused on incorporating users' value choices to the inter-domain routing level. User application's incentives can further be translated into choices at layers lower than 3, e.g., link layer forwarding options, physical layer modulation options. Such vertical optimizations are much needed to find the right tradeoffs in function placement among the application and the lower layers, such that the overall cost of two conflicting goals is minimized: (i) maximally satisfying application's end-to-end requirements and (ii) minimally using resources to implement lower layer services.

A. Top-Down Value/QoS Choices – layers 5, 3, and 2

We now present a *top-down optimization* framework for designing efficient interactions between application, network, and link layers so that protocol optimization based on users' value choices can be done. We define "application-specific costs" for each service being offered by the lower layers. For instance, E_3 represents the price the user application will have to pay for using path services (e.g., shortest-path, max-capacity path, min-delay path) offered by layer 3, and E_2 represents the price for using link services (e.g., priority forwarding, reliable forwarding, best-effort forwarding) offered by layer 2. An application very sensitive to loss (e.g., web browser) might perceive a lossy link as costly and assign a higher E_2 value for that link, whilst another application (e.g., video streaming) might perceive the same link as not costly. We further define a cost budget B to represent the total amount of money the user application is willing to pay for the end-to-end service. The problem of finding the minimum cost network service under *application-specific constraint* B can be formulated as:

$$\min_{P,L}[C_3(P,Q_3,W_3)+C_2(L,Q_2,W_2)] \quad \text{s.t. } E_3+E_2 \le B \tag{1}$$

where C_2 and C_3 represent the cost of implementing a particular layer 2 and 3 service respectively. This "implementation cost" can be in terms of state, messaging, computation or physical resources. Hence, solution to (1) is the set of value choices minimizing the network costs.

Once the value choices from (1) are available, the application may be given the opportunity to select the best options across the network services that maximize its utility. This means that the application and the network will dynamically *re-adjust* themselves to *maximize utility* and *minimize costs* respectively. Similar concurrent optimizations are adopted in some of the existing protocols. For example, the network drops packets to minimize its cost (e.g., buffer space) and the TCP source will reduce its sending rate to maximize its utility (i.e., end-to-end throughput). *Our key innovation here is that the meta-headers will extend this dynamic re-adjustment to all layers and serve as the Lagrange multipliers between the application and the network. Hence, temporal and spatial granularity of the re-adjustment will be much finer.*

In problem (1) above, Q represents knowledge about the network state. The proportion of Q in comparison to the whole network state represents how distributed the problem is, and is a key issue in the stability and consistency of the end-to-end service functions. Also, E *represents the application-specific view of the network*, and it might not be completely available at each node. In practice, nodes will have to solve the problem (1) on-the-fly, maybe every few minutes depending

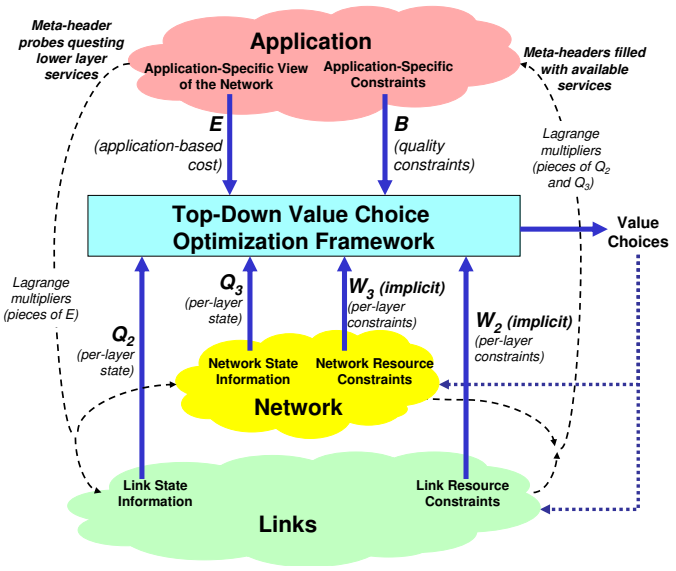

Fig. 5. Top-Down Value/QoS Choice Optimization

on mobility or disruptions. Thus, availability of Q and E at appropriate nodes is crucial. E can be easily conveyed to appropriate nodes related to desired end-to-end function (e.g., to the nodes around a desired path) with our packet meta-headers. Applications at the source and destination nodes can embed their perceived cost of per-layer services into meta-headers, which will then be received by the nodes in between. The form and quantity of Q varies by the network dynamism. For instance, Q for a geographic routing protocol will be the location of relevant points/nodes in the network while Q for a distance-vector routing protocol will be the reachability information. This application- and network-specificity of the problem is the key motivation for our optimization framework.

V. Summary

We presented a top-down networking architecture employing packet meta-headers that can travel vertically across protocol layers. The architecture allows vertical optimizations of protocols and a finer temporal and spatial granularity for the application-network interface. We illustrated how top-down optimizations (e.g. top-down value/QoS choices of protocols with layers 5, 3, and 2) can be done within the architecture. It is possible to make other "top-down" designs within the architecture, e.g. top-down dynamic transport (layers 4, 3, and 2). Such vertical optimizations present a new class of design problems aiming to improve joint performance of multiple layers while respecting the isolation among them.

