Autonomous Transport Protocols for Content-based Networks *

Florin Sultan†, Aniruddha Bohra†, and Liviu Iftode‡
† *Department of Computer Science*
Rutgers University, Piscataway, NJ 08854-8019
{*sultan, bohra*}*@cs.rutgers.edu*
‡ *Department of Computer Science*
University of Maryland, College Park, MD 20742
iftode@cs.umd.edu

Abstract

In this paper, we argue for a new approach to reliable transport protocols in which connections are established between endpoints in a generic content-based network. In such a network, endpoints are named by content and mapped to host addresses using a content-based location and routing scheme.

We identify two key design principles for autonomous protocols: (i) an endpoint name space decoupled from the network host address space and (ii) relaxed end-to-end semantics, which allows intermediate nodes in the network to be dynamically involved in the transport. We use these principles to propose a transport protocol with dynamic connection splitting for reliable communication in content-based networks.

Autonomous transport layer protocols decouple communication endpoints from the network addressing and routing, therefore they are particularly suitable for handling endpoint mobility. They can be deployed in a wide range of architectures, from networks of embedded systems or sensors to overlay networks over the Internet.

1 Introduction

There is an increasing gap between the traditional transport protocols designed for the Internet-based communication and the characteristics of emerging nonconventional network architectures and networking applications. Recent technology advances allow building networks of small devices with powerful computing and communication capabilities. Notable examples which have stimulated research in the recent past are ad hoc networks of mobile computing devices, networks of embedded systems, and sensor networks [13, 10, 16] Another example is the recent explosion in peer-to-peer computing over the Internet leading to the development of overlay networks with discovery and routing infrastructures over IP [21, 17, 25, 18].

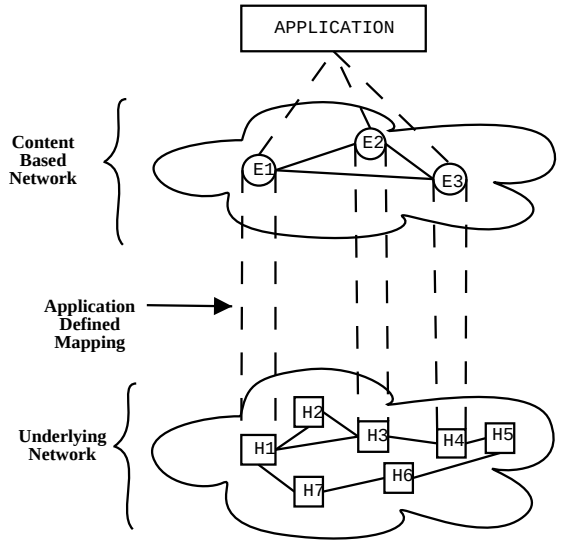

Figure 1: *Mapping of a content-based network into a host network.*

In the following, we use the term *content-based network* (CBN) to designate an endpoint name space E (e.g., defined by properties, attributes or content), in which location and routing operations are defined. A content-based network is mapped into an underlying host network H by the application that builds the CBN (Figure 1). This mapping can be dynamic as content can move or disappear from a host. Examples of CBNs and their applications are: *(i)* E can be a set of geographic regions and H a set of cars (with network connectivity) crossing these regions while moving on a highway. Cars passing through a region receive information, e.g., traffic advisories. *(ii)* E is a set of unique user identifiers in an application and H a set of hosts in the Internet. A user may move between hosts while data is continuously being delivered to him.

Traditional transport protocols originally designed for the Internet are not suitable for CBNs due to the

*This work is supported by the National Science Foundation under the ITR Grant Number ANI-0121416.

mismatch between the logical naming of the communication endpoints specific to CBNs and the endpoint naming schemes that these protocols support. The former define application-specific names for endpoints, while the latter use host addresses. As a result, a traditional transport protocol like TCP [15] reduces the connection between application-defined endpoints to a connection between the hosts on which the application executes. This works if the endpoint-to-host binding is permanent or if at least does not change during connection lifetime. In case the mapping changes, for example due to the host moving and attaching to another point in the network, the connection is hard to maintain. A large body of research exists that addresses this problem by adapting existing transport protocols to host mobility [2, 5, 24, 19].

