On the benefits of Cross Layer Feedback in Multi-hop Wireless Networks

Harkirat Singh
Samsung Electronics, 75 W Plumeria Dr., San Jose, CA 95134
Email: har.singh@samsung.com

***Abstract*—Wireless networks operate under harsh and time-varying channel conditions. In wireless networks the time varying channel conditions lead to variable SINR and high BER. The wireless channel is distinct from and more unpredictable than the far more reliable wireline channel. *Cross layer feedback* is a mechanism where layers provide *selective* information to other layers to boost the performance of wireless networks. *Cross layer feedback* can lead to a tremendous increase in the performance of the TCP/IP stack in wireless networks, and an increase in the user's satisfaction level. However, it is possible that naive feedbacks (or optimizations) can work non-coherently; therefore, these can negatively effect the performance of the TCP/IP stack. In this paper, we holistically analyze each layer of the TCP/IP stack, and propose possible Cross layer feedbacks which work coherently. The proposed Cross layer feedbacks can greatly enhance the performance of the TCP/IP stack in wireless networks.**

I. Introduction

Over the last decade we have observed a phenomenal growth of the Internet both in terms of size and use. To a large extent, this dramatic growth can be attributed to the well designed and layered TCP/IP stack. In such a modular design, the system is composed of different components or layers. These layers can be implemented separately without sharing too much information with other layers. A layer does not need to know how exactly other layers are implemented; it only needs to know what operations other layers perform. Therefore, layers provide non-redundant functions and operate in a cohesive manner. For example, a commonly used link layer design methodology in wireline networks leaves the link layer design as simple as possible and introduces maximum complexity near the application layer [25]. This "end-to-end argument" says that appropriate error-control can be invoked at the network end-points, since these end-points have maximum information about the application [31].

Furthermore, the modular design provides developers flexibility to carry out independent implementation and still ensures seamless integration with minimal effort. In a modular layered system, each layer can be independently modified; therefore, it provides easy extensibility and maintainability. However, there are a few drawbacks to such modular layered architectures. For example, in the TCP/IP stack (a modular layer architecture), the layers are designed or optimized to function in the worst conditions, and such techniques do not allow layers to adapt to changing and improved network conditions [35].

While the standard TCP/IP stack results in optimum performance when used with wired links, the same stack achieves non-optimum performance when used over wireless links [7], [20], [4]. So the natural question is why the TCP/IP stack does not perform optimally over wireless links. This behavior can be attributed to the unique wireless link properties in contrast with wired links. In general, wireless links have: (1) lower bandwidths, (2) higher latency, (3) higher outage probability, and, (4) higher channel fading, which results in higher bit error probability [4]. Typically, wireless links have an order of magnitude higher packet error probability in comparison to a wired link. These adverse wireless link properties are detrimental to the performance of the TCP/IP stack in wireless networks because TCP's design is not optimized for the channel conditions that are endemic to wireless networks. Thus, TCP misinterprets the packet loss due to error as congestion and invokes congestion control, which is incorrect.

Cross layer feedback is a mechanism where layers provide *selective* information to other layers to boost the performance of wireless networks. Consider the wireless node where the link layer has two different active streams. The first stream has a reliable-data transfer, and the second stream has a real-time data flow, for example, voice. In the first stream, lost packets need to be retransmitted end-to-end, since the foremost requirement of this stream is to have reliable data transfer. However, the second stream can withstand a few packet drops, since the primary requirement is that the end-to-end latency does not exceed the user's dissatisfaction limit. Therefore, we need a relatively *strong* link error-recovery scheme for the reliable data

Most of this work was done while the author was at the Portland State University, Department of Compute Science.

transfer stream, and, a relatively *weak* link error-recovery scheme for the real-time data stream. [25] presents "flow-adaptive link-recovery", a novel error-recovery algorithm, where the link layer has a *knob* to tune the error-recovery mechanism. The link layer "learns" from the network layer about the QoS requirement and appropriately selects a link error-recovery mechanism. This is an example of *cross layer feedback*, where the link layer learns from the network layer to optimize the performance. However, this design necessitates some degree of violation of strict layering principles of the traditional layered architecture. An absolute orthogonal approach to the strict layered approach would be to merge all the layers. This system will have better performance, but it cannot be used in other settings. Further, this *single* layer approach will not be able to evolve in the long run.

Cross layer feedback design can lead to a tremendous increase in the performance of the TCP/IP stack in wireless networks, and an increase in the user's satisfaction level. While designing a cross layer feedback, it is paramount for the designers to analyze whether the cross layer feedback breaks any semantics of the stack. It is possible that naive feedbacks (or optimizations) can work non-coherently; therefore, these can negatively effect the performance of the TCP/IP stack.

In the next section we highlight the benefits of modular architectures and provide a few examples of them. Section III describes the key properties of wireless communication and how they affect system performance. In Section IV, we present the formal definition of *Cross Layer Feedback*. Section V describes how cross layer feedback can be applied to the different layers of a TCP/IP stack in the context of wireless networks. Finally, Section VI concludes the paper.

II. MODULAR DESIGN AND BENEFITS

In a modular design, the system is composed of different components or layers. The layers can be implemented separately without sharing too much information from other components. The design also ensures seamless integration with minimal effort. The following are the benefits of a modular design:

- Fast independent development of the layers with minimal coordination among them.
- Easy debugging and system verification.
- Flexibility to replace any component without affecting the performance of the rest of the system.
- Easy maintainability and future extensibility.

A. TCP/IP Stack

The TCP/IP stack is, by any measure, the most successful example of a modular architecture. The traditional TCP/IP stack comprises four layers (Figure 1). Each layer provides a set of primitives to the higher layer, and *Protocol* defines the implementation of these primitives. Adjacent layers communicate via predefined primitives. Each layer is responsible for providing a specific functionality. For example, the *link* layer breaks input data into frames (a set of bits), and each frame is padded with redundant bits to provide error correction and detection. The link layer is responsible for exchanging frames between a sender and a receiver in the same segment. The *network* layer handles the delivery of packets across the network, as well as providing the abstraction of a path. The *transport* layer provides end-to-end delivery of application data between a sender and a receiver. This layer uses the path or route primitive provided by the *network* layer. Similarly, the *network* layer uses the primitives from the *link* layer.

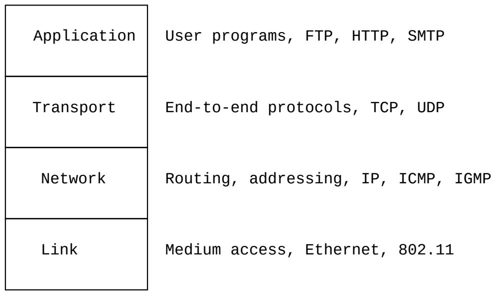
Application
User programs, FTP, HTTP, SMTP
Transport
End-to-end protocols, TCP, UDP
Network
Routing, addressing, IP, ICMP, IGMP
Link
Medium access, Ethernet, 802.11

Fig. 1. TCP/IP protocol stack [33].

Each layer can have more than one Protocol, which defines the set of primitives provided by that layer. For example, the *transport* layer can have UDP or TCP. Moreover, the modular design gives software developers the flexibility to develop new protocols and upgrade existing protocols with minimal coordination with the protocols at other layers. The *link* layer is the lowest layer in the stack; therefore, it communicates with the physical layer hardware whose function is to transmit raw bits over a communication channel. For each type of physical layer hardware, there is a corresponding link layer protocol, which normally includes the device driver in the operating system. The modular design provides flexibility to use different hardware, for example, IEEE 802.3 (Ethernet), or IEEE 802.11 (WLAN), and still use the same TCP/IP stack.

B. TinyOS

TinyOS is an event-driven, microthreaded operating system designed for sensor network nodes that have limited resources; for example, 8 Kbytes of program memory, 512 bytes of RAM [22]. TinyOS is a component based architecture; it provides a set of reusable system components. A component consists of four interrelated parts:

a set of command handlers, a set of event handlers, an encapsulated fixed-size frame, and tasks. Tasks, commands, and handlers execute in the context of the frame and operate on its state. To facilitate modularity, each component declares the commands it uses and events it signals. These declarations are used to compose modular components specific to an application's requirement. The components create an abstraction of a layered modular architecture, wherein adjacent components communicate through events and commands. The modular design helps in reducing the application development and debugging cycle.

III. Wireless Networks

Wireless networks operate under harsh and time-varying channel conditions. The wireless channel is distinct from and more unpredictable than the far more reliable wireline channel. A wireless node wishing to transmit a packet simply radiates *Radio Frequency* (RF) energy over the air and the receiver receives this RF energy in the presence of other radiation (or transmissions) and background- and thermal-noise. The non-negligible probability of interference (RF energy radiations) from other nodes and environment noise make wireless transmission susceptible to error. Erroneous packets are dropped at the receiver's link layer. Unlike wireline networks where a Bridge/Switch can shutdown (or block) its interfaces to receive only one packet, in wireless networks we do not have this notion of selective reception.

A. *Wireless Link Model*

Let $\mathcal{S}$ be the set of transmitting nodes; each node $i \in \mathcal{S}$ is transmitting with power $P_t(i)$, G denotes the path gain (inverse of path loss) between pairs of nodes, $G = \{G_{ij}\}$ is an $n \times n$ matrix with $G_{ii} = 0, \forall i$, η_{noise} defines the sum of background- and thermal-noise. We are interested in determining Signal-to-interference-and-noise-ratio (SINR) at node $b \notin \mathcal{S}$ from the transmitter $a \in \mathcal{S}$, in the presence of i transmitting nodes. We can write the following expression for SINR at b,

$$\Gamma(b) = \frac{P_t(a)\ G_{ab}}{\sum_{\forall i \in \mathcal{S}, i \neq a} P_t(i)\ G_{ib} + \eta_{noise}} \tag{1}$$

b will correctly decode the signal if the received signal-to-interference-and-noise-ratio is above SINR threshold Γ_{Th}, otherwise the signal is lost. Γ_{Th} is a pre-specified SINR threshold, it is a factor of: acceptable bit-error-rate (BER), channel coding/decoding rate, and, modulation/demodulation scheme. However, instantaneous SINR $\Gamma(b)$ is a factor of: path gain (inverse of path loss), transmit power, antenna gain, and multiuser interference. The denominator of the above expression denotes the sum of interference and noise. The interference from other nodes is known as co-channel interference (CCI). Both CCI and noise reduce SINR at the receiver.

In the above scenario, Γ_b, the SINR at b can be increased by either increasing the numerator or by decreasing the denominator. If a increases the transmit power $P_t(a)$ to boost the SINR at b, it will also increase the interference at other nodes. Therefore, it is a non-trivial task to use variable transmit power at different nodes. In other words, CCI from other nodes cannot be combatted by simply increasing the transmit power. The SINR can also be increased by reducing the denominator. We cannot do much with the thermal noise which is hardware dependent; however, we can use methods to reduce the noise from other nodes i.e. to mitigate the effect of CCI and increase SINR. We will discuss such techniques later in this paper.

A wireless channel is non-static; it often varies with time. For example, the wireless channel can *fade*, which affects the channel capacity. The amount of attenuation in the signal strength due to fading depends on environmental obstacles such as the presence of hills, trees, and high-rise buildings. SINR can vary due to the following time varying channel properties:

- **Noise:** background and thermal.
- **Multipath Fading:** which is caused by the multiple paths the transmitted signal takes to the receiver. The signals from multiple paths add with different phases, resulting in a reduced amplitude. In the worst case, when multipath signals are out of phase a significant reduction in the signal strength occurs, [23].
- **Delay Spread:** which is caused by the difference in the propagation delays among the multipath signals. This results in multiple symbols arriving at the same time, and therefore, high bit error probability, [23]. This phenomenon is called intersymbol interference (ISI).
- **Path Loss:** loss in received power due to diffusion and shadowing.
- **CO-channel interference (CCI):** which is caused by multiple access and co-channel transmissions.
- **Handoff:** due to mobility and multiple access.

[32] presents experimental results of a IEEE 802.11 network; the results show a first-order relation between SINR and higher-layer network performance parameter such as throughput and delay. We use OPNET to simulate a simple wireless network comprising one sender and one receiver for the IEEE 802.11a MAC model. We plot the system throughput as a function of SNR in Figure 2. We observe that the throughput continues to increase with the increase in the SINR, however, after a certain point the throughput saturates; this is the maximum channel capacity. Similar to [32], we observe a clear relation between the system throughput and SINR.

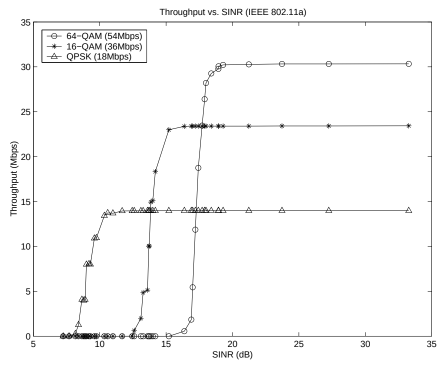

Fig. 2. Throughput vs. SINR (IEEE 802.11a).

In wireless networks the time varying channel conditions lead to variable SINR and high BER. From the link layer's point of view, high BER results in high data loss and large latency. On the other hand, in wireline networks the main causes of data loss are congestion and buffer overflow. In wireline networks packets are seldom discarded due to bit errors. Therefore, wireline networks attain relatively stable throughput and low link layer losses. Furthermore, in wireline networks it is a trivial task to add additional bandwidth. Unfortunately, it is not a simple task to increase the bandwidth of a wireless network. In the United States, the Federal Communications Commission (FCC) allocates the bandwidth and it cannot be changed. The following are a few examples of the negative impact of fixed low-bandwidth and high data loss on the performance of the TCP/IP stack in wireless networks:

- **TCP over wireless links:** The Transport Control Protocol (TCP) is a connection oriented byte stream end-to-end transport layer protocol. TCP is designed to provide protection against packet loss due to error and congestion. TCP's design is optimized for wireline links. A wireless link has high packet loss due to fading, mobility, multipath, and hand-offs, and the only protection TCP provides against packet loss is congestion control i.e. to throttle the flow of the packets into the network. Therefore, in wireless networks TCP misinterprets packet loss as congestion resulting in poor throughput.
- **Minimum hop count routing metric over wireless links:** In traditional wireline networks, routing protocols typically use minimum hop count as the routing metric to find paths. In wireless networks, since SNR is inversely proportional to the hop length, using minimum hop count will select long range links. This is a problem, since these types of links are more susceptible to changing channel conditions. Therefore, in wireless networks the shortest path routing metric will discover better paths but they will have a large average hop length.
- **Real-time traffic over wireless links:** In a real-time flow, such as voice, the time of packet arrival at the receiver is highly pertinent – if a voice packet misses a pre-defined deadline, it may be of no use to the listener [4]. It is better to accept erroneous voice packets or to even drop them and replace them with silence [31]. Since a *strong* retransmission mechanism at the link layer will block the delivery of all the subsequent voice packets following a loss, we need a content based error recovery mechanism.

The above examples show that the design of the traditional TCP/IP stack does not take into account the dynamic nature of wireless links. Furthermore, it limits the interactions among various layers. Therefore, the traditional layered stack performs poorly over wireless links.

IV. Cross Layer Feedback

The discussion above articulates the benefits of a modular design while simultaneously highlighting the drawbacks of using this design in wireless networks. There are two obvious approaches to design the Protocols for wireless networks - use a monolithic design philosophy where the code is highly optimized for that environment, or use a modular approach with adaptability built in using *Cross Layer Feedback*. The monolithic design idea has little merit because of its inability to evolve. In Cross layer design, there is limited inter-layer *feedback* (or interaction), which allows adaptation. A Cross Layered system improves the performance of the TCP/IP stack at the cost of limited signaling. For example, limited feedback from the application– or the network-layer to the link layer about the type of traffic can be useful in deciding whether a link layer error recovery should be invoked. Since real-time flows are delay sensitive and link layer error recovery can significantly increase the delay for the flows end-to-end recovery may be preferred. Wireless channels *fade* due the reasons presented in Section III; during fade periods multiple packets are dropped because of low SNR or high BER. If we enable a cross layer feedback from the physical layer to the transport layer about the channel quality such as SNR, TCP, the transport layer protocol can *freeze* its state during the periods the SNR is below minimum SNR threshold [13]. This avoids multiple packet retransmissions which also saves energy resources of wireless nodes. Another example is CDMA 2000 HDR (High Data Rate) [2], where each node periodically measures the quality of the up-link channel and sends it to the base station. The base station gives higher priority to the user with the better channel condition to improve overall system performance.

Clearly, cross layer design necessitates some degree of violation of the strict layering principles of the traditional layered architecture. Therefore, we need to expend some effort in standardizing a cross layer architecture. We present one such architecture in the following section.

A. A Cross Layer Feedback Architecture

Performance is a short term gain, while a sound architecture can lead to exponential growth of the system [20], as illustrated by the success of the TCP/IP architecture. Thus, while we advocate Cross layer design, we must note the very real danger: unbridled cross layer design can lead to spaghetti-like code which is hard to maintain or trust. For instance, a naive Cross layer optimization can produce unintended interaction among different layers resulting in poor or incorrect performance. For example, an undesirable adaptation loop is introduced when strong error recovery, i.e., the packet is re-transmitted until a success occurs, is invoked both at the link layer and end-to-end [25]. If a Medium Access Control (MAC) protocol at the link layer implements in-order *strong* error recovery function then a lost packet blocks the delivery of all the other packets residing in its output queue. During this period the TCP sender does not receive acknowledgment from the TCP receiver. The TCP sender is forced into spurious timeout and invokes slow start, that will trigger false retransmission of all the un-acknowledged packets. It is possible that the MAC protocol has more than one copy of a packet residing in its output queue at a given instant of time. We notice that there are two loops of error-recovery, both of which are working on the same packet, therefore, to achieve better performance these two control loops have to coordinate with each other. This is one of the reasons that IEEE 802.11b MAC limits the number of retransmissions at the link layer.

Thus, as in [20], we feel that Cross Layer design should be holistic, which should not break the existing stack architecture. For example, Cross Layer Feedback should not break the modular design in a way that it creates any hindrance to the future growth and evolution of the system. We quote from [20]:

> Various optimization opportunities do present themselves through cross layer design, and the temptation cannot be ignored. Caution needs to be exercised, though. Once the layer is broken, the luxury of designing a protocol in isolation is lost, and the effect of any single design choice on the whole system needs to be considered.

At the moment when researchers across the globe are trying to solve performance issues of wireless applications using cross layer optimizations, it is paramount that some effort should be carried out towards standardizing a cross layer architecture. In a Cross Layer Feedback architecture we want to preserve the benefits accrued due to layered architecture, at the same time we want layers to export important performance parameters or *feedback* such as remaining battery life, SINR, packet loss rate, Quality-of-Service (QoS) requirement, etc., to optimize the performance of the TCP/IP stack over wireless links. The traditional TCP/IP stack can be a good baseline design for the cross layer architecture. Figure 3 exemplifies a simple *Cross Layer Feedback* architecture, derived from the TCP/IP stack. Each layer publishes some layer specific parameters (or *feedback*) which are used by the protocols at other layers to dynamically adapt the protocol behavior to optimize the performance. For example, the physical layer can publish SINR, BER, transmit power, etc., which can be used by the MAC protocol to adapt the access control algorithm. Similar to MobileMan [7], the benefits of cross layer feedback architecture are:

- We retain the benefits of a modular stack, therefore, the architecture will enable longevity and future extensibility of the system.
- A clean interface enables the possibility of adding more optimization parameters, and to determine if any adaptation loop exists.
- A legacy system will be able to work correctly with the cross layer optimized stack, although with degraded performance.

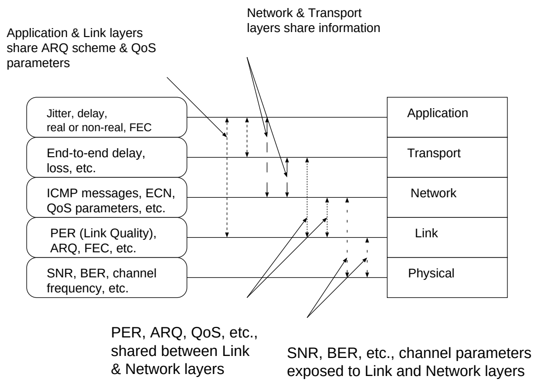
Application & Link layers share ARQ scheme & QoS parameters
Network & Transport layers share information
Jitter, delay, real or non-real, FEC
End-to-end delay, loss, etc.
ICMP messages, ECN, QoS parameters, etc.
PER (Link Quality), ARQ, FEC, etc.
SNR, BER, channel frequency, etc.
Application
Transport
Network
Link
Physical
PER, ARQ, QoS, etc., shared between Link & Network layers
SNR, BER, etc., channel parameters exposed to Link and Network layers

Fig. 3. Cross layer feedback architecture based on TCP/IP stack.

V. Examples of Cross Layer Feedback in Wireless Networks

In the following sections we present possible cross layer optimizations at the link, network, and, transport layers. Table I lists the parameters published by each layer and possible usage at other layers in optimizing the performance of the protocols. For example, the link layer is suitable for handling local congestion, therefore, the link layer protocols use the physical layer parameters to optimize the channel access algorithm.

A. Link Layer

The medium access control (MAC) protocols control the access rights of the shared medium. The IEEE 802.11 standard specifies a 2.4 GHz operating frequency with multiple data rates. These multiple data rates are possible because of different modulation techniques optimized for a specific data rate. IEEE 802.11b supports transmission of data at a rate of 1, 2, 5.5, or 11 Mbps. The Quadrature Phase Shift Keying (QPSK) with 8-bit Complementary Code Keying (CCK) error correction codes provides a data rate of 11 Mbps, since this technique has 8 bits/symbol, therefore, high SNR is required to support a constant BER. On the other hand, QPSK with 11 bit Barker Sequence supports 2 Mbps data rate with 2 bits/symbol, therefore, it requires low SNR to achieve the same BER. To understand the relation among modulation, throughput, and SNR, we need to understand how modulation works. *Modulation* is a process of translating an outgoing data stream into a form suitable for transmission on the physical medium. In digital modulation the data stream is translated into a sequence of *symbols*. The number of bits encoded to one symbol depends on the modulation scheme. For a constant transmit power, higher data rate modulation schemes pack more bits per symbol, therefore, *average energy/bit* is reduced. This is the reason that the transmission range is inversely proportional to the modulation rate. In general, a high data rate modulation scheme will have smaller range in comparison to a lower data rate modulation scheme.

The IEEE 802.11a standard supports an OFDM physical layer (PHY) that splits an information signal across 52 sub-carriers to provide data rates of 6, 12, 18, 24, 36, 48, or 54 Mbps. We plot OPNET simulations results of a 802.11a network comprising of one sender & receiver pair in Figure 2. We observe that there is a first order relation between SNR and system throughput. To achieve 54 Mbps data rate a very high SNR is required.

The medium access control (MAC) protocol controls the access rights of the shared medium. A MAC protocol uses physical layer parameters such as SINR to determine if the channel is busy or not. Even in the case when the channel is free, the channel conditions can degrade due to time varying physical layer properties described in Section III. The fact that the SNR strongly correlates with the system throughput is exploited by rate adaptive MAC protocols, ARF, RBAR, and OAR ([19], [14], [30]).

1) Auto Rate Fallback Scheme (ARF): The Auto Rate Fallback (ARF) [19] protocol was the first implementation which used SNR feedback from the PHY layer to adapt the transmission rate to optimize the system throughput. With ARF, the sender attempts to use a higher data rate after a series of successful data transmissions, since a series of successes indicates good channel conditions. The sender lowers the data rate after a fixed number of consecutive failures. ARF uses long term channel estimates, therefore, if channel quality changes rapidly then the selected transmit-rate might not give optimum results. Another drawback of ARF is that when ACKs are missed at the sender it does not indicate that channel conditions at the receiver are also bad. It could be due to high channel contention at the sender. *ARF overlooks the fact that it is the receiver whose channel conditions need to be sampled.*

2) Receiver Based Auto Rate Scheme (RBAR): The fact that it is necessary to sample channel conditions at the receiver is exploited by the Receiver Based Auto Rate (RBAR) [14]. In a cellular network, a pilot channel is used to feedback the instantaneous channel conditions at the receiver. The base station uses this channel characteristic to optimize the performance in a cell. Unfortunately, WLAN is half-duplex, and unlike cellular networks there is no pilot channel which can provide instantaneous feedback to the sender. Therefore, in a WLAN a simultaneous feedback mechanism is impossible. The IEEE 802.11 standard uses RTS and CTS control packets before the actual transmission of data packets. In RBAR, the receiver samples the channel and selects the *best* data rate and uses reserve fields in the CTS control packet to convey the transmission rate to the sender. The piggybacked transmission rate is used by the sender to transmit the forthcoming data packet. *Does one RTS packet accurately estimate the channel? And how long does the channel remain stable?* This question is answered by OAR.

3) Opportunistic Auto Rate Scheme (OAR): The Opportunistic Auto Rate Scheme (OAR) [30] exploits the fact that when channel conditions are good more packets should be transmitted in a row. The OAR exploits the properties of *channel coherence time*. The *Coherence time* is the time duration for which the channel impulse response is essentially invariant, and quantifies the similarity of the channel response at different times [23]. In other words, the coherence time is the time duration for which two received signals have a strong potential for amplitude correlation or the SNR values do not change and the BER remains fairly constant. Thus channel quality remains stable for a fixed window. For IEEE 802.11 networks the coherence period corresponds to multiple packet transmission times. OAR exploits the coherence time of the channel and uses good channel conditions at the receiver to transmit multiple packets in a row. Similar to RBAR, in OAR the receiver uses the RTS packet to obtain instantaneous channel conditions and uses the CTS control packet to provide feedback about the receiver's channel condition. OAR uses the coherence time to transmit a burst of data packets after successful exchange of RTS & CTS control packets.

4) Medium Access Diversity (MAD): In a cellular network, the signals from different users are transmitted over channels having different characteristics. This results in different received power at the base station. The average received power of a given user depends upon its distance from the base station, therefore, there is a certain loss in the signal strength which is known as *path loss*. On the other hand, the instantaneous power is time varying due to multipath fading [23]. To overcome this time varying power phenomenon, certain *power-control* schemes are used in a cellular network. A scheme is proposed in [21] where a user with the highest instantaneous power relative to other users is allowed to transmit. It is also possible that all the users have instantaneous received power below a certain pre-specified threshold. Under such condition none of the users will be using the channel. This is known as multiuser diversity (MUD).

Medium Access Diversity (MAD) [17] is motivated by MUD. MAD is designed for IEEE 802.11 networks. In MAD, the sender broadcasts a *query* message including an ordered list of potential receivers for channel condition probing. Each receiver replies in the order of the receiver's id in the probe list. The *reply* message contains channel condition information obtained from sampling the preceding *query* message. The sender selects a user which reports the highest transmission rate, or a receiver whose channel has highest SNR value. After determining the potential receiver which can maximize the system throughput, the sender transmits data packets, and the receiver responds with an ACK if the received packet is error free.

[29] presents a CDMA based multiple packet reception MAC protocol for wireless networks. The CDMA codes allocated to a transmitting node are dynamically adjusted based on the traffic demand. With an increase in the channel load, the channel access policy is shifted to a contention-free from a contention- based scheme. The physical layer parameters such as BER and CDMA code collision probability are used by the MAC protocol to determine whether it should operate in contention-free or contention-based mode.

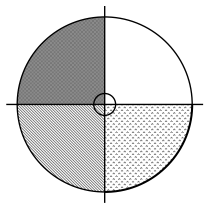

Fig. 4. Sectored Antennas.

5) Directional Antennas and MAC Scheduling: In this section we illustrate a simple scheduling algorithm at the link layer in the context of a cellular network where channel conditions are fed to the link layer as *cross layer feedback*.

An omnidirectional antenna receives and transmits RF energy equally in all directions. Therefore, it creates undesired interference at some of the nodes, resulting in limited spatial reuse of the shared channel. In comparison to omnidirectional antennas, directional antennas radiate energy in smaller zones. Therefore, using directional antennas it is possible to reduce CCI, hence, SNR can be increased which results in high system throughput. Sectored antennas are the simplest form of directional antennas. Sectored antennas are made of M sectors, where each sector has a spanning angle of $2\pi/M$ radians. Figure 4 provides an example of four-sectored antennas.

Let us assume that the MAC protocol can use only one sector for either transmission or reception at any given instant. The wireless channel corresponding to each sector can take one of the two states, *Good* or *Bad*, equally likely. Furthermore, the channels are random and i.i.d. (independent and identically distributed) process, therefore, they are statistically independent of each other. The four sectors can be in one of the sixteen states from $\{$Good, Good, Good, Good$\}$ to $\{$Bad, Bad, Bad, Bad$\}$. Assume that the base station uses some kind of multi user detection (MUD) algorithm to determine if the channel corresponding to a user falling within a sector is in Good or Bad condition. For example, GSM uses a pilot channel to obtain this information. For simplicity, assume that there are four users - A, B, C, and D - belonging to each sector, as shown in Figure 5.

Let us observe the maximum throughput attained by each user if the MAC protocol at the base station implements a naive scheduling algorithm such as round-robin. The user can be either in the Good or in Bad state. Since these events are equally likely, therefore, the probability that the user is in the *Good* state is 1/2. Since the base station uses round-robin scheduling, therefore, each user will have service time equal to 1/4 of the total service time. Thus, each user achieves a throughput of $1/2 \times 1/4 = 1/8$.

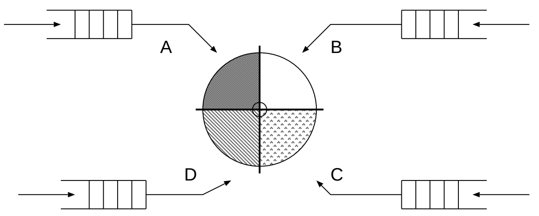

Fig. 5. MAC scheduling using Cross layer feedback.

Now we can use *cross layer feedback* to optimize the scheduling algorithm. More specifically, the base station uses cross layer feedback from the physical layer to determine the instantaneous state of the channels. The base station uses this information to schedule the delivery of a packet to a user whose channel conditions are $Good$. The base station cannot transmit if all the four channels are in Bad state; the probability of this event is 1/16. Therefore, the probability that at least one of the channels is in $Good$ condition is $1 - 1/16 = 15/16$. On average, each user will achieve a throughput of $15/16 \times 1/4 = 15/64$. Using this very simple cross layer feedback based MAC scheduling algorithm, each user doubles the throughput of the round-robin scheduling scheme.

6) Summary: In each of the above schemes we show the potential benefits of cross layer feedback in designing access control and scheduling algorithms at the link layer. The physical layer parameters *fed* to the link layer are used in designing high performance MAC protocols. We summarize the aforesaid cross layer feedback schemes as follows:

- Adaptive modulation and IEEE 802.11 MAC.
- Directional antennas and adaptive scheduling.
- Adaptive CDMA codes and CSMA/CA MAC.
- SNR and IEEE 802.11 MAC.

B. Network Layer

The *network* layer handles the delivery of packets across the network. The *network* layer also provides the abstraction of a path. Wireline routing protocols normally use a hop count routing metric to find optimal routes. The minimum hop count metric tends to find the routes which minimize the hop count or maximize the physical distance of each hop. For example, in Figure 6 A has a packet for F, a minimum hop count routing metric selects $A \rightarrow H \rightarrow G \rightarrow F$ over $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$.

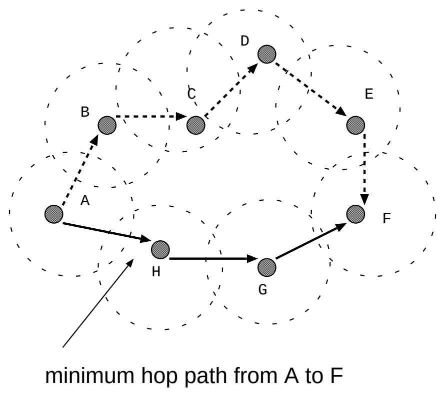

Fig. 6. Wireless Network.

Unfortunately, in wireless settings there are numerous drawbacks to this form of path selection:

1) The hop count routing metric assumes that the wireless channel quality does not vary with time. Therefore, even under conditions when one of the links of the selected path ($A \rightarrow H$ or $H \rightarrow G$ or $G \rightarrow F$) undergoes channel fading, the routing protocol continues to use the least hop count path. This results in poor throughput for the flow.
2) In comparison to short range links, long range links tend to have smaller SNR. If the channel quality of one of the minimum hop count paths fluctuates with time, it leads to a phenomenon called "grey zones". Therefore, in comparison to the short range links, the long range links are more vulnerable to time varying channel conditions.
3) Assume the minimum hop count metric is used with the adaptive data rate protocols discussed in Section V-A. The adaptive MAC protocols favor short range links to increase the system throughput; on the other hand the minimum hop count metric is biased towards long range links. This is a more serious issue, since we have created two optimization loops which are working against each other. Therefore, a naive implementation of cross layer feedback techniques can eventually end up in negatively affecting performance.

In the following sections we present the benefits of cross layer feedback schemes in designing routing metrics suitable for wireless ad hoc networks.

1) Expected Transmission Count Metric (ETX): [8] Introduces a path metric, ETX, which finds high-throughput paths in multi-hop wireless networks. ETX abstracts the total number of packet transmissions (including retransmissions) required to successfully deliver a packet. ETX selects a path with minimum number of transmissions to the receiver. ETX captures the effects of link loss, asymmetric links, and interference among successive links.

The ETX of a link is calculated using the forward and the reverse delivery ratios of a link. Given d_f, the measured probability that data arrives at the destination, and d_r, the probability that the ACK is successfully received, ETX for the link can be calculated as follows:

$$ETX = \frac{1}{d_f \times d_r} \tag{2}$$

How to measure the link probabilities? Each node periodically broadcasts link probes (134 bytes of 802.11b payload, every 1 sec, use jitter of ± 0.1 per probe). Each probe sent by a node X contains the number of probes received from its 1-hop neighbors during the last w seconds (10sec). Each node updates the delivery ratio for a sender based on the probes received in the last w seconds. It is important to periodically transmit probe packets to avoid probe drop due to channel error or buffer overflow. The implementation maintains priority queues for probe packets, protocol packets, and data packets, in that order. Each of these queues can hold up to five packets. ETX was implemented in DSDV and DSR. The simulation results show that ETX often finds higher throughput paths than minimum hop count paths, particularly for routes with three or more hops.

[9] propose three new metrics - ETX, per hop RTT, and per hop packet pair. Per-hop RTT is computed based on round trip delay seen by unicast probes between neighboring nodes. To calculate RTT, a node sends a probe packet (137 bytes) to each of its neighbors every 500msec. Each neighbor immediately responds to the probe with a probe acknowledgment echoing the time stamp. This enables the sending node to measure the RTT to each of its neighbors. The node keeps an exponentially weighted moving average of the RTT samples to each of its neighbors.

$$\text{Average} = 0.1 \times \text{RTT Sample} + 0.9 \times \text{Average}. \quad (3)$$

If the probe response or the data packet is lost, the moving average is increased by 20%. The routing algorithm selects the path with the least total sum of RTTs. Therefore, this RTT metric avoids highly loaded or lossy links. There is a downside to this algorithm where a lightly loaded or reliable link will have lower RTT value, therefore, it attracts more traffic causing its RTT value to go up, leading to route flapping. This RTT metric can also have the side-effect of long queuing delays.

The per-hop metric is based on measuring the actual delay between two neighbors, it also overcomes some of the short-comings of the RTT metric. A node sends two back-to-back probes. The first probe is small (137 bytes) and the second probe is large (1000 bytes). The neighbor starts a timer upon receiving the first probe packet, and stops the timer upon receiving the second packet. It then sends the value of this timer, which is the delay between receipt of the first and the second packet, back to the sending node. Similar to the RTT metric the sender maintains an exponentially weighted moving average of these delays of its neighbors.

The three metrics were implemented in DSR, the new protocol is named Link-Quality Source Routing (LQSR), which is shown to perform better than simple hop count routing.

2) Signal Strength Based Metric: The shortest hop count metric does not take into account the varying wireless channel conditions described in Section III, therefore, it treats all the one-hop neighbor links alike. If nodes could filter the reliable one-hop links and use these links in the routing path to other nodes, it is possible to increase the network performance. This is the concept of the Signal strength based route selection metric, [6]. A node periodically sends Echo requests, each neighbor's link quality is calculated based on the echo response from the neighbor. *Ipchains* were used for implementing this Signal-strength-based metric. It was found that a neighbor selection based on the quality of the link resulted in reliable multihop connections in a real testbed.

3) Routing Metric in Multi-radio Networks: The decrease in 802.11 radio prices has opened research challenges in designing the routing metrics for nodes equipped with multiple radios [10], [28]. Wireless networks comprising of multiple-radio nodes have the following advantages over single-radio nodes.

1) Traditionally IEEE 802.11 networks are half-duplex, i.e. a node can either transmit or receive, not both at the same time. However, using multiple-radios (or, interfaces) nodes can transmit and receive simultaneously, hence, providing a full-duplex channel.
2) Nodes can have higher reuse of the channel, thus providing higher spectral efficiency.
3) Multiple radios provide high fidelity against hardware malfunctioning.
4) Multiple heterogeneous radios (for example, 802.11a & 802.11b, 802.11b & low power and low bandwidth wireless sensors) provide the flexibility for traffic engineering and increasing network performance.

The IEEE 802.11 standard provides several orthogonal channels, for example, 802.11b provides 3 and 802.11a provides 12 orthogonal channels. When nodes are equipped with multiple radios, a shortest-path algorithm does not result in optimum performance. Consider a network where each node is equipped with two radios tuned to channel 1 & 11. Consider a 2-hop path in this network. A shortest-path routing metric will result in using either channel 1 or 11 over the entire path, therefore, it does not provide simultaneous transmissions.

[10] presents a Multi-Radio Link-Quality Routing (MR-LQSR) protocol for nodes equipped with heterogeneous radios. MR-LQSR is a combination of the LQSR [9] protocols and WCETT (Weighted Cumulative Expected Transmission Time) routing metric. ETT (expected transmission time) can be computed using ETX [9], packet size S, and B bandwidth of the channel as follows:

$$ETT = \frac{ETX \times S}{B} \quad (4)$$

Consider a n-hop path. Assume that the system has a total of k channels. X_j, the sum of transmission times of hops on channel j is defined as:

$$X_j = \sum_{Hop\ i\ is\ on\ channel\ j} ETT_i \quad 1 \leq j \leq k \quad (5)$$

The total path throughput will be dominated by the bottleneck channel j which has the highest X_j. Using equation 4 & 5, we can write a simple equation for WCETT:

$$WCETT = (1-\beta) \times \sum_{i=1}^{n} ETT_i + \beta \times \max_{1 \leq j \leq k} X_j \quad (6)$$

where β is a tunable parameter between [0,1].

WCETT favors paths that are more channel diverse, and therefore, provides simultaneous transmissions and high spectral efficiency.

4) Summary: In the previous sections we reviewed the research work in the context of routing metrics using cross layer feedback. Link quality metrics such as ETX, MTX, WCETT, etc., use the PHY layer parameters such as channel quality, channel data rate, channel frequency, etc., in designing routing metrics suitable for wireless networks. The experimental studies show that these metrics outperform the traditional minimum hop count metric in wireless networks.

C. Quality of Service (QoS) and Application Layer

The higher bandwidth provided by the next generation of WLANs, such as IEEE 802.11a and IEEE 802.11g, has enhanced the possibility of using video over wireless applications. However, real-time applications require some QoS support such as guaranteed bandwidth, delay, and jitter. To meet the QoS requirements of these applications, numerous cross layer techniques have been proposed in the literature. We present a few such techniques to emphasize the benefits of cross layer feedback in improving QoS.

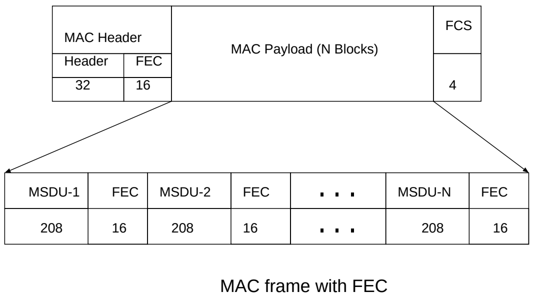

Fig. 7. Forward Error Correction (FEC) [34].

1) Forward Error Correction (FEC): Forward Error Correction (FEC) involves padding redundant bits that help to recover bits received in error. Therefore, using FEC a frame could be made resilient to few bit errors. Employing FEC at the MAC level for improving the robustness to losses is currently under consideration in the IEEE 802.11 standards committee [34]. Each MAC Protocol Data Unit (MPDU) frame consists of a MAC header, a variable length MAC Service Data Unit (MSDU), and a frame check sequence (FCS). Using shortened Reed-Solomon (RS) codes (224, 208) a MSDU can be encoded as shown in Figure 7. Note that any RS block can correct up to 8 byte errors. FEC can be applied at the application layer or at the link layer. [34] compares the trade-off of applying FEC at the application layer against the link layer. It was observed that under good channel conditions, the optimal application layer FEC outperforms the MAC FEC, but under moderate and poor channel conditions the MAC FEC provides greater benefits. This is because the bit level correction codes provided by the MAC FEC is more effective in combating bit errors than the packet level protection provided by the application layer FEC.

Thus, using cross layer feedback from the PHY layer, it is possible to dynamically switch between application layer FEC and link layer FEC as the channel conditions worsen. We can envision that such a scheme will outperform strict application layer- or link layer-FEC for any channel condition.

D. Automatic Repeat reQuest (ARQ)

ARQ is an error control mechanism implemented at the link and the transport layers. In the IEEE 802.11 standard, a sender transmits a data frame and waits for the reception of the acknowledgment (Ack). In the event the Ack is lost, the sender retransmits the frame, and this process continues until the correct Ack is received or the sender reaches a maximum retransmit limit. To combat high error rates in the wireless channel, some reliability is introduced at the link layer via. hop-by-hop acknowledgment. We exemplify the benefits of hop-by-hop acknowledgment by providing a simple example. Consider a multihop wireless network where one-hop successful packet transmission probability is P_{success}, N, the number of transmission attempts before a packet is discarded by the transmitter, X, the number of hops a sender A is from the destination B. We can write a simple equation for the successful packet reception probability at the destination as follows:

$$P_{\text{success}}(A,B) = (1-(1-P_{\text{success}})^N)^X \quad (7)$$

We present analytical results of end-to-end success rate as a function of X for $N = 1$ in Figure 8 and for $N = 3$ (i.e. the transmitter discards the packet after two retransmission attempts) in Figure 9. We observe that:

- End-to-end success probability decreases with the increase in the number of hops.
- Retransmissions help in improving end-to-end packet success rate.

This simple example demonstrates the benefits of the link layer error recovery mechanisms in wireless networks. The current MAC of IEEE 802.11 WLAN uses the ARQ

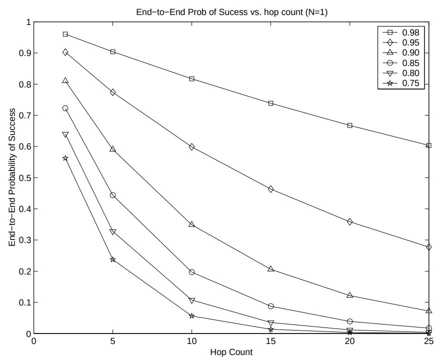

Fig. 8. End-to-end success rate in different networks.

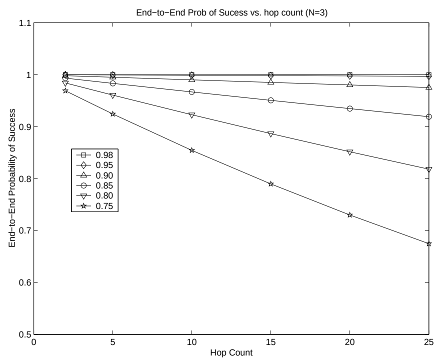

Fig. 9. End-to-end success rate in different networks.

mechanism because of its simplicity over FEC [24]. However, FEC provides constant time overhead, while ARQ provides a simplified algorithm at the cost of higher latency. Therefore, based on the application's latency requirement (such as real-time voice traffic or video streaming), and cross layer feedback from the application layer or the network layer, a proper error recovery algorithm can be selected which meets the latency requirements of the data stream.

Both the transport and the link layers can implement ARQ. The link layer ARQ ensures error free transmission of the packet over a single hop. In general, the link layer ARQ is more efficient in comparison to the application/transport layer ARQ. The reason is that the link layer ARQ is invoked over a single wireless link, while application/transport ARQ is end-to-end, resulting in a higher delay. Another potential benefit of link ARQ is that it operates on relatively small size frames in comparison to IP datagrams, therefore, it is more efficient to re-transmit link frames [25].

However, this does not always mean that the link layer ARQ is the best choice. [25] shows how an adverse control loop is introduced when ARQ is implemented both at the link and the transport/application layers. In an adverse situation, the link layer is retransmitting one or more packets, and simultaneously one of the network end points may assume that the packet is lost, triggering a retransmission. It may even happen that two or more copies of the same packet reside in the sending buffer of the link layer at the same time. [25] presents a scheme to prevent such adverse control loop interactions when ARQ is implemented at both link and application/transport layers.

In [3] application level ARQ is implemented, where based on the *temporal* and *perceptual* importance of each packet, a priority of the packet is determined. A packet retransmission is scheduled based on this priority. For example, in the case of MPEG2 video, I-frames are more important than B- and P-frames in terms of perceived quality. Simulation results show performance gains when application layer ARQ is introduced over link layer ARQ mechanism for this application.

E. Transport Layer

Transmission Control Protocol (TCP) provides a connection oriented reliable byte stream on top of the unreliable datagram service provide by the IP layer [27]. TCP provides reliability by sending data combined with positive acknowledgments (ACKs) and retransmissions. Each byte in the transmission data is numbered, starting at some value and increasing over a period of time. The TCP sender keeps a variable called Congestion Window (cwnd). The cwnd is initialized to one segment, and each time the Sender receives a positive ACK it increases cwnd by one segment. It is possible that packets can be lost due to congestion and buffer overflow. When a TCP receiver misses a packet (due to reordering or loss) on a connection but receives several packets that are in sequence, it generates an ACK for the missing packet. TCP sender maintains an estimate of the round trip time (RTT), i.e., the time it takes for the segment to travel from the Sender to the Receiver plus the time it takes for the ACK (and/ or any data) to travel from the Receiver to the Sender. The variable RTO (Retransmit Time Out) maintains the value of the time to wait for an ACK after sending the segment. Therefore, the TCP Sender detects loss by either triple duplicate ACKs or timeout (expiry of RTO), Figure 10. The TCP Receiver also maintains a variable called Advertised Window (awnd), and if the receiver's buffer is full then it can inform the Sender to not send any further data. This prevents a fast sender swamping a slow receiver.

The TCP Sender uses *feedback* from the Receiver and Network to regulate the packet flow. The TCP Sender uses ACKs from the Receiver to increase the *cwnd*, or increase the transmission rate. The Sender uses duplicate ACKs to determine that some transmitted packets are lost

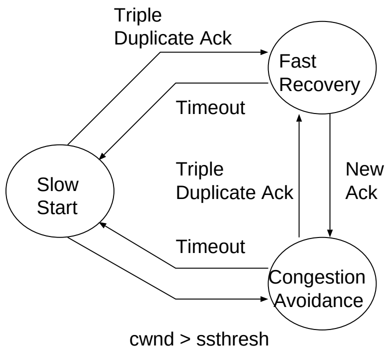

Fig. 10. TCP state diagram.

and that need to be re-transmitted in the future. The TCP Sender uses the *cwnd* variable to throttle the flow. The TCP Sender uses RTO to invoke Congestion Control by reducing the cwnd to one or two segments and performing Slow Start. TCP dynamically adapts the data flow rate to the receiver's advertised window and network's level of congestion. We feel that the design of TCP is a perfect example of *Control Feedback*. However, TCP results in low throughput when used in wireless networks ([1], [15], [12]). This poor performance can be attributed to frequent channel errors caused by fading and multipath, handoffs, node-mobility, and large RTTs. Since TCP treats packet loss and congestion alike, it cannot distinguish between the two conditions and invokes congestion control even when a packet is lost. This is because TCP is designed for wireline networks where most of the time congestion is the cause of packet loss. We identify three scenarios where due to limited cross layer feedback the TCP connection attains low performance:

- *Routing failure:* As nodes move, the route between the sender and receiver may break and the underlying routing protocol will attempt to find a new route. However, if the time to find a new route and forward the packet is longer than the retransmit timer of the sender, we will see a timeout event at the sender which causes the congestion window to shrink. A related problem pointed out in [15] is that the underlying routing protocol (DSR [18] in this case) may find invalid routes due to the cache reply mechanism incorporated into the protocol. This causes further delays in finding a route and can lead to serial timeouts at the sender which can become very long. Finally, it is possible that even if a route is found, the TCP sender may not send data because it is waiting for a timeout to retransmit data. By the time the sender times out, the route may no longer be valid! In all of these cases, the TCP sender is idle for long periods of time, many retransmissions take place when there is no route, and the congestion window is small. Thus, the TCP throughput will be very small.
- *Out-of-order packets:* Due to route changes during the lifetime of a TCP connection or due to multi-path routing, it is possible for packets and/or ACKs to arrive out-of-order. This can cause the sender to receive *triple duplicates* which in turn results in the sender retransmitting the offending packet and shrinking its congestion window by a half. Thus, the overall system throughput will be very small.
- *Routing Congestion:* Routing failure can cause network congestion if there are several active connections ([26]). This is because each route failure forces the routing protocol to find new routes and as a consequence, the number of control packets in node buffers can increase significantly. Similarly, TCP senders will timeout and retransmit packets that may already be present in the buffers of intermediate hops. These two factors taken together can result in congestion at one or more nodes in the network which in turn results in lowered TCP throughput.

TCP's design is not optimized for the three network conditions discussed above that are endemic to wireless networks. Therefore, in each case, TCP misinterprets the network state, resulting in poor throughput and excessive packet retransmissions. Thus, in order to improve TCP's performance we need to consider *cross layer feedback* from other layers so that TCP can take appropriate action to achieve high throughput. The cross layer feedback from different layers that changes TCP's operation and increases its performance are described in the following sections. We call the modified TCP design TCP-ECN-ELFN.

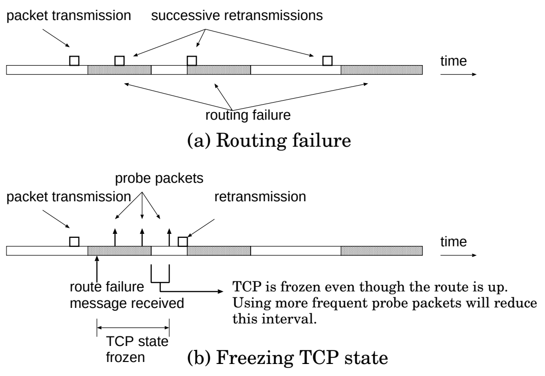
packet transmission
successive retransmissions
time
routing failure
(a) Routing failure
probe packets
packet transmission
retransmission
time
route failure message received
TCP is frozen even though the route is up. Using more frequent probe packets will reduce this interval.
TCP state frozen
(b) Freezing TCP state

Fig. 11. Fixing the routing failure problem.

1) Routing Failure: [15] describes the interplay between routing failure (due to link outage or propagation of stale routes) and TCP throughput in detail. In this section, we provide a simple description of this interplay and use this description to explain why TCP's throughput will be increased if we adopt the solution proposed in [15], [5]. Figure 11(a) shows the evolution in time of a TCP connection. The shaded boxes represent periods when there is a routing failure resulting in dropped packets. As shown in the figure, after transmitting one window of packets, the sender waits for ACKs that never arrive (due to routing failure). The sender then times out at t_1 and retransmits the packet. It so happens that this packet is transmitted during the time when there is no route and therefore the packet is lost. A second timeout occurs at t_2 which also lies in a period of routing failure. The third timeout occurs during a time when the route is up and the packet is delivered successfully. The following conclusions can be drawn:

- TCP mistakenly invokes congestion control due to a routing failure. This reduces the size of the congestion window which in turn reduces the throughout.
- Retransmissions that occur during a routing failure are wasteful in two ways – it is an unnecessary packet retransmission since the packet will likely be lost anyway, and the exponential backoff algorithm increases the timeout interval for the next timeout.
- Due to the large timeout intervals (due to exponential backoff), the sender does not send packets even when the route is up, thus reducing throughput.

Figure 11(b) shows the fix that was proposed by [15]. Here, a *route failure* message is propagated back to the TCP sender from the intermediate node that detects the route failure, for example, *ICMP Destination Unreachable*. The IP layer of the sender does not forward this packet to the the TCP layer. However, if the TCP/IP stack is modified so that the *route failure* message is signaled to the the TCP layer, the TCP sender can freeze TCP's state and initiate the transmission of probe packets. When there is a response to the probe packet (i.e., TCP ack of data from the TCP sender), TCP's state is unfrozen and transmission resumes. As shown in the figure, this solution ensures that there are no timeouts (and hence no unnecessary retransmissions), and that the TCP sender begins sending packets soon after the route is up. It is noteworthy that the probe packet period needs to be fairly small to minimize the time the sender remains frozen after a route is found.

[15] presented this solution in the context of using DSR as the underlying routing protocol. Observe, however, that this solution performs well regardless of the underlying routing protocol because no protocol can guarantee a route at all times. Thus, using a route failure cross layer feedback to freeze the TCP state until a route is found appears to be the best solution for TCP.

2) Out-of-order Packets, Timeouts, & Triple Duplicate ACKs: Mobility of nodes can cause packets belonging to the same connection to be routed along different routes. This can result in the receiver getting out-of-order packets which causes duplicate ACKs to arrive at the sender. Likewise, packet loss due to link layer errors can result in triple duplicate ACKs or timeouts. On receiving three duplicate ACKs, the sender reduces its congestion window by a half and retransmits the out-of-sequence packet while in the case of timeouts, the window is reduced to one or two segments. This congestion avoidance behavior has the net effect of reducing the throughput of the connection (due to the smaller congestion window) and thus increasing overall energy consumption. We believe that the appropriate fix for this problem is for the TCP sender to retransmit the offending packet but *not adjust its congestion window*.

3) Using Explicit Congestion Notification (ECN): A problem with our approach above in section V-E2 is that if the triple duplicates (or timeout) were generated as a result of packet drops due to congestion, then the solution of simply retransmitting the packet without reducing the congestion window will have negative consequences. (This is the reason why TCP reduces its congestion window). In our design, we rely on *explicit congestion notification* [11] to signal imminent congestion along a route. Here, a node whose buffer occupancy crosses some threshold, sets a bit (the CE bit) in all data packets it sees. Receivers reflect this flag back in the ACKs they generate by setting the ECN-ECHO bit. Upon receiving an ACK with the ECN-ECHO bit set, TCP senders enter a recovery phase in which they reduce the congestion window by a half. The sender sets a CWR (Congestion Window Reduced) bit in new data packets. If the receiver sees another CE bit set in a future packet and sees that the sender had sent a CWR bit, this indicates that there is still congestion in the network. The receiver again sets the ECN-ECHO bit in new ACKs, thus forcing the sender to enter another recovery phase. This can go on until the sender's window has shrunk to one or two segments. In order to avoid race conditions, the sender only responds once per window to ECN-ECHO flags. In the ad hoc environment, using ECN rather than relying on timeouts or triple duplicates makes it possible for the protocol to react "appropriately" to different loss events in the network.

4) Summary: Table II summarizes the changes made to the operation of TCP. In summary, we exploit cross layer feedback in freezing TCP in the presence of routing failure ([15]) as well as using explicit congestion notification ([11]) to distinguish between congestion events and out-of-order or lost packets. This set of modifications will avoid spurious invocation of TCP's congestion control algorithms in quick succession. Thus, maintaining large *cwnd* and achieving high throughput.

One of the drawbacks of our modified approach is that TCP-ECN-ELFN will not share bandwidth fairly with other traditional TCP connections. This is because normal TCP will reduce its congestion window when it sees timeout or triple duplicate events whereas the TCP-ECN-ELFN protocol will only reduce the congestion window when it receives an ECN. Thus, random packet loss that occurs in networks will only serve to increase the throughput of the TCP-ECN-ELFN protocol in relation with normal TCP connections and may completely starve these TCP connections over time. For this reason, we believe that the TCP-ECN-ELFN protocol should only be used in networks where all nodes use the TCP-ECN-ELFN protocol. However, it is noteworthy that even in this case, TCP-ECN-ELFN connections starting at different times may see an unequal sharing of bandwidth over a shared link for long time periods. Therefore, using cross layer feedback between the transport and the link layer to implement some control of bandwidth allocation would be useful to ensure fair sharing.

VI. Conclusions

Cross Layer Feedback provides opportunities for improving the performance of wireless networks. However, we see that while applying cross layer feedback caution must be exercised to avoid any negative adaptation loops which exacerbate the problem. Cross layer feedback mechanisms achieve increased performance at the cost of some degree of violation of the strict layering principle of the traditional layered architecture. We feel that a cross layer stack architecture should be proposed for the massive proliferation of wireless multimedia applications. The TCP/IP stack is a good candidate for the underlying stack architecture.

References

[1] Mark Allman, Chris Hayes, Hans Kruse, and Shawn Ostermann. Tcp performance over satellite links. In *5th International conference on telecommunication systems*.

[2] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi. Cdma/hdr: A bandwidth-efficient high-speed wireless data service for nomadic users. *IEEE Communications Magazine*, 38:70–77, July 2000.

[3] P. Bucciol, G. Davini, E. Masala, E. Filippi, and J.C. De Martin. Application-level perceptual ARQ for H.264 video streaming over 802.11 wireless LAN's. In *Proc. Wireless Personal Multimedia Communications (WPMC)*, volume 1, pages 132–136, Abano Terme, Italy, September 2004.

[4] Gustavo Carneiro, Jose' Ruela, and Manuel Ricard. Cross-layer design in 4g wireless terminals. *IEEE Wireless Communications Magazine*, 11(2):7–13, Apr 2004.

[5] Kartik Chandran, Sudarshan Raghunathan, S. Venkatesan, and Ravi Prakash. A feedback based scheme for improving TCP performance in ad-hoc wireless networks. In *International Conference on Distributed Computing Systems*, pages 472–479, 1998.

[6] Kwan-Wu Chin, John Judge, Aidan Williams, and Roger Kermode. Implementation experience with manet routing protocols. *SIGCOMM Comput. Commun. Rev.*, 32(5):49–59, 2002.

[7] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-layering in mobile ad hoc network design. *IEEE Computer, special issue on Ad Hoc Networks*, 37(2):48–51, Feb 2004.

[8] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146. ACM Press, 2003.

[9] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 133–144. ACM Press, 2004.

[10] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 114–128. ACM Press, 2004.

[11] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):10–23, 1994.

[12] M. Gerla, K. Tang, and R. Bagrodia. Tcp performance in wireless multi-hop networks. In *IEEE WMCSA'99, (New Orleans, LA)*, Feb. 1999.

[13] Tom Goff, James Moronski, Dhananjay S. Phatak, and Vipul Gupta. Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments. In *INFOCOM (3)*, pages 1537–1545, 2000.

[14] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A rate-adaptive mac protocol for multi-hop wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 236–251. ACM Press, 2001.

[15] Gavin Holland and Nitin H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *ACM Mobile Computing and Networking (MOBICOM'99)*, pages 219–230, 1999.

[16] Fei Hu and Neeraj K.Sharma. The quantitative analysis of tcp congestion control algorithm in third-generation cellular networkss based on fsmc loss model and its performance enhancement. In *Proceedings INFOCOM 2002, New York, NY*, 2002.

[17] Zhengrong Ji, Yi Yang, Junlan Zhou, Mineo Takai, and Rajive Bagrodia. Exploiting medium access diversity in rate adaptive wireless lans. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM Press, 2004.

[18] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[19] A. Kamerman and L. Monteban. Wavelan-ii: A high-performance wireless lan for the unlicensed band. *Bell Labs Technical Journal*, pages 118–133, 1997.

[20] Vikas Kawadia and P. R. Kumar. A cautionary perspective on cross layer design. *IEEE Wireless Communications Magazine*, July 2003.

[21] R. Knopp and P. A. Humblet. Information capacity and power control in single-cell multiuser communications. In *IEEE ICC '95*, June 1995.

[22] Philip Levis, Sam Madden, David Gay, Joe Polastre, Robert Szewczyk, Alec Woo, Eric Brewer, and David Culler. The emergence of networking abstractions and techniques in tinyos. In *First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004.

[23] J. C. Liberti and T. S. Rappaport. *Smart Antennas for Wireless Communications*. Prentice Hall, 1999.

[24] Hang Liu, Hairuo Ma, Magda El Zarki, and Sanjay Gupta. Error control schemes for networks: An overview. *Mobile Networks and Applications*, 2(2):167–182, 1997.

[25] R. Ludwig. *Eliminating Inefficient Cross-Layer Interactions in Wireless Networking*. PhD thesis, Aachen University of Technology, April 2000.

[26] Jerey P. Monks, Prasun Sinha, and Vaduvur Bharghavan. Limitations of tcp-elfn for ad hoc networks. In *MOMUC'00*, 2000.

[27] J. Postel. rfc793: Transmission control protocol.

[28] Ashish Raniwala, Kartik Gopalan, and Tzi cker Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(2):50–65, 2004.

Layer	Cross layer feedback	Example of usage
PHY (Physical)	SNR, BER, channel frequency, CDMA codes, modulation etc.	1. Link layer in channel access method 2. Network layer in computing routing metrics
Link	Packet Error Rate (PER) Adaptive Repeat reQuest (ARQ), Forward Error Correction (FEC) Latency etc.	1. Network layer in computing routing metrics 2. Application/Transport Layer in deciding ARQ/FEC 3. PHY layer in using higher coding rate
Network	Internet Control Message Protocol (ICMP) messages Quality-of-Service (QoS), Explicit Congestion Notification (ECN), etc.	1. TCP layer in differentiating loss & congestion 2. Link layer in deciding ARQ/FEC
Transport	Latency & loss	1. Network & Link layers in deciding QoS

TABLE I
CROSS LAYER FEEDBACK BY DIFFERENT LAYERS.

Event	*TCP's Behavior*	*TCP-ECN-ELFN*
Routing Failure	Timeout, CWND $\longleftarrow$ 1 Retransmissions Exponential backoff timer	Freeze state [15], [5], [16] Probe network Unfreeze when route restored
Triple Duplicate (TD) ACKs	Retransmit packet CWND $\longleftarrow$ CWND/2 + 3	Retransmit packet
Timeout	CWND $\longleftarrow$ 1 Retransmit Exponential backoff timer	Retransmit packet
Explicit Congestion Notification	No action	CWND $\longleftarrow$ CWND/2

TABLE II
SUMMARY OF CHANGES MADE TO TCP.

[29] Marc Realp and A.I. Prez-Neir. Decentralised multi-access mac protocol for ad-hoc networks. In *Proceedings of 14th IEEE International Symposium on Personal, Indoor and Mobile Communications (PIMRC' 2003)*, volume 2, pages 1634–1638, Beijing, China, September 2003.
[30] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media sccess for multirate ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 24–35. ACM Press, 2002.
[31] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov 1984.
[32] Sanjay Shakkottai, Theodore S. Rappaport, and Peter C. Karlsson. Cross-layer design for wireless networks. *IEEE Communications Magazine*, 41(10):74–78, Oct 2003.
[33] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated, Volume I (The Protocols).* Addison Wesley, 1994.
[34] Xiaofeng Xu, Mihaela van der Schaar, Sunghyun Choi, Santhana Krishnamachari, and Yao Wang. Adaptive error control for fine-granular-scalability video coding over ieee 802.11 wireless lan. In *IEEE International Conference on Multimedia & Expo (ICME)' 03*, 2003.
[35] Z.H.Haas. Design methodologies for adaptive and multimedia networks. *Guest Editorial, IEEE Communications Magazine*, 39(11):106–107, Nov 2001.

Harkirat Singh received a B.E. in Electrical Engineering from the Indian Institute of Technology (IIT) Roorkee in 1993 and a Ph.D. degree in 2005 from the Portland State University at Oregon in Computer Science. Currently he is working as a Staff Engineer in the wireless connectivity at Samsung Electronics R&D center at San Jose. His areas of research include next generation WMAN, WLAN, WPAN and sensor networks. He is a member of the IEEE.