Acknowledgment

This work is supported in part by the U.S. National Science Foundation awards 0721600 and 0721609.

References

[1] D. Niculescu and B. Nath, "Trajectory based forwarding and its applications," in *Proceedings of ACM MOBICOM*, 2003.
[2] S. Subramanian, S. Shakkottai, and P. Gupta, "Optimal geographic routing for wireless networks with near-arbitrary holes and traffic," in *Proceedings of IEEE INFOCOM*, 2008.
[3] Y. Shi, Y. T. Hou, H. D. Sherali, and S. F. Midkiff, "Cross-layer optimization for routing data traffic in UWB based sensor networks," in *Proceedings of ACM MOBICOM*, 2005.
[4] D. Pompili, T. Melodia, and I. Akyildiz, "Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks," in *Proceedings of ACM MOBICOM*, 2006.
[5] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak, "Minimal and maximal exposure path algorithms for wireless embedded sensor networks," in *Proceedings of ACM SenSys*, November 2003.
[6] L. Popa, A. Rostami, R. Karp, C. Papadimitriou, and I. Stoica, "Balancing the traffic load in wireless networks with curveball routing," in *Proceedings of ACM MOBIHOC*, 2007.
[7] "Google WiFi," "http://wifi.google.com".
[8] "MuniWireless," "http://www.muniwireless.com".
[9] "CUWin - Community Wireless Solutions," "http://cuwin.net".
[10] D. Clark and D. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of ACM SIGCOMM*, 1990.
[11] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Addison-Wesley, 1984.
[12] S. H. Low, L. L. Peterson, and L. Wang, "Understanding TCP Vegas: A duality model," in *Proceedings of ACM SIGMETRICS*, 2001.
[13] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular Content Delivery Using WiFi," in *Proc. of MOBICOM*, 2008.
[14] "Skyhook wireless," http://skyhookwireless.com.
[15] V. Srivastava and M. Motani, "Cross-layer design: A survey and the road ahead," *IEEE Comm. Magazine*, pp. 112–119, Dec 2005.
[16] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Comm.*, pp. 3–11, Feb 2005.
[17] H. S. Lichte, S. Valentin, H. Karl, I. Aad, L. Loyola, and J. Widmer, "Design and evaluation of a routing-informed cooperative MAC protocol for ad hoc networks," in *Proc. of INFOCOM MiniConf*, 2008, pp. 21–25.
[18] L. Larzon, U. Bodin, and O. Schelen, "Hints and notifications," in *Proceedings of IEEE WCNC*, Orlando, FL, March 2002.
[19] Q. Wang and M. A. Abu-Rgheff, "Cross-layer signaling for next-generation wireless systems," in *Proc. of WCNC*, 2003, pp. 1084–1089.
[20] V. T. Raisinghani and S. Iyer, "Cross-layer feedback architecture for mobile device protocol stacks," *IEEE Comm. Mag.*, pp. 85–92, 2006.
[21] Q. Liu, S. Zhou, and G. B. Giannakis, "Cross-layer combining of adaptive modulation and coding with truncated ARQ over wireless links," *IEEE Tran. on Wireless Comm.*, vol. 3, no. 5, pp. 720–724, 2004.
[22] R. Braden, T. Faber, and M. Handley, "From protocol stack to protocol heap – role-based architecture," *ACM CCR*, vol. 33, pp. 17–22, 2003.
[23] S. W. O'Malley and L. L. Peterson, "A dynamic network architecture," *ACM Tran. on Computer Systems*, vol. 10, no. 2, pp. 110–143, 1992.
[24] C. Tschudin, "Flexible protocol stacks," in *Proceedings of ACM SIGCOMM*, 1991, pp. 197–204.
[25] B. Ford and J. Iyengar, "Breaking up the transport logjam," in *Proceedings of ACM HotNets*, 2008.
[26] S. Bhattacharjee, K. Calvert, and E. Zegura, "Active networking and end-to-end arguments," *IEEE Network Magazine*, 1998.
[27] Q. Mahmoud, *Cognitive Networks: Towards Self-Aware Networks*. Wiley-Interscience, September 2007.
[28] X. Yang, D. Clark, and A. Berger, "NIRA: A new inter-domain routing architecture," *IEEE/ACM Tran. on Net.*, vol. 15, pp. 775–788, 2007.
[29] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan, "Declarative routing: Extensible routing with declarative queries," in *Proceedings of ACM SIGCOMM*, 2005.
[30] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353.
[31] N. Milosavljevic, A. Nguyen, Q. Fang, J. Gao, and L. Guibas, "Landmark selection and greedy landmark-descent routing for sensor networks," in *Proceedings of IEEE INFOCOM*, 2007.
[32] M. Perillo and W. Heinzelman, "DAPR: A protocol for wireless sensor networks utilizing an application-based routing cost," in *Proceedings of IEEE WCNC*, June 2004.
[33] C. Gui and P. Mohapatra, "Virtual patrol: A new power conservation design for surveillance using sensor networks," in *Proc. of IPSN*, 2005.
[34] D. D. Clark, "Architecture from the top-down," http://www.nets-find.net/Meetings/S09Meeting/Talks/clark.ppt, 2009.