The 11th International Symposium on Wireless Personal Multimedia Communications (WPMC'08)

DELAY TOLERANCE AND THE FUTURE INTERNET

Jörg Ott
Helsinki University of Technology (TKK)
Finland

ABSTRACT

The Internet protocols were designed to provide end-to-end connectivity across heterogeneous networks, yet primarily assuming a "fixed" and rather static network environment. Mobile and wireless communication have fundamentally invalidated many aspects of these assumptions, for (heterogeneous) wireless access networks and even more so for mobile ad-hoc networks which could be formed between mobile users. This results in challenged networking environments, facing unpredictable and frequently changing connectivity (capacity, error conditions, latency) and node constraints (energy, computing and storage resources), among others. Considering the increasing relevance of wireless communications, we argue that a future Internet architecture should inherently consider challenged networking conditions as a regular case rather than treating them as errors. We identify two important architectural directions and highlight some of challenges to be considered.

I INTRODUCTION

The continuously growing importance of mobile communications and the desire to offer competitive services to mobile users makes operators invest in wireless infrastructure to realize *ubiquitous* and *seamless* connectivity and keep users *always best connected*: The coverage of wireless infrastructure networks (e.g., cellular networks) is expanding and their capacity increasing. Complementary link layer technologies have been developed (e.g., WiMAX) and start seeing deployment. And different networks become more integrated, using multi-access technologies and vertical handover schemes to enable seamless mobile device usage across all of them. This is well-paired with the evolution of mobile devices which are capable of exploiting such offerings increasingly to their full potential.

Nevertheless, experience has shown that mimicking stable and reliable connectivity in the wireless domain may easily fall short of the expectations. Physical constraints on radio propagation and interference inherently limit the reach and stability of wireless communication channels. Commercial and plain practical deployment considerations will likely cause less well connected regions to continue to exist for the foreseeable future. And cost considerations and limited battery lifetime of end systems or intermediate nodes are further factors which may lead to instability and disconnections.[1] Even where full network coverage exists, massive parallel mobile use may still saturate the available capacity in a limited geographic region or seriously impact performance (as we can witness every New Year's Eve around midnight or when attempting to use a Wireless LAN in a large event). That is, communication constraints due to (temporary) failures or (over)load conditions come in as another dimension for discontinuous connectivity. Finally, social conventions or legal restrictions may require communication devices to be turned off. The effective result is that ubiquitous and seamless mobile communication are still far from reality and may not be achievable for some time to come [19].

Even though the communication environment has changed substantially over the past decade with the widespread adoption of mobile communications and the proliferation of powerful mobile devices, the major applications have largely remained the same (with maybe a few adaptations for mobile devices)—and so has the way they operate, i.e., their protocols and their assumptions on communication resources [19]. In fact, a common premise when expanding the reach of the Internet through the wireless communication infrastructure as described above is that *existing* applications must continue to work.

This is in line with the abstraction offered by the Internet Protocol suite, running everything over IP and IP over everything by just providing the necessary mappings to new link layers as they appear. It also seems to be the only viable approach for the short-term where any (possibly commercial) solution must consider incremental deployment and thus cannot assume new applications to appear overnight, to be accepted, and even a fraction of half a billion hosts[2] to change in order to adopt them. Finally, the Internet applications were built on the assumption of an always available infrastructure as shown in figure 1: a) the infrastructure provides global reachability as a communication substrate, b) well-connected servers offer reliable repositories for private and public information, and c) infrastructure services provide rendezvous and mediation functions which may be application-independent (such as DNS or PKI servers) or application-specific (like mail or SIP servers).

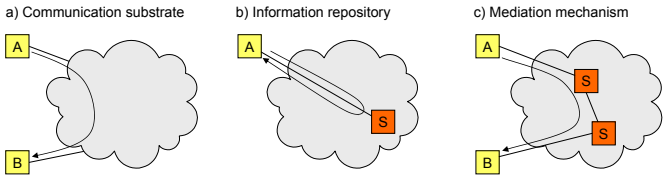

Figure 1: Abstract functions of the Internet infrastructure

However, the need for seamless and ubiquitous communications is highly application-dependent. While there is little doubt that audio and video conversations require packets to basically flow continuously between the communicating endpoints, the situation may be entirely different for other applications: in many cases, applications do not communicate at all

[1] The increasing diversity and number of wireless (mobile) devices and the growing variety of mobile communications uses further add to this.

[2] http://www.isc.org/index.pl?/ops/ds/host-count-history.php

for most of the time and only have occasional packet exchanges once in a while. This applies, for example, to asynchronous applications such as email clients and even to synchronous interactive ones such as web browsers [20]. Thus, many applications would, in principle, not need continuous Internet access.

Yet, today's presumption on providing network-based services seems to equate a user's *Quality of Experience* and the technical side of *Quality of Service* in an access network. While these may in fact be closely related in some cases, they need not be so much in others, as the examples above show. Unfortunately, even if many *applications' semantics* do not depend on continuous Internet access at large, most the underlying application protocols do, at least while carrying out a sequence of operations [19], as we will discuss further in section II.B. This dependency may make mobile users miss communication opportunities under imperfect conditions which they may encounter every day—and may limit the innovation potential for mobile applications due to unnecessary constraints.