We advocate a transport protocol which preserves and exploits the independence of the endpoint name space, as provided by the CBN, from the host address name space. We call the class of such protocols *autonomous transport protocols* (ATP). In an autonomous protocol, mobility can be trivially supported as the connection continues to exist between the attribute-named endpoints regardless of the hosts on which they reside.

There are two important characteristics of ATPs over a CBN. First, in order to ensure autonomy of the transport protocol from the network address space, routing must be performed exclusively in the content-based network. If routing were to be done towards *hosts* where endpoints reside, this means that a specific destination host has to be located in the underlying network at the time a connection is set up. This would result in the connection being bound to specific hosts throughout its lifetime.

Second, nodes in a CBN may act as both routers and hosts where application executes. This enables nodes in the network to participate in the transport protocol, e.g., nodes can temporarily buffer data and become responsible for its delivery. The mechanism we propose to support this behavior is to *dynamically* split a transport connection. Because the split takes place in the endpoint space of the CBN, the temporary intermediate buffers can be named using the same content-based addressing scheme. In this way, splitting a connection naturally amounts to nodes in the network occasionally piercing into the transport layer and assuming transport functionality for that connection.

Connection split in a CBN provides a clean solution to handling of endpoint mobility and to localized recovery from failures. We believe that end-to-end mechanisms at the transport layer fail to capture and exploit the essence of CBNs and do not properly address their intended use and applications. In contrast, the relaxed end-to-end semantics introduced by connection splitting allows the protocol to use knowledge about the CBN for performance improvements. For example, a split for localized recovery might be useful where end-to-end mechanisms for error recovery would be too expensive, e.g., by avoiding a source-based retransmission that consumes resources at intermediate nodes (power, bandwidth). Several mobile TCP schemes [2, 24], have also suggested a similar approach, trading end-to-end semantics for better performance over a hybrid wired/wireless network. However, they are limited to one permanent, static split, at the boundary of the wired network.

The remainder of the paper is structured as follows. We start with two motivating examples in Section 2. Section 3 presents the ATP model. Section 4 presents design guidelines for ATP protocols. Section 5 summarizes related work. Finally, Section 6 concludes the paper.

2 Examples

Consider the following example of a content-based network where there is a need for reliable transport protocols: Suppose that each user in a corporation is assigned a badge, which becomes her identity. This identity is unique throughout the corporation and allows her to access a variety of computing resources ranging from her desktop to a smart board. Suppose two users, let us call them Alice and Betty, start a conversation. To initiate a connection, Alice starts her chat application and connects to Betty. Note that both Alice and Betty are unique identifiers in the endpoint name space E of a CBN. The computers they are currently using have a host identifier which is mapped to the endpoints (user identities) by the chat application.

After establishing a connection, Alice and Betty exchange data reliably. Suppose that after a while Betty needs to move and starts using a handheld device which, through an interaction with her badge, automatically acquires her identity in the CBN. Since the connection was set up between users Alice and Betty, not between the machines they are using, it is not broken. The CBN rediscovers Betty (now on her handheld) and resumes the transfer of data. The communication application discovers and sets up the routing paths for the endpoints in the CBN, this time to different hosts in the underlying network.

The above example illustrates a simple scenario with a CBN. This network decouples its endpoint name space from the underlying host name space. The application provides the mapping between the two name spaces as well as routing and lookup in the CBN.

Another scenario for reliable transport protocols is in a geographical CBN over an ad hoc network of cars on a highway. In this case, the CBN defines endpoints as

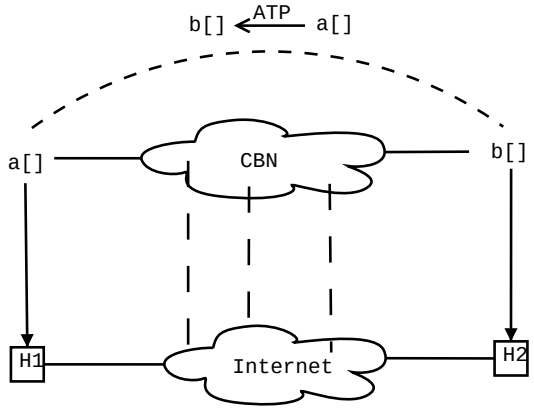

Figure 2: *An ATP connection in a CBN.*

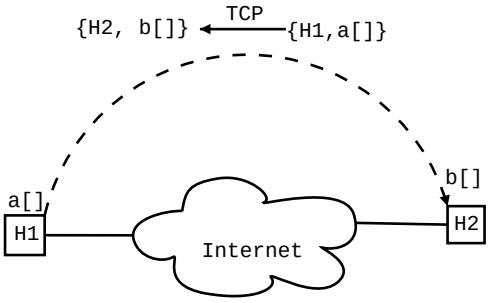

Figure 3: *A TCP connection in an IP network.*

milestones on the highway, which can be subsequently mapped to cars driving close to that milestone.

In such a network, we can define an application that allows cars to collaborate in order to exchange a continuous stream of updates on road conditions. For instance, a car A located at mile X may want to find out what are the road conditions at mile Y. To do this, its collaborative application will request a connection with mile Y. A car B in that target area may accept the connection. Subsequently, data acquired by car B will flow to car A. Once the transfer is completed, car A can hand over the already established connection to another car C which approaches mile X. Car B can do the same for a car D close to mile Y. Now, car C and D can execute the same application (or continue it), without paying the cost of connection establishment.

In this way, what we achieve is in fact a connection between mile X and mile Y that can exist as long as there are hosts interested in this connection around mile X and mile Y.

3 The ATP Model

To understand the ATP, we use a simple model to describe the endpoint name space of a CBN and the data transfers between endpoints. In essence, we use

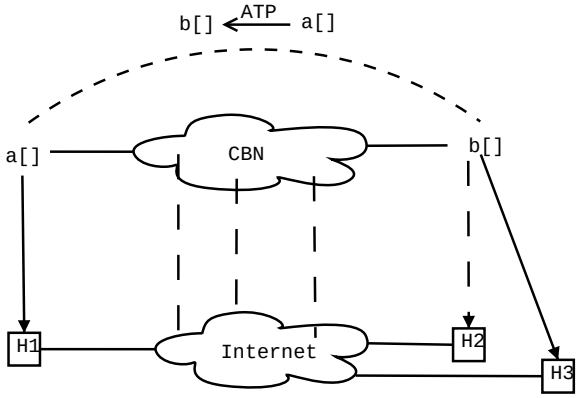

Figure 4: *ATP enables endpoint migration across nodes during connection lifetime.*

array variables to describe a generic endpoint name space, and assignments between these variables to describe data transfers. In this model, arrays $a[]$ and $b[]$ designate the content itself, and the array names a, b are the endpoint names in the CBN. For example, in the car example in Section 2 the endpoint a represents information available at Mile Y (e.g., a continuous stream of updates on traffic conditions, or a video stream of the highway captured locally) while b is the target endpoint for this information represented by Mile X. At different moments in time, a and b, respectively, may be mapped to different cars moving through the two locations.

In the CBN, an ATP is responsible for reliable delivery of data between endpoints, which can be expressed as an assignment $b[] = a[]$ (for simplicity, throughout this section we consider unidirectional transport transfers). In other words, the ATP performs $b[0] = a[0]$, $b[1] = a[1]$, etc., regardless of the location of a and b. Figure 2 shows a transfer over a CBN built as an Internet overlay.

In the following we assume that application-defined endpoints can be hosted anywhere in the network and are unique. In particular, if $b[]$ is hosted by node H_2 and $a[]$ by node H_1 and if the endpoints never move, the ATP reduces to a regular TCP to establish a connection between H_1 and H_2 and transfer a into b (Figure 3). However, because our ATP model defines communication in terms of content names and not network hosts, the content can move (change hosts in our example) during connection lifetime as the connection endpoints do not change their identity (Figure 4).

3.1 Endpoint-to-Host Mapping

To perform reliable communication over a CBN, an ATP will eventually use the underlying network. In order to do so the ATP must map endpoints defined

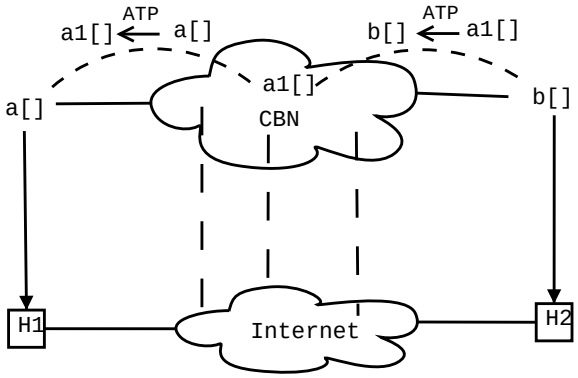

Figure 5: *ATP can dynamically split a connection at multiple endpoints inside the CBN.*

by the application into hosts where they reside using the CBN services for location and routing.

To establish a connection between a and b the ATP will first register the endpoints a and b with the CBN. This allows the endpoints to be located and packets to be routed to them in the CBN. A connection request is then routed to the host H_2, where b is located, and an `accept` for it is routed back to a's host, H_1. After the connection is set up, ATP performs an $b[] = a[]$ transfer by sending data and ACKs between a and b through the CBN.

For this transfer, the resolution of an endpoint name into a node can be either *eager* or *lazy*. With eager resolution, the mapping of the destination endpoint to a host is performed once for every `send(a,b)` operation. In our example, a `send(a,b)` operation would mean that b's destination node address is found first, then data is sent to that node. At the destination, sending an ACK back to a requires locating a's node, then sending the ACK to it. A `send(a,b)` operation can be formally described in this case as:

```
M=locate(b)
send(a,M)
```

Here, `locate` uses the location service of the CBN to map the endpoint in the underlying network.

In the eager resolution scheme, packets between endpoints will be routed over the underlying network. This restricts endpoint mobility during the transfer: if b changes location after a `send` was issued from a, data may not reach it. This problem is solved if lazy resolution is used instead.

With lazy resolution, the mapping of the destination endpoint to a host is performed through routing from a to b in the CBN. In this case, the `send(a,b)` operation can be formally described as:

```
do
  t=next_hop(b)
  send(t, a)
until (t == b)
```

Here, `next_hop` finds a next hop on route from a to b in the CBN and content is forwarded to that hop until the target endpoint is matched.

3.2 Dynamically Split Connection

Consider for example a scenario where an intermediate node N on the current path used for reliable transport from source endpoint S to destination endpoint D is not able to forward a packet towards D due to a next-hop link failure. The normal behavior of N in a traditional "network layer" would be to drop the packet and possibly reply back with a control packet to the originating host. The source would timeout and retransmit, in the hope that the error is transient and the route would eventually come back up. If this is not the case, the connection is dropped after a number of retransmissions. Some protocols like TCP make an attempt to discover a new route after a few retransmissions, but this is still a sender end-host operation. Such a route repair blindly triggered by a source may result in flooding the network with useless communication for discovering a new route. The reason is that the end host reacts inadvertently to a problem somewhere deep in the network, while it would make more sense instead for the repair to be localized, close to the place where the trouble occurred.

Situations like this suggest that the transport protocol may benefit from augmenting intermediate nodes with transport functionality. In the above example, instead of dropping the packet, N may buffer it while taking corrective action to discover a new route towards destination, instead of leaving this operation to the source.

3.2.1 Split Transport Model

With an ATP, a connection can be dynamically split in the CBN by creating new intermediate endpoints and using them to break the original connection in multiple connections. The intermediate endpoints are subsequently registered in the same CBN.

In our example (Figure 5), suppose an endpoint a_1 is created on an intermediate node (according to some policy) for splitting the transfer b[] = a[].

This can be described by the following pseudo-code (note that in this high-level description we ignore details of content fragmentation into protocol data units):

```
a1[]=a[]
ack(a)

register(a1)
connect(b)
send(a1,b)
```

The above pseudo-code lists the steps that a node on the route from a to b takes when deciding to participate in the transfer. It first creates a new endpoint identifier a_1 to name temporarily buffered content. Next, it starts buffering the data and acknowledges it back to the a endpoint. The `ack(a)` statement marks the actual moment the connection is split: by acknowledging the content, the new node commits itself to reliable delivery of a's content to b. The source a would accept the ACK and discard the corresponding data.

On the forward path (towards b), b must be able to locate the new endpoint to send ACKs to it. For this, the new node must register itself as *the* a_1 endpoint. It will next connect to b using the normal connection establishment procedure, and continue with the transfer of its content $a_1[]$ to b. The registration of a_1 ensures that ACKs from b will be properly routed towards it. The intermediate endpoint in a split connection can later switch back to simple forwarding when it decides to give up its transport functions for a connection.

Connection splitting can be performed repeatedly on any segment of the (now) split connection. This results in multiple intermediate endpoints a_1, a_2, a_3, etc., if needed (e.g., for localized recovery) for delivery of the original content. The invariants maintained on any segment are: *(i)* a source always sends data to the *original* destination endpoint; *(ii)* a destination endpoint acknowledges data to the *last* source endpoint.

Note that the scheme exploits routing in the CBN for efficiency, through an asymmetry in the endpoints of a split with respect to an intermediate endpoint. An endpoint needs to know only the identity of its previous intermediate endpoint (for routing ACKs to it). As a source, the endpoint does not need to know the identity of its next intermediate endpoint. It will always send towards the destination endpoint of the transfer (b in the previous example). The protocol can exploit this asymmetry: if a link fails on the forward path, an intermediate endpoint can discover the failure and trigger a route discovery towards the destination. No data is lost (since it is buffered), and communication would resume on the new route.

3.2.2 Routing Optimization

With dynamic splitting, the burden shifts to the registration of the intermediate endpoints, which can be an expensive operation if registration would be advertised throughout the CBN. In fact, intermediate endpoints need only be advertised on a forward segment after a split-up. This will set up reverse path routes for routing ACKs from the next endpoint.

However, if an ACK cannot be routed back to the intermediate node due to network changes (e.g., endpoint mobility), then it will be dropped and a retransmission will take place from the source, causing a new route to be discovered. An alternate solution is for the ACK to discover a route before the retransmission timeout expires.

3.3 Fault Tolerance

We can classify the faults in a CBN into two categories: *(i)* link failures and *(ii)* node failures.

Link failures in a CBN are equivalent to a route failure in the underlying network. Since the route discovery service of the CBN can discover an alternate path if available, the transport protocol can use it to locally recover from link failures. Because any intermediate node can assume transport responsibility, such failures can act as *triggers* for a node at the "edge" of the failed link to start participating in the transport protocol. Such a node would simply split the connection (buffering data for the transfer, etc.) while invoking in parallel the CBN route discovery service to find a next hop towards the destination.

Node failures present a bigger challenge for ATP over a CBN. In the ATP model, we entrust the intermediate nodes to participate in data transport by splitting a connection. This relaxes the end-to-end semantics and relieves the source endpoint from error recovery operations. In the model, we assume that once an intermediate node enters the transfer, it commits itself to reliably delivering to the next endpoint the data it has acknowledged to the previous endpoint. However, this approach also increases the probability of data loss in case of an intermediate node failure.

To ensure that data is reliably delivered, we suggest that redundant copies of data be kept in the network. The number of copies in the network may be chosen by the application according to the network characteristics. For example, if an application chooses to keep k copies of the data in the network, each node releases its data buffers only when it has received k local acknowledgements for the data. While the data buffers are released only on receiving k ACKs, flow control in the protocol is still governed by the next endpoint in the connection. At any given time in the connection lifetime, we have at least k copies of the data buffered in the network. In case of a node failure, one of the $k-1$ endpoints that are buffering initiates a repair of the replica set.

We outline here a very simple scheme for fault-

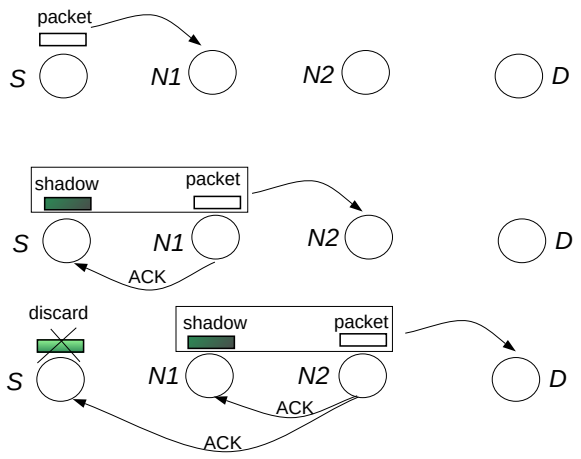

Figure 6: *Tandem replication protocol.*

tolerant operation in an ATP based on a *tandem replica protocol*. We call it a tandem protocol because it maintains its replica set on a chain of successive nodes, while propagating it forward on the path towards the destination.

Figure 6 shows an example of its operation, where the size of the tandem set is $k = 2$. We omit naming issues, details of the flow control mechanism, etc. A packet is sent on a connection from source S towards destination D. Node N_1 becomes the tandem head after it receives it and sends an ACK back to S. Next, N_1 forwards the packet to the next hop N_2 on the route to D. Upon receiving the packet, N_2 becomes the new head of the tandem: it acknowledges the packet back towards N_1 and S, and continues moving the tandem forward by sending the packet to the next hop N_3 to D. The source S knows that it has to stay in the tandem as long as it has not received k ACKs for the packet, which means the tandem has safely moved forward. When the second ACK arrives, S drops its copy of the packet. Note that with respect to the packet (now with copies on N_1 and N_2) N_2 has become a new source. The process is repeated, until the tandem head reaches D, which issues a special ACK to release the tail of the tandem.

There are many performance implications of maintaining additional copies of data in the network: additional buffer space is required at each intermediate endpoint in a connection, increased number of packets in the network, etc. A straightforward application of traditional fault tolerance schemes in this context may be too expensive. We believe that schemes for fault tolerance in ATPs should exploit its localized and autonomous nature along with the relaxed end-to-end semantics for better performance.

4 Design Guidelines

We believe that a transport protocol for content-based networks should be autonomous, unconstrained by details of lower-level mechanisms. At the same time, it should take into account and exploit the inherent nature of the CBN architectures for better performance, fault tolerance and flexibility. We identify the following key guidelines that we believe the design of a transport protocol should take into account:

- **Decoupled transport and network name spaces**. The endpoint name space should be decoupled from the physical node name space. The binding between the two is dynamic and may change during the lifetime of a connection.
- **Flexible endpoint naming scheme**. The endpoint naming scheme is solely managed by the application. The transport protocol does not enforce a particular endpoint naming scheme. The application is responsible for uniqueness of endpoint identifiers. The CBN must provide a service for registering endpoint names to the protocol.
- **Flexible routing and discovery**. There is no pre-defined routing infrastructure or preferred discovery mechanism. Routing can be content based, property based, etc. In general, an application can choose to implement its own discovery and routing mechanisms. The CBN must provide a service for endpoint location and routing.
- **Handling network instability**. No assumptions should be made on network stability. The network can be heterogeneous in terms of volatility. For example, it may be highly stable is some regions, while containing pockets or zones of high instability. The protocol must be able to adapt to such situations.
- **Relaxed end-to-end semantics**. The protocol should not be confined to rigid end-to-end semantics. It should exploit the intermediate nodes' ability to locally participate in protocol processing by creating intermediate endpoints when necessary and splitting connections. However, it should be flexible enough to fall-back to rigid end-to-end semantics when needed.

5 Related Work

Localized algorithms (simple local node behavior towards a global goal) have been advocated for sensor network coordination [10]. While the nature of the problem (data transport vs. coordination) is different in our case and the setup might be different in various types of networks, we believe that this idea reflects

a salient feature of networks addressable by content. We believe that data transport protocols should also exploit it and depart from the traditional end-to-end model, using cooperation between in-network nodes and moving towards localized mechanisms.

Recent work on large networks of embedded systems has focused on developing data aggregation and collection protocols [10, 13], new naming and addressing schemes for sensor networks [9], and system architectures for fixed-function sensor networks [12]. In particular [10, 13] have argued for attribute-based (data-centric) naming and dissemination of data in sensor networks. Recently, [6] proposed *cooperative computing* as a solution for programming distributed applications in unstable networks with unknown configurations, using application-controlled routing.

A number of systems that use naming by content for location and routing have been designed in the context of peer-to-peer services over the Internet [21, 17, 25, 18]. All these systems use distributed hashing schemes over named content for content location and build overlays over IP for location and routing. Work has also been done in the area of non-conventional schemes for location and routing based on names in the Internet [1, 11]. However, none of the above systems is concerned with using names to conduct communication between peers.

A source of inspiration for this work was our previous experience with connection migration in a standard transport protocol. We have developed Migratory TCP (M-TCP) [22, 20, 23], a reliable, connection-oriented protocol that enables transparent connection endpoint migