On Transport Layer Adaptation in Heterogeneous Wireless Data Networks

Aravind Velayutham[1], Hung-Yun Hsieh[2], and Raghupathy Sivakumar[1]

[1] Georgia Institute of Technology, Atlanta, GA 30318, USA,
[2] National Taiwan University, Taiwan, ROC

Abstract. Numerous transport protocols and protocol enhancements (e.g. TCP-ELN, WTCP, STP, etc) have been proposed for optimal performance in different types of wireless networks. In this paper, we define "transport layer adaptation" as the behavior of the transport protocol, with the goal of obtaining best performance, when a mobile host moves across different wireless networks. While defacto assumptions have been made in related work on the ideal characteristics of such transport layer adaptation, no explicit work has been performed in either identifying the nature of adaptation required, or the granularity at which the adaptation should occur. In this paper, we argue that : (i) Transport mechanism changes are how ideal transport adaptation should be performed. Neither transport protocol nor protocol parameter change is sufficient enough for optimal performance across heterogeneous wireless networks. (ii) Transport adaptation has to be performed at a granularity finer than interface handoffs. Ideal transport adaptation should change mechanisms even when the network characteristics change within the same wireless network. We then present the design and implementation of an adaptive transport layer framework called *TP that accommodates fine-grained runtime adaptation of transport mechanisms to achieve the best performance in a given wireless network.

1 Introduction

A tremendous amount of research has been performed in the area of transport protocols for wireless data networks over the last decade or so. It is well established that appropriately designed wireless transport protocols can substantially improve the performance experienced by mobile users [1–3]. Such protocols are designed specifically to address characteristics of the wireless environment they are targeted for. For example, TCP-ELN uses explicit loss notification to aid its congestion control in lossy networks [1]. WTCP [2] uses techniques specifically targeted to address the challenging characteristics of wireless wide-area networks such as low and variable bandwidth and high and variable delay. Similarly, STP addresses the limited reverse path bandwidth problem by aggregating feedback messages from the receiver [3].

Although, the problem of transport layer design for any given wireless environment is well addressed by existing research, the problem of transport layer behavior when a mobile host moves across different types of wireless networks is relatively unexplored. At the same time, the fact that mobile hosts are increasingly equipped with multiple heterogeneous wireless interfaces has elevated the significance of the issue.

In this context, we define *transport layer adaptation* [3] as the behavior of the transport layer protocol, when the mobile host moves across different wireless networks. Appropriately designed transport layer adaptation is critical for achieving the best performance for a mobile host moving across heterogeneous wireless data networks. Toward identifying the ideal transport adaptation behavior required for such mobile hosts, we consider two issues in this work:

1. *What should be the ideal nature of transport adaptation?* Should the adaptation involve changing entire transport protocols at a time, or changing transport mechanisms as required, or simply changing only protocol parameters? We argue that ideal transport adaptation should accommodate changes at the level of transport mechanisms, and that neither transport protocol nor protocol parameter changes are sufficient enough for optimal performance across heterogeneous wireless networks.
2. *At what time granularity will such transport adaptation be required?* Should the adaptation be done only when there is a handoff between network interfaces or will it be required even when network conditions change within the same wireless network? We argue that the transport adaptation has to be performed at a time granularity much finer than that of interface handoffs. Essentially, ideal transport adaptation should change mechanisms even for change in network characteristics within the same wireless network.

We then design and implement a runtime adaptive transport layer framework called *TP[4]. *TP is a transport layer solution that accommodates the ideal nature and granularity of transport adaptation. It allows for the reconfiguration of transport layer behavior, while minimizing the impact of such transformations on applications, and hiding it completely in the best case. Briefly, *TP provides a clear separation in the realization of core and non-core transport functionalities, is fully modular, employs an event-driven execution model, and allows for effective state propagation between different avatars of the transport protocol as it transforms.

The rest of the paper is organized as follows : In Section 2, we present in more detail the transport adaptation issues addressed by this work. In Section 3, we present the design and implementation of the adaptive transport layer framework called *TP. We present case studies to evaluate the performance of the proposed *TP transport layer framework in Section 4 and conclude the paper in Section 5.

2 On Transport Adaptation

In this section, we discuss the two key issues with respect to achieving ideal transport adaptation. The arguments presented serve as the basis for the design and implementation of the *TP transport layer framework presented later in the paper.

[3] For purposes of brevity, we refer to transport layer adaptation as simply transport adaptation in the rest of the paper

[4] The "*" in *TP represents a "wild-card" that can take the form of any desired transport protocol solution.

2.1 What is the required nature of ideal transport adaptation?

A transport protocol can be viewed at different levels of complexity and these levels dictate the choices available for transport layer adaptation. The coarsest level of detail is the entire transport protocol consisting of all the mechanisms used to implement the different functionalities. Change at the protocol level would require the replacement of one transport protocol (say TCP-ELN) by another (say WTCP) for optimal performance. At a finer granularity than protocol change, is change of one or more mechanisms used by transport protocols. Examples of transport mechanisms include loss-based congestion detection, rate-based congestion control, self-clocked data transmission and timeout-based loss recovery. We note that this level of adaptation is the most complex because it involves the co-existence of multiple previously unrelated modules. The finest level of detail of transport protocols are the parameter values used by the transport mechanisms. Protocol parameters include increase and decrease values in AIMD congestion control mechanism, the number of SACK blocks used by the SACK acknowledgment scheme, etc. Although protocol parameter change is the finest level of detail within the transport protocol, it is also the easiest level of adaptation wherein the values of variables are changed within the same transport protocol.

We argue that transport mechanism change is the ideal nature of transport adaptation for efficient performance across heterogeneous wireless networks. Varying network characteristics directly impacts the performance of transport mechanisms. For instance, a high wireless loss rate adversely impacts any loss-based congestion detection mechanism. Although the loss-based congestion detection mechanism used by say TCP-ELN is affected adversely under high loss conditions, other transport mechanisms of TCP-ELN such as window-based congestion control might not be affected. Thus change at the granularity of transport protocols is not sufficient for ideal transport adaptation. Protocol parameter change is also not sufficient for achieving optimal performance under all network environments. For instance, under high loss conditions, merely increasing the number of SACK blocks (a protocol parameter) would not achieve efficient congestion detection as long as loss is used to detect congestion. In this case, an alternative mechanism to loss-based congestion detection such as delay-based congestion detection should be used[5]. Thus, any transport adaptation framework should have the ability to change transport mechanisms constituting transport protocols. We incorporate this observation as a design element in the *TP adaptive transport layer framework.

2.2 What is the ideal time granularity for transport layer adaptation?

The coarsest level of time granularity for transport layer adaptation is across transport layer sessions. This level of adaptation is triggered by application requirements rather than wireless networks. Another level of transport adaptation granularity is change of elements when there is a *vertical handoff* from one wireless network to another. In [5], the authors define a vertical handoff as the shift from one wireless network to another. Another alternative level of granularity for transport layer adaptation is when there is

[5] In [4], we present a comprehensive set of simulations based performance results that support the above arguments.

"significant" change of network characteristics within the same wireless network. By "significant" change, we refer to change in network characteristics which would lead to degradation of the performance of the transport protocol currently being used. These can happen due to several reasons including "horizontal handoffs" from one access point (or base station) to another possibly overloaded access point, traveling through a low-signal area like a tunnel, etc.

Any wireless network can be characterized in terms of its bandwidth, loss rate and delay properties. Each transport mechanism has an optimal operating region in this network characteristics space governed by the specific operations of the mechanism. For example, loss-based congestion detection is not efficient under high wireless loss conditions. At the same time, it can be shown that wireless loss rates can vary significantly even within the same wireless network such as WLANs depending on the location of the communicating entities. Similarly, depending on the amount of multiplexing performed at the gateway of a WWAN cell, the delay jitter characteristics of the network can change within the WWAN. Thus we argue that, given the fact that transport mechanisms are effective in specific network conditions and that network conditions can vary even within the same wireless network, transport layer adaptation should be performed when network characteristics change significantly even within the same wireless network[6].

3 *TP : A Unified Transport Layer Framework

In this section, we present the design elements of *TP, an adaptive transport layer framework, that can accommodate the mechanisms used in different transport protocols, and dynamically transform itself to exhibit the behavior of the transport protocol best suited for a given environment. The principal focus of the adaptive framework is the ability to accommodate multiple alternative transport mechanisms that will be absorbed into the framework from different transport layer solutions. Hence, we primarily focus on how the framework supports such dynamic reconfigurability.

3.1 Design Goals

The design goals of the *TP framework reflect the solutions to the transport adaptation problems we have identified. The goals include:

Reconfigurability : A key design goal of *TP is the ability to reconfigure itself to use the transport layer mechanisms best suited for a given environment. The reconfiguration of mechanisms is triggered by changes in network characteristics such as the increase in loss rates and delay jitter, or simply interface handoffs. Unlike other configurable frameworks and protocols proposed in related work [12], *TP is designed to support run-time reconfigurability with minimal application intervention. Note that since most

[6] Interested readers are referred to [4] for performance results substantiating the above arguments.

of the changes in network characteristics do not require the awareness of the application, it is desirable to design a transport layer framework that can *seamlessly* perform reconfiguration with minimal disruptions to the application.

Extensibility : *TP is designed as a generic framework that can accommodate various mechanisms used in different transport layer protocols. Therefore, *TP by nature is an extensible framework that can "plug-in" any new or existing transport mechanisms. The performance of *TP is not limited to any specific transport protocol or mechanism. Instead, whenever a better mechanism tailored to the characteristics of a given environment becomes available, *TP can use these protocols for achieving higher baseline performance. The design of *TP ensures that whenever the network characteristics become favorable to any module already registered, it will be invoked to perform the corresponding functionality. Toward this goal, *TP defines a set of interface functions that facilitate new protocols to be incorporated into the *TP framework.

Minimal Overheads : While *TP allows flexible reconfigurability and extensibility of transport mechanisms, it does not trade overheads for the ability of fine-grained transport adaptation. *TP is designed to incur minimal overheads compared to a static transport protocol (e.g. WTCP and STP). The overheads that need to be minimized in a dynamic protocol like *TP include: (i) Complexity: The execution efficiency of *TP should not be sacrificed simply because modules are dynamically composed. In other words, *TP should minimize the computation overheads, such that any host can support as many connections using *TP as connections using a static transport protocol. (ii) Redundancy: The redundancy due to repetitive implementations of any functionality in different modules should be minimized. In other words, the memory footprint of *TP should be kept at a minimum. (iii) Latency: The latency incurred during reconfiguration of mechanisms can cause interruptions or disruptions at the application layer. Since it is possible that reconfiguration occurs several times in a connection, such a "reconfiguration" latency should be kept to a minimum.

3.2 Design Elements :

We now present the key design elements in *TP that allow it to meet the design goals described earlier.

Triggers : The reconfiguration of mechanisms in *TP is triggered by changes in one or multiple network parameters pertinent to the transport mechanism in consideration. The transport functionalities have dominating parameters associated with them which influence which strategy to use in specific network conditions. Each potential module to be used by *TP first specifies the network parameters that need to be monitored, as well as the conditions (e.g. threshold values) for triggering the reconfiguration. *TP is responsible for initiating the reconfiguration when the network conditions are met. Whenever a decision is made to swap in a module, the corresponding module is loaded into *TP, replacing the current module in use. The monitoring of the triggers and related parameters is again performed by *TP.

Separation of Core and non-Core Modules : As we mentioned earlier, the *TP framework is invariant and independent of the specific protocols used, while individual mechanisms may be swapped in and out depending on the network characteristics. *TP adopts a structured separation of permanent *core* and configurable *non-core* modules. The core is the the *TP framework itself, and the non-core can be considered as different transport mechanisms that may change across different operating conditions. Since the core does not change, it can be optimized for achieving higher execution efficiency and minimizing overheads. The non-core modules on the other hand consist of all the transport mechanisms implementing various transport functionalities.

Modular Architecture and Execution Model : *TP uses a modular architecture for incorporating the non-core modules.[7] Since the non-core transport mechanisms are those that change across different network environments, the modular design of the non-core mechanisms allows for *fast swapping of modules* in and out of the kernel, and *fine-grained adaptation* of the transport protocol mechanisms. Together with an event-driven execution model, the modular architecture facilitates ease of reconfigurability. In *TP, the core maintains an event queue that is served through the invocation of non-core modules. When modules are invoked, they may in turn register further events in the event queue, thus ensuring the execution of the proper set of mechanisms in the appropriate order. Specifically, the event-driven execution greatly simplifies the reconfiguration process in *TP as it merely involves replacement of appropriate event-handlers.

State Propagation : *TP allows the inheritance of transport layer state from one non-core module to another when a reconfiguration is performed. Examples of states that can be inherited across modules include the data buffer, the SACK scoreboard for reliability, and the advertised window size of the receiver, etc. *TP enables such state propagation by allowing non-core modules to maintain both *public*, and *private* state. Any public state is maintained by the core, while the private state is maintained by the non-core module. Thus, when a non-core module is swapped out, and a new non-core module is swapped in, the new module has access to the public state left behind by the old module.

Mobile-host Centric Operations : *TP allows dynamic switching of different protocols on the fly, and extending the protocol stack when newer protocols are developed. While it is reasonable to assume that the mobile host will need to accommodate these protocols for achieving the best performance in different wireless environments, it may not be the case for the static Internet host. This is because static hosts in such a scenario will have to accommodate *all possible* transport protocols in anticipation of communication from any mobile host in the Internet – which is clearly infeasible given that there can be a multitude of protocols corresponding to the large number of heterogeneous wireless access technologies. Therefore, a setting adopted by *TP is to make the mobile-host the primary seat of transport layer intelligence, irrespective of whether it is acting as a sender or a receiver. In such a setting, any change runtime or otherwise can be performed solely at the mobile host without any intervention at the static host.

[7] Note that the *TP core is a static component, and hence does not need to be modular.

3.3 Software Architecture

*TP is a mobile-centric framework with the mobile host being the primary control for the protocol operation. The static host in *TP is very simple. When the static host acts as the receiver, it simply sends feedback information used by the mobile host (sender) such as ACK and SACK information. When the static host acts as the sender, on the other hand, it merely responds to requests from the mobile host for sending data. In other words, the static host plays a passive role responding only to instructions sent by the mobile host. The reconfiguration occurs at the mobile host depending on the network environment.

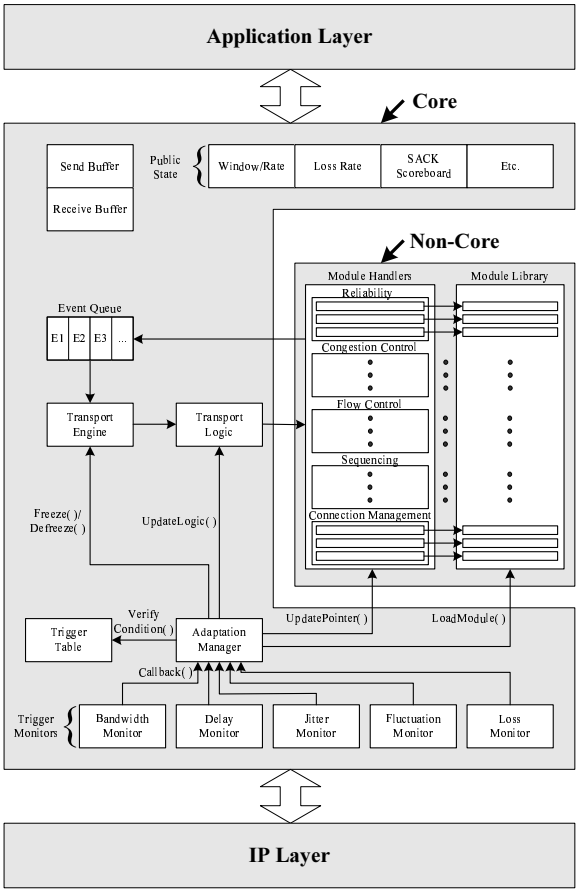


Fig. 1. *TP Framework

Figure 1 shows a high level architectural diagram of *TP at the mobile host. We refer to it as *TP for purposes of presentation, and qualify it only when the reference

is to the static host. As shown in the figure, the *TP functionality is separated into the fixed *core* and reconfigurable *non-core*.

The core consists of the following components:

Interfaces with the Application and IP : The core provides a fixed interface for the application layer and the IP layer to communicate with *TP. Any communication coming in from the application including data and control (say, socket options, connection open and close) is handled by the core. Similarly, any communication coming in from the IP layer is handled by the core.

Global Data Structures : The data maintained by the core includes the send and receive buffers, the public state, and the event queue. The handling of the buffers by the core is clear since the reconfiguration of transport modules should not affect the data in the buffer. The public state is used for *state inheritance* and serves as a shared space for non-core modules to communicate with each other. Finally, the event queue is related to *TP's execution model.

Transport Engine : The core in *TP supports the backbone of the transport layer framework, and hence any intelligence that pertains to the transport protocol operations (which change under different environments) is provided by the non-core modules. Note that in *TP, not only the transport modules can be reconfigured, but the *logic* (e.g. sequence of execution) for the execution of these non-core modules (in response to transport layer events) is also reconfigurable. Hence a transport protocol developed in the *TP framework not only can use *TP to incorporate new transport modules, but also can decide how and in what form the modules are used. This is facilitated by allowing the transport protocol developed to provide a *transport logic*. The logic is loaded along with the non-core modules, but it is the core's transport engine that *executes* the transport logic. The transport engine, depending on events registered in the event queue, uses the transport logic to execute the appropriate non-core modules.

Reconfiguration Entities : There are three components in the core that are related to the reconfiguration initiation process: the trigger table, trigger monitors, and the adaptation manager. Non-core modules register the trigger and the condition for module invocation with the core. The trigger is a logical combination of network parameters monitored by various trigger monitors. The adaptation manager receives callbacks from the trigger monitors when the conditions specified by non-core modules are met. The adaptation manager uses the trigger table to identify which modules and logic to use, and loads the corresponding modules from the module library into the non-core.

The non-core modules reflect the traditional transport layer intelligence, including reliability, congestion control, and flow control. The specific logic used to combine the modules together for achieving a transport layer functionality is part of the *transport logic*. The transport logic in *TP is in fact an event/handler table that maps the events registered with the engine and the available non-core modules. Thus, it acts as the liaison between input events to the transport framework, and the actual transport functionality. Moreover, it also acts as the enabler for communication between the non-core

modules, through the generation and servicing of internal events. Non-core modules communicate with each other solely through the generation and servicing of events, and the public state maintained by the core. The non-core module can read and write from/into the public state. Non-core modules can interface with the core through registering events in the event queue, and data exchange in the send/receive buffer.

4 Case Studies

In this section, we present case studies to evaluate the performance of the *TP framework in comparison with other transport protocols. The specific transport protocols we compare against are TCP-ELN [1], WTCP [2], STP [3]. We use these protocols as representative tailored protocols for their respective target environments of wide-area, satellite, and local-area wireless networks respectively. The case-studies help in understanding the benefits obtained by using an adaptive transport framework solution, which can perform runtime re-reconfigurability and modular composition of transport mechanisms.

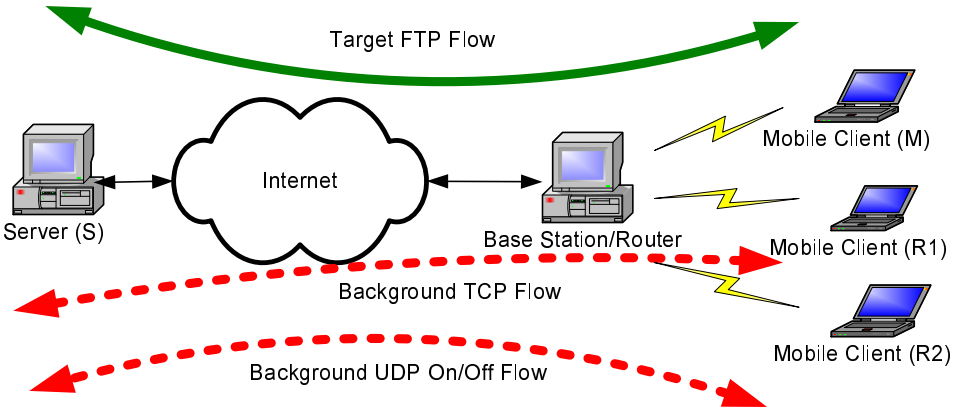


Fig. 2. Network Topology

4.1 Experimental Network Topology

Our experimental test-bed consists of Dell Inspiron laptops and Pentium-based personal computers. The static machines are interconnected using 10Mbps WAN connection and the mobile hosts are connected to the base station (access point) using an IEEE 802.11b wireless connection with a raw signaling bandwidth of 2Mbps. The network topology is depicted in Figure 2. We have implemented the three protocols, TCP-ELN, WTCP and STP in the Linux 2.4 kernel. Implementation details about the mechanisms used by each of the protocols can be found in [4].

4.2 Network environment parameters

Any network environment can be captured by the bandwidth, packet loss and delay of the network. We use the mean (average) bandwidth, packet loss and delay as well as the variance of the bandwidth and delay to capture the specific wireless environment. Different wireless networks have different values for the above parameters and we analyze the performance of the different transport mechanisms across varying values of the above-mentioned parameters. We use the values given in Table 1 for the different wireless data networks. The data rate is the average bandwidth that the wireless link supports; the fluctuation period is the frequency at which the bandwidth varies (the magnitude of variation is 30% of the mean bandwidth of the link) - the fluctuation period is represented as the percentage of the delay of the link; the packet loss rate is the average drop rate of the uniform loss module. The magnitude of jitter is represented as a percentage of the average one-way delay across the wireless link.

Table 1. Wireless network characteristics

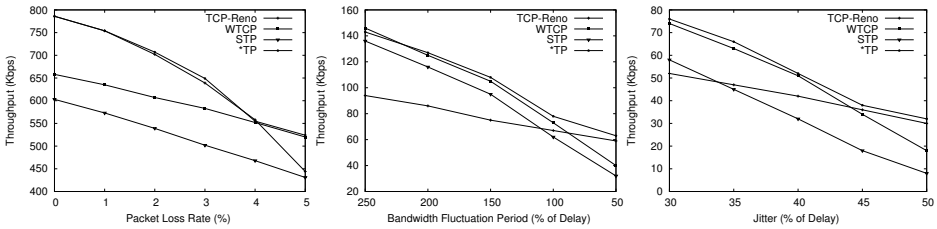
	Data rate (Kbps) (Fluctuation Period)	Average Packet Loss Rate (%)	Delay (ms) (Jitter)
WLAN	2000 (500%)	1	50 (10%)
WWAN	300 (250%)	5	250 (20%)
Satellite network	100 (250%)	5	1000 (30%)

4.3 Lossy WLANs

We know that in WLANs, as the loss rate increases, loss-based congestion detection mechanism degrades in performance. This is due to the inability of the mechanism to detect congestion because of insufficient information about the network at high packet loss rates. We use loss-rate as the trigger parameter for the loss-based congestion detection mechanism and the threshold value is set to 2%. When the *TP trigger module detects that the loss rate increases beyond 2% then it switches the loss-based congestion detection to a delay-based congestion detection mechanism which does not suffer much in lossy environments. We can see from Figure 3(a) that *TP performs well even as the loss rate increases within the WLAN environment. We can see that the slope of the curve corresponding to the throughput of *TP follows the best set of mechanisms for the given conditions. This is because of the capability of *TP to choose and use the best available mechanisms at run-time.

4.4 Bandwidth Fluctuation in WWANs

The tuned-rate acknowledgment scheme, such as the one used by WTCP, that performs well in WWAN conditions suffers when the bandwidth fluctuation increases. Since bandwidth fluctuations can be considered to be the norm in both CDPD and higher generation wireless wide area networks, an ideal transport adaptation solution should be



(a)WLAN (b)WWAN (c)Satellite network

Fig. 3. Performance study of the *TP framework

able to change the acknowledgment scheme when the bandwidth fluctuation increases beyond a certain threshold value. *TP has the capability to accommodate both the self-clocked (such as the one used in TCP) and tuned-rate acknowledgment schemes and can swap between them depending on the operating conditions. We use bandwidth fluctuation as the trigger for the acknowledgment scheme and the threshold value to be 100% of the delay in the network. We observe from the results in Figure 3(b) that *TP achieves the best performance by swapping the acknowledgment mechanism used when the network conditions degrade. As noted earlier, *TP achieves the best throughput for a given set of transport mechanisms because of its ability to change mechanisms in an intelligent fashion using triggers.

4.5 Jitter in Satellite networks

Both delay-based and inter-packet separation-based congestion detection mechanisms suffer under high delay variations. Although the delay-based scheme performs well in high-loss satellite environments, it has to be replaced by the loss-based scheme when the jitter increases beyond a threshold. *TP achieves precisely this functionality by swapping the congestion detection mechanisms if the trigger, namely jitter, value is beyond a threshold value. We see from the result in Figure 3(c), that by performing such fine-grained adaptability *TP is able to achieve best performance even in varying network conditions. Here we can see that even as the performance degrades due to the original congestion detection mechanism suffering at high jitter, *TP is able to adapt its behavior to use a different mechanism better-suited for the high jitter conditions.

5 SUMMARY

In this paper, we consider the problem of transport adaptation in heterogeneous wireless data networks. Specifically, we answer the questions relating to the ideal nature and granularity of transport adaptation. We argue that an ideal adaptation solution should be able to change mechanisms at a granularity finer than normal interface handoffs. The change in mechanisms should happen even as the network characteristics change within a single wireless network. We design and implement a runtime adaptive transport layer framework, called *TP, that accommodates the requirements of adaptation determined by the performance evaluation.

Acknowledgment

This work was supported in part by a Motorola UPR grant, and National Science Foundation grants ECS-0225497, CCR-0313005, ANI-0117840, and ECS-0428329.

References

1. H. Balakrishnan and R. Katz, "Explicit loss notification and wireless web performance," in *Proceedings of IEEE GLOBECOM*, Nov. 1998.
2. P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.
3. T. Henderson and R. Katz, "Trasnport protocols for internet-compatible satellite networks," *IEEE Journal on Selected Areas in Communications(JSAC)*, vol. 17, no. 2, pp. 345–359, Feb. 1999.
4. A. Velayutham, H.-Y. Hsieh, and R. Sivakumar, "*TP : An adaptive transport layer framework for multi-homed mobile hosts," in *GNAN Research Group Technical Report*, 2005.
5. M. Stemm and R. Katz, "Vertical handoffs in wireless overlay networks," *Mobile Networks and Applications*, vol. 3, no. 4, pp. 335–350, 1998.
6. C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, and R. Wang, "Tcp westwood: End-to-end congestion control for wired/wireless networks," *Wireless Networks Journal*, vol. 8, pp. 467–479, 2002.
7. I. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A new congestion control scheme for satellite IP networks," *ACM/IEEE Transactions on Networking*, vol. 9, pp. 307–321, June 2001.
8. C. P. Fu and S. C. Liew, "Tcp veno: Tcp enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 5, pp. 802–817, Feb. 2003.
9. M. Handley, S. Floyd, J. Pahdye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000.
10. L. Brakmo and L. Peterson, "Tcp vegas: End to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.
11. R.Rejaie, M.Handley, and D.Estrin, "Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *INFOCOM*, Mar. 1999.
12. G. Wong, M. Hiltunen, and R. Schlichting, "A configurable and extensible transport protocol," in *INFOCOM*, Apr. 2001, pp. 319–328.