In the past, wireless devices for Internet access on the move were in the minority so that point solutions to address the shortcomings of the wireless (last) hop were acceptable. With the number of mobile communication devices having surpassed fixed ones in Japan[3] and mobile Internet usage booming in Europe[4] future Internet protocols have to address networking challenges arising from mobility in a more fundamental way.

In this paper, we elaborate on one particular dimension of the implications arising from user mobility for communications at large and suggest general directions for consideration in a future Internet architecture. Our intention is not (yet) to suggest a specific architecture of any kind. But we use the delay-tolerant networking architecture as a concrete example in support our line of reasoning. In section II, we identify two fundamental issues arising from mobile and wireless networking, which we address in further detail in sections III and IV, respectively. We discuss potential implications on networking paradigms and application protocol design in section V and conclude with a discussion of further (non-technical) challenges in section VI.

II Disconnections, Disruptions, and Delays

In wireless communication environments involving mobility, we can distinguish between *nomadic* and/or *mobile* users. Mobile users move relative to the wireless network infrastructure and each other while communicating, e.g., walking or riding in a vehicle. Nomadic users remain stationary while connected, but access the network from different locations. A special case constitute *nomadic-mobile* users who remain stationary with respect to their immediately surrounding wireless access infrastructure, which in turn moves as a whole, e.g., a mobile user accessing the network via a mobile network on a train or plane. These users may appear alone or in groups and communicate with other users (possibly of the same group) and access resources in or via the Internet.[5]

[3] http://www.tecchannel.de/news/international/442785/

[4] http://networks.silicon.com/mobile/0,39024665,39244960,00.htm

[5] Nevertheless, from section III onwards, we will use the term *mobile node* to generally refer to all of the above in contrast to *fixed* or *well-connected node*.

Users of all classes connect *expressly* to the wireless access network (via an explicit setup function) or are some kind of *always on*, with their devices automatically establishing an access link whenever available (figure 2b). Network access may be directly to a fixed or mobile infrastructure network or, today in rather rare cases, via ad-hoc networks formed between mobile users (2c). (Local) ad-hoc networks could also enable the users to communicate directly with each other, with or without infrastructure access (2d), but this is not well supported.

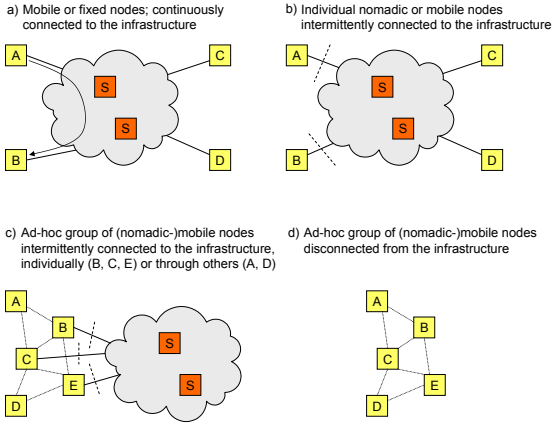

Figure 2: Connectivity variants for mobile nodes

As noted above, user mobility and nomadicity, unstable wireless links, and activities of other users may result in communication challenges. Besides fluctuations in network performance and the obtained QoS (data rate, loss, latency), the most disturbing and from our viewpoint most relevant is loss of connectivity to the access network and/or a local peer:

- We use *disconnections* to refer to the intentional and controlled teardown of network connectivity, e.g., when a user shuts down the mobile device and explicitly disconnects from a wireless network. This is the typical case for nomadic users (unless they suddenly run out of battery) and usually also applies to mobile users in well-covered areas.
- With *disruptions*, we denote the unanticipated loss of effective connectivity, e.g., when leaving wireless coverage, when (continued) service is denied, or when the service quality deteriorates. This rather happens to mobile and nomadic-mobile users and affects nomadic users only in cases of failures or overload.

When orderly disconnecting, the user may have paused or terminated her communication activities prior to shutdown, whereas the user experiencing disruptions may be caught by surprise and his communication applications simply fail. Network access may be regained after some time—established manually by the user or automatically by the mobile device. Losing and regaining connectivity at different points of attachment will usually cause the IP address of the mobile device to change and applications to fail as transport and application protocols often use IP addresses for node identification.

II.A Mitigating Disruptions and Disconnections

In order to deal with the above issues, various approaches have been developed which accept the existence of disconnections and disruptions for mobile users and provide remedies at different layers. *Application-independent* approaches aim at concealing link and network layer outages and other disruptions to some extent from higher layers: from maintaining stable IP addresses [25] to introducing stable endpoint identifiers by splitting identifiers and locators [18] to preventing transport layer disconnections [34, 32, 39, 1, 10] to mitigating the latter by offering disconnection-resilient session layers [21, 16, 30, 9].

These solutions employ variants of incremental fixes and patches to adapt the communication environment as much as possible to what the respective—unchanged—applications need and expect. They often treat the wireless network as a special case of an edge network and attempt to preserve as much as possible of the Internet's end-to-end reliability semantics [8]. However, such an abstraction may easily become leaky in various cases and lead to application protocol failures [8, 19] as we will discuss in the following subsection.

Besides protocol-agnostic approaches, *application-specific* mobility or disconnection support has been designed for various application protocols. Examples include mobility support in SIP (e.g., [27]) as well as adding disconnection tolerance to web browsing [5, 22], to SIP-based multimedia [24], and to secure remote system access and tunneling using *ssh* [13], among others. Further application-specific support may cover automated continuation of communication relationships after connectivity loss; examples include *skype* and many peer-to-peer file sharing systems. The awareness at the application layer helps avoiding the above problem of leaky abstractions.

II.B Impact on the Applications

The above solutions to mitigate the impact of nomadicity and mobility from the applications may be able to conceal temporary connectivity loss and changing addresses from the applications: the loss of connectivity is no longer recognized by the application by other means (such as a disconnection event from the link or the transport layer) and thus the application connections appear to remain intact. However, a lower layer disconnection still prevents data from flowing and protocol operations from completing. From the perspective of the application protocol instances—on the mobile device as well as on its (possibly also mobile) remote peer(s)—the resulting gaps in communication become visible as *delays*. These may last anything from milliseconds to hours or longer until connectivity is (re)gained. Delays may also arise if the network path becomes congested or the remote peer is temporarily (over)loaded and does not respond, among other reasons. All these are treated in the same way because an application may not be able to differentiate between the various cases in the first place.

Unfortunately, an application cannot distinguish delays from failures: a protocol instance cannot predict whether its remote peer will become available or respond shortly, whether it will re-appear after a temporary disconnection, or whether it has crashed. Applications tend to handle this uncertainty with timeouts. If retries, if any, persistently do not succeed within the timeout set, a protocol instance at one layer will declare failure and delegate the responsibility for dealing with this failure to the next higher layer in the stack so that the timeout needs to be dealt with by the application—and ultimately reaches the user as the last resort of "manual" repair. In the end, it is thus up to the user to recreate the communication context for an application and retry suspended or interrupted operations after some time. With this, application layer timeouts are, in the good case, an application designer's attempt to model a user's patience (or: *delay tolerance*) to wait for an action to be carried out.[6] This tolerance may be highly context and/or application-specific and hence can hardly be fixed in some specification.

In principle, from a semantics perspective, many applications and their users may be very delay-tolerant and thus can operate across short or longer disconnections. They do not even require to be connected most of the time. *In practice*, however, it is the application protocols which are not, nor are the lower layer and infrastructure protocols they rely on.

In addition, today's application protocols and architectures place heavy dependencies on the availability of the Internet infrastructure: for mediating communications via rendezvous points, as universal information repository, for resolving names, identifiers, and addresses, for validating the authenticity of peers, among others. To access this infrastructure, today's application implementations expect to be well-connected: they are designed to act instantaneously upon a user's request and expect responses virtually immediately. Infrastructureless operation is usually not foreseen.

II.C Two Architectural Issues

An Internet architecture should support communications across the entire range of mobility and connectivity scenarios depicted in figure 2. In this respect, we can essentially distill two key architectural concerns from the preceding discussion for proper application operation in challenged mobile environments, both of which arise from intermittent connectivity:

- *Infrastructure access and ad-hoc reachability*

  A future Internet will likely be built around some notion of a core network infrastructure. We assume that direct and continuous access will remain limited, at least in various non-negligible scenarios. Therefore, the basic communication architecture should inherently support delay tolerance in reachability and communication—in addition to instant end-to-end packet switching for the well-connected case—as part of regular operation and thus enable delay-tolerant applications to continue operation in challenging conditions. This should apply to accessing and communicating via the infrastructure as well as to reaching peers in an ad-hoc environment.

- *Communication mediation*

  Mobile applications often depend on infrastructure networks and services in many ways which limits their capability of operating autonomously: most importantly, un-

[6] In other cases, the timeout may also be just a "technically" motivated, convenient for the implementation, or an otherwise random choice.

reachability of (nodes in) the infrastructure may delay or prevent communication with peers even if the latter would be in direct reach. This happens if nodes mediating the interaction between peers are located in the infrastructure. Such delays can be circumvented by taking the infrastructure out of the loop: by removing the dependency on infrastructure access and/or on individual domains or nodes. Ideally, a novel approach to mediating communications would also support the case of a disconnected group of mobile users just trying to communicate with one another.

Both these aspects are different facets of *robustness* in communication systems. Since we do not have to distinguish between disconnections, disruptions, and delay and their respective sources, supporting delay tolerance as an inherent networking property and reducing the dependencies on the infrastructure appear also as appropriate mechanisms to deal with failures—and hence could be useful even if we had perfect access networks with all nodes permanently connected. We discuss these aspects in the following two sections, respectively, and then review the impact on application protocol design.

III Towards Delay Tolerance

We have discussed above that increasing delay tolerance of applications may improve their capability of operating in challenged mobile environments and thus improve the *user-perceived performance* compared to those cases where the applications simply report failures. Nevertheless, given the dominant role of the Internet infrastructure (now and in the future) to reach remote peers and engage in conversations or access information, maximizing access to the infrastructure remains a prerequisite for good application performance. As discussed above, traditional Internet protocols do not support mobile and nomadic users well when they experience of intermittent connectivity because the end-to-end connection—to another (fixed or mobile) node—breaks.

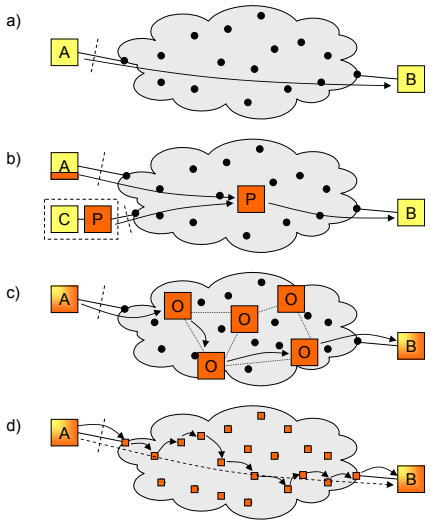

Figure 3: Incremental introduction of delay tolerance

The different approaches in support of disconnection/disruption tolerance we briefly touched upon in section II.A may mitigate the impact of short-term disconnections: some of them operate end-to-end as depicted in figure 3a), others introduce intermediaries (or: proxies, 'P') as shown in figure 3b) which may operate as separate entities or as integral part of the endpoints. The proxies must be configured and/or discovered and they introduce additional points of failure as all communication needs to pass through the intermediaries (of which there may be many and different ones for different mobile nodes) pinning down a particular path. While some approaches such as OCMP [30] allow for state transfer between different intermediaries, this is a mechanism for performance improvement rather than for failure handling. This could be expanded further to creating an overlay on top of the existing Internet infrastructure (see figure 3c); however, the basic shortcomings noted above would remain.

The disconnection tolerance mechanisms discussed above fall short in three areas:

1. Unless application-specific, they are mostly tied to a particular transport protocol (typically TCP) and its connection-oriented semantics. If intermediaries are used their introduction violate the Internet end-to-end principle, which leads to a loss of robustness. This loss becomes even more pronounced if the intermediaries add further application-specific support.

2. They rely on instant access to the infrastructure from a mobile node to reach their respective peers, intermediaries, or overlay nodes and assume end-to-end connectivity to these. This imposes the major limitation that two nodes not connected to the Internet and not sharing an instant end-to-end path remain disconnected (=unreachable) from an application perspective. Hence, the reliance on the infrastructure makes these mechanisms unsuitable for ad-hoc environments and areas without end-to-end connectivity and thus fundamentally limits their applicability.

3. None of these mechanisms help with longer gaps in connectivity unless the application protocols are suitably adapted—which also applies to the security mechanisms they employ [19]. Note that this is not a fault of the present approaches, but simply suggests that delay tolerance needs to be handled in a more encompassing fashion.

Nevertheless, while insufficient by themselves, appropriate delay tolerance at the lower layers can increase connectivity and thereby improve overall communication performance. A prerequisite, however, is that these mechanisms truly improve connectivity and not just mask disconnections and disruptions. Since delay tolerance is anyway up to the application protocols, we can focus on (access to) the infrastructure for now.

The major issue leading to disconnection and disruptions is the "instant nature" of IP-based communications which, again, stems from the notion of an always connected environment: if a packet to send cannot be forwarded immediately—e.g., because there is no route available at this point—it is dropped and,

optionally, an (ICMP) error message is generated. Routing protocol implementations for mobile ad-hoc networks (MANETs) relax this slightly in that at least the reactive ones attempt to determine a route on demand. However, if this route search fails—and it will if the source and the destination reside in different network partitions—the result is the same.

Delay-tolerant networking [8] suggests to eliminate this need for instant connectivity: instead, packets may be stored for an extended period of time, from milliseconds to hours or days or more. The nodes may move and thus physically carry stored packets around. Storage and carriage effectively increase the flexibility of a communication system in finding routes to a destination by allowing considering *future* forwarding opportunities in addition to *instant* ones. This comes at the cost that error reporting about non-available paths and non-existent targets may be delayed. And, of course, the observed round-trip times and jitter may increase significantly.

We note that just because packets *may* be delayed to improve connectivity in some circumstances and for some applications, this *does not* mean that all applications have to be subject to potentially increased delays. A *type-of-service* field could be used by applications to indicate whether or not they require the instant packet delivery of the present Internet so that real-time applications can continue to work.

An architecture supporting delay-tolerant in addition to instant packet forwarding should be capable of addressing the disconnected cases depicted in figure 2a–c), in particular also supporting the indirect infrastructure access by nodes A and D in figure 2c) even if no instant path to the infrastructure exists. Such an architecture could be introduced based upon proxies and expanded to become an overlay (figure 3b and c).

The key difference to aforementioned mitigation approaches is that the packet remains the basic transmission and it preserves its properties of independent and uncorrelated forwarding which makes IP robust in connected environments. This inherently avoids the need for flow state (which is costly and complicates failover) and thus could make the necessary functionality lightweight enough to allow pushing delay-tolerant forwarding from an overlay into the elementary routing and forwarding fabric of the infrastructure. This is shown in figure 3d) where, ultimately, end-to-end operation gets restored as all networks, yet with increased robustness.[7]

Integrating delay tolerance as such an elementary function eliminates the need for well-known control points in the infrastructure and thus enables supporting delay-tolerant ad-hoc communications as shown in figure 2d), thus blurring the distinction between an infrastructure network and mobile nodes. One prominent example is the comprehensive architecture for delay-tolerant networking[4, 29] defined by the IRTF DTNRG[8] which supports internetworking in highly diverse—regular and challenged—environments, for which various types of mobility supported have been explored (e.g., [31]).

[7]Depending on the details of such an architecture, the core network nodes may not see much of a difference anyway as they are usually well connected and may thus always forward packets instantaneously whereas more responsibility may reside on nodes closer to the edges of the network.

[8]http://www.dtnrg.org/

An important aspect in support of delay tolerance is that frequent interactions between peers may not be possible in a short time frame. The de-facto IP MTU size of 1500 bytes may thus be too limiting, particularly when used in conjunction with transport protocols which perform ramp-up as a function of the RTT. The DTN architecture [4] uses the concept of (in theory) arbitrarily sized messages as basic unit of data exchange. The increased flexibility of transmitting larger information units at once allows for expressing semantically richer operations [19], but raises issues for congestion control in the network and protecting resources in hosts. While arbitrarily sized messages may not be feasible for a globally scoped Internet, revisiting the—historically and technically motivated—choice of the size of information units for future internetworks may nevertheless be a worthwhile exercise to better support application interaction in general. Any such activity will need to address the issue of identifying, dealing with, and reducing unwanted traffic to prevent larger information units from being exploited for denial-of-service and other attacks or misuses.

What we have sketched in figure 3 above could also become a suitable migration path to introduce delay tolerance functionality into the Internet. A starting point are simple proxy-based approaches which are particularly suitable for the near term—one such approach is specifically pursued by the EC FP7 project CHIANTI[9]. When choosing the "the right" abstraction for delay tolerance support, a proxy-based architecture could evolve into a routing overlay, in which ingress and egress nodes initially perform the translation when talking to conventional Internet nodes and applications. As the new architecture would gain adoption over time, the functionality could migrate into the routers and the overlay would slowly disappear.

IV Enabling Infrastructureless Operation

While adding delay tolerance as a basic element to the communication substrate increases reachability, the dependency on the infrastructure as means for mediating communications and information repository remains. Application-independent services such as DHCP, DNS, and PKI are infrastructure-based and so are application-specific intermediaries such as mail, SIP, or web servers. Also, peer-to-peer systems both for storing information and mediating communications (e.g., [33, 2, 37]) mostly rely on well-connected overlay nodes and partly on bootstrap servers in the Internet.

This dependency limits communications in the scenarios of figure 2c) and d): particularly if two nodes (e.g., A and D) would be in easy reach of each other, they may still not be able to talk (instantly) because the mediating function is (temporarily) unreachable. Even if the infrastructure protocols are delay-tolerant and allow accessing infrastructure services in a more robust fashion, the temporary unavailability will lead to delays which harm the perceived quality of experience. Some (application-specific) solutions have been proposed for peer discovery in connected mobile ad-hoc networks (e.g., [15]) and content sharing in DTNs [38, 28, 12].

[9]CHIANTI: Challenged Internet Access Network Technology Infrastructure, http://www.chianti-ict.org/

To address this issue in a general fashion, indirections requiring access to (certain nodes in) the infrastructure should be avoided as much as possible. Rather than creating application-specific solutions, a general purpose mediation substrate should be provided. This requires a proper split between generic functionality to be included in the communication infrastructure and all mobile nodes and specific functionality left to higher (e.g., transport and application) layer protocols.

In today's Internet, the most common "infrastructure" services used include: 1) (dynamic) address allocation functions (e.g., DHCP or PPP); 2) mapping functions (registration, resolution) of one address into another (e.g., using DNS or SIP servers) as well as rendezvous and relaying (e.g., via mobile IP home agents, HIP rendezvous servers, mail and SIP servers); and 3) validation functions (e.g., by certification authorities).

1) Address allocation: Instead of using dynamically allocated addresses (e.g., via DHCP), persistent node identifiers (such as a HIP *Host Identity, HI* [18]), or a uniquely constructed DTN *Endpoint Identifier, EID* [4] could be used. This is applicable to mobile as well as well-connected nodes.

2) Mapping, rendezvous and relaying: These mediating functions may operate on anything between static (such as DNS names) and highly dynamic bindings (such as mobile IP care-of addresses). At the potentially disconnected edges of the network, the respective functionality has to be supported by arbitrary nodes, without infrastructure access and without end-to-end paths between nodes: to reach other mobile nodes at the (same) edge as well as to reach well-connected nodes. This rules out building up and maintaining consistent databases in which bindings could be stored. Instead, individual nodes have to operate on partial knowledge which suggests not to attempt a mapping in the first place, but to follow the concept of *late binding*, deferring the resolution as long as possible.

Rather than performing an initial mapping (e.g., of a DNS name to an IP address, a Host Identity, a SIP or mailto URI to a server name), one approach could be to allow forwarding to take place based upon the identifier which is supposed to be looked up and/or an algorithmically derived one such as the *Host Identity Tag, HIT*. If such forwarding is not possible or an algorithmic translation is not available, a topologically limited search (e.g., using a receiver-filtered broadcast) might be used to reach a potentially close by node or a node connected to the infrastructure.[10] This would entirely avoid the resolution process at the edge and enables both localized communication among mobile nodes as well as reaching the infrastructure with limited knowledge.[11]

In the well-connected infrastructure, the traditional distribution of mediation responsibility could (but need not) be preserved. When offering a generic mediation mechanism, all nodes (including the mobile and nomadic ones) need to register with the infrastructure so that their mutual reachability is ensured (provided that they are reachable at all). In contrast to the edge, state synchronization protocols could be applied in order create, maintain, and replicate the respective bindings. To support identifier-based routing (in addition or instead of locator-based routing), a variant of "default routes" could be used to further defer the resolution process if no (current) local knowledge is available. This would allow keeping the binding resolution probabilistic and limit (the need for) the distribution and storage of the bindings. Finally, the roles of dedicated intermediaries (e.g., SIP servers) could be preserved: they can act as a backup for message delivery or a last resort for lookups, so that traditional mapping approaches could be preserved where necessary or desirable. In either case, nodes acting as "gateways" to the edges could perform the necessary additional translation on behalf of the mobile node that originated a message.

As a result, mobile nodes should be opportunistically reachable for co-located nodes in topologically limited regions of the edge by means of local search, realized by reactive, proactive, or hybrid mechanisms. Mobile nodes could reach other mobile or fixed nodes via delay-tolerant and identifier-based forwarding towards the infrastructure. And, through registrations, the mobile nodes remain reachable from/via the infrastructure as long as delay-tolerant paths exist.

3) Validation: For the purpose of validating information about a node, new concepts are required if the dependency on the infrastructure shall be reduced. Entirely eliminating this dependency does not seem possible: if, for example, information about revoked certificates is only reliably available via the infrastructure, infrastructureless operation is naturally limited. Yet, it is worthwhile to explore mechanisms that allow reducing the *instant* dependency on the infrastructure and make validation processes delay-tolerant where acceptable to the application. For example, *Identity-based cryptography* could be used to transmit confidential messages from a disconnected node to another node and validation of incoming messages may be deferred or be carried out based upon cached information. If nodes are expected to be regularly (e.g., once per day) in contact with the infrastructure, suitable validation and key exchange protocols can be designed to inquire the infrastructure and update information on the mobile nodes as needed.

For all three aspects, the requirements on correctness and freshness of the information (relative to its change/update frequency) will ultimately determine how much disconnected and delay-tolerant operation is feasible which, in turn, will depend on the specific application.

Distributing the responsibility for mediation and thus enabling direct communication between peers, however, raises several further issues besides timely information availability. Most importantly, applications relying on intermediaries in the infrastructure provide institutions and/or users with a (single) point of control: intermediaries may implement access policies (e.g., who is allowed to obtain an address mapping), filtering (e.g., for spam), and other value-added functions (e.g., time-of-day or caller-based handling of incoming calls). Such rule sets may be confidential (or require access to confidential information bases) so that generic nodes may not be considered sufficiently trustworthy to execute them; or they may just be too complex to be described or too dynamic to be distributed

[10] Note that we make the—in most areas of industrialized countries not unreasonable—assumption that disconnected ad-hoc networks will grow to a limited size only before local infrastructure gets deployed (e.g., in a train).

[11] Note that, while using potentially longer identifiers may increase the per-message overhead, this may be outweighed by the use of larger messages.

across the infrastructure. While all of this may limit the disconnected operation of mobile nodes, they can implement some of the functions on the receiver side: For example, they may simply deny accepting or reacting to incoming messages if they are not permissible according to their policy; or they may require the indirection through their responsible intermediaries which can tag a message as "validated". In any case, as noted for item 3) above, enabling disconnected operation may come at a *cost* or *risk*, to be evaluated specifically for each application/user.

Finally, application-specific support could be considered, e.g., store/cache contents dynamically in arbitrary nodes as suggested for web applications [23, 11]. Functions may range from simple caching to content adaptation and increase the availability or accessibility of information stored in the infrastructure [23, 26]. How far application-specific optimizations should reach and how much (if any) application awareness should be supported in the infrastructure deserves further investigation. In any case, they should be purely probabilistic and minimize the interaction between application and elementary forwarding/resolution functions in order to keep the network and application functions well separated and avoid feature interactions (as discussed in the context of *Active Networks* [36]).

V Application and Communication Paradigms

We have repeatedly pointed out above that those application protocols and application implementations basically capable of tolerating delays require adaptation to operate in a delay-tolerant environment. This means particularly to support asynchronous communications inside the application protocol and at the layers used below. Design recommendations (e.g., [8, 19]) include that application protocols should minimize the number of end-to-end interactions needed to perform an operation ("chattiness") and hence encode their protocol operations self-contained in messages [6]; untangle application from lower layer (transport) state and identifiers; minimize the dependency on infrastructure nodes and be flexible in when to access them; avoid asymmetric protocols requiring translating intermediaries (such as mail servers) to enable peer-to-peer besides client-server style of operation; be explicit about the use of or need for intermediaries; and realize the intended end-to-end semantics in all respects at the application layer.

Applications which are inherently not delay-tolerant could largely continue working as before. However, they may improve their overall performance and capability of operating in a disconnected environment by at least leveraging the delay-tolerant mediation mechanisms.

Finally, the traditional notion of an end-to-end design of transport and application protocols in the Internet places the burden of observing and adapting to the dynamic network conditions on the transport and application instances on the endpoints, with some optional network support for QoS and congestion control. Similar considerations could be applied to delay and disconnection tolerance in future application protocol design: adaptive applications should tune their behavior to match delay and degree of disconnection and change their way of operation according to the observed characteristics.

Networking paradigms and APIs

A key reason for most of today's applications failures in challenged environments is that their assumptions about the underlying connectivity fail. These assumptions are reinforced by transport layer services and the BSD *socket* API offering an abstraction from the underlying network which is essentially binary: an operation works (and then it does so from the beginning to the end) or it fails without recovery. As we noted above, challenged networks may lead to partial failures which are not dealt with appropriately by this abstraction and thus *leak* to the application layer. As a result, the only remedy an application (protocol) designer has is to ignore the offered abstractions and realize all the necessary functionality again. Given the complexity of the matter and the aforementioned desire to offer a common infrastructure, a new abstraction for the underlying communication services appears necessary in order to support the required shift in application protocol design.

As suggested above, message-based communication may be one suitable option as a minimal enhancement in support of delay tolerance. This could be augmented by a delay-tolerant transport or session layer functionality—for which the notion of an end-to-end "connection" likely needs revisiting—and be exported via an asynchronous API. System implementations might provide direct buffer control [17] and offer hints about lower layer connectivity for each interface, possibly extending to (parts of) the path to make applications aware of their (immediate) surroundings—related to or independent of a particular communication relationship. An API would further provide support for late binding and offer access to the application-independent mediation functions.

The present endpoint-oriented addressing scheme in the Internet (with some support for multicasting and anycasting) is focused on communications to and from individual nodes, thus making access to information dependent on knowing about its location(s). While delay tolerance could be realized based upon such addressing, recent research has suggested that this node-centric communication paradigm may be inappropriate given today's often content-centric applications [3, 14]. This is also reflected in the above generalization of the mediation function: instead of accessing a particular mediation node, only the respective rendezvous information is of interest, and this could be obtained in numerous ways and from multiple nodes.

Suggestions were made to apply the *publish/subscribe* paradigm which may be much more natural particularly to information retrieval than node-based communications and offer a corresponding API [7, 35]. Such concepts appear suitable for the mediation function and could also be applied as a basis for data communications. Whether they should be applied to all applications [35] or only to the subset of delay-tolerant ones (leaving the real-time applications alone)—in short: which kind of mix of communication paradigms appears suitable for a future Internet—is an important field for further investigation.

VI Conclusion

In this paper, we have discussed the need for delay tolerance in a future Internet architecture and identified (in addition to

the applications) two areas to be addressed in future research: delay-tolerant forwarding and communication mediation. A key characteristic is that these are enablers for better exploiting the capabilities of mobile devices. Expanding the reach of a future Internet and enhancing the end-to-end design principle to disconnected mobile nodes, allows for broader innovation and service creation at the edges in the mobile endpoints, with and without (instant) infrastructure support. While both are described as elementary building blocks of a clean-slate design, similar functionality might also be incrementally introduced into the existing Internet.

VII Acknowledgments

The author would like to thank Carsten Bormann for comments and suggestions on this paper. This research is supported by Teknologiateollisuus ry and the EC FP7 project CHIANTI.

References

[1] Kevin Brown and Suresh Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM Computer Communication Review*, 27(5), 1997.

[2] David Bryan, Philip Matthews, Eunsoo Shim, and Dean Willis. Concepts and Terminology for Peer to Peer SIP. Internet Draft draft-ietf-p2psip-concepts-01, Work in progress, November 2007.

[3] Antonio Carzaniga and Alexander L. Wolf. Forwarding in a Content-Based Network. In *Proceedings of ACM SIGCOMM*, 2003.

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H.Weiss. Delay-Tolerant Network Architecture. RFC 4838, 2007.

[5] Henry Chang, Carl Tait, Norman Cohen, Moshe Shapiro, Steve Mastrianni, Rick Floyd, Barron Housel, and David Lindquist. Web Browsing in a Wireless Environment: Disconnected and Asynchronous Operation in ARTour Web Express. In *Proceedings of ACM Mobicom 97, Budapest, Hungary*, pages 260–269, 1997.

[6] David D. Clark and David L. Tennenhouse. Architectural Considerations for a new Generation of Protocols. In *Proceedings of ACM SIGCOMM Symposium on Communication Architectures and Protocols*, 1990.

[7] Michael Demmer, Kevin Fall, Teemu Koponen, and Scott Shenker. Towards a Modern Communications API. In *Proceedings of the ACM SIGCOMM HotNets VI workshop*, 2007.

[8] Kevin Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of ACM SIGCOMM 2003*, August 2003.

[9] Bryan Ford. Structured Streams: a New Transport Abstraction. Proceedings of ACM SIGCOMM, August 2007.

[10] Tom Goff, James Moronski, and D.S. Phatak. Freeze-TCP: A True End-to-end TCP Enhancement Mechanism for Mobile Environments. In *Proceedings of IEEE Infocom*, 2000.

[11] Pan Hui, Jérémie Leguay, Jon Crowcroft, James Scott, Timur Friedman, and Vania Conan. Osmosis in Pocket Switched Networks. In *ChinaCom*, 2006.

[12] Gunnar Karlsson, Vincent Lenders, and Martin May. Delay-tolerant Broadcasting. In *ACM SIGCOMM Workshop on Challenged Networks (CHANTS)*, 2006.

[13] T. Koponen, P. Eronen, and M. Särelä. Resilient Connections for SSH and TLS. *Proceedings of the Annual Technical Conference on USENIX'06 Annual Technical Conference*, 2006.

[14] Teemu Koponen, Mohit Chawla, Byung-Gon Chung, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A Data-Oriented (and Beyond) Networking Architecture. In *Proceedings of ACM SIGCOMM*, 2007.

[15] Simone Leggio. *A Decentralized Session Management Framework for Heterogeneous Ad-Hoc and Fixed Networks*. PhD thesis, University of Helsinki, Finland, 2007.

[16] Yun Mao, Björn Knutsson, Honghui Lu, and Jonathan Smith. DHARMA: Distributed Home Agent for Robust Mobile Access. In *Proceedings of the IEEE Infocom*, 2005.

[17] Jeffrey Mogul, Lawrence Brakmo, David E. Lowell, Dinesh Subhraveti, and Justin Moore. Unveiling the Transport. *ACM SIGCOMM Computer Communications Review*, 34(1):99–105, January 2004.

[18] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas R. Henderson. Host Identity Protocol. Internet Draft draft-ietf-hip-base-10.txt, Work in progress, October 2007.

[19] Jörg Ott. Application Protocol Design Considerations for a Mobile Internet. In *Proceedings of the 1st ACM MobiArch Workshop*, 2006.

[20] Jörg Ott and Dirk Kutscher. Why Seamless? Towards Exploiting WLAN-based Intermittent Connectivity on the Road. In *Proceedings of the TERENA Networking Conference, TNC 2004, Rhodes*, June 2004.

[21] Jörg Ott and Dirk Kutscher. A Disconnection-Tolerant Transport for Drive-thru Internet Environments. In *Proceedings of IEEE Infocom*, 2005.

[22] Jörg Ott and Dirk Kutscher. Bundling the Web: HTTP over DTN. In *Proceedings of WNEPT 2006*, August 2006.

[23] Jörg Ott and Mikko Pitkänen. DTN-based Content Storage and Retrieval. In *The First IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC)*, June 2007.

[24] Jörg Ott and Lu Xiaojun. Disconnection Tolerance for SIP-based Real-time Media Sessions. Proceedings of the International Conference on Mobile Ubiquitous Multimedia (MUM), December 2007.

[25] Charles E. Perkins. (ed.) IP Mobility Support for IPv4. RFC 3344, 2002.

[26] Mikko Pitkänen and Jörg Ott. Enabling Opportunistic Storage for Mobile DTNs. *Journal on Pervasive and Mobile Computing*, In press.

[27] Henning Schulzrinne and Elin Wedlund. Application-Layer Mobility Using SIP. *ACM Mobile Computing and Communications Review*, 4(3), July 2000.

[28] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. Haggle: A Networking Architecture Designed Around Mobile Users. In *Proceedings of IFIP WONS*, Les Ménuires, France, January 2006.

[29] Keith Scott and Scott Burleigh. Bundle Protocol Specification. RFC 5050, November 2007.

[30] A. Seth, S. Bhattacharyya, and S. Keshav. Application Support for Opportunistic Communication on Multiple Wireless Networks. Manuscript available from http://blizzard.cs.uwaterloo.ca/ keshav/home/Papers/data/05/ocmp.pdf, November 2005.

[31] A. Seth, P. Darragh, S. Liang, Y. Lin, and S. Keshav. An Architecture for Tetherless Communication. Available at http://blizzard.cs.uwaterloo.ca/ keshav/home/Papers/data/05/tca.pdf, July 2005.

[32] Alex C. Snoeren and Hari Balakrishnan. An End-to-End Approach to Host Mobility. In *Proceedings of Mobicom*, 2000.

[33] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. 2001.

[34] Florin Sultan, Kiran Srinivasan, Deepa Iyer, and Liviu Iftode. Migratory TCP: Highly Available Internet Services Using Connection Migration. Technical Report DCS-TR-264, Rutgers University, December 2001.

[35] Mikko Särelä, Teemu Rinta-aho, and Sasu Tarkoma. RTFM: Publish/Subscribe Internetworking Architecture. In *Proceedings of the IST Mobile and Wireless Communication Summit*, June 2008.

[36] David L. Tennenhouse and David J. Wetherall. Towards an Active Network Architecture. In *Proceedings of ACM SIGCOMM*, 1996.

[37] Wesley W. Terpstra, Jussi Kangasharju, Christof Leng, and Alejandro P. Buchmann. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In *Proceedings of ACM SIGCOMM*, 2007.

[38] Liangzhong Yin and Guohong Cao. Supporting Cooperative Caching in Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, January 2006.

[39] Victor C. Zandy and Barton P. Miller. Reliable Network Connections. In *Proceedings of ACM MobiCom*, 2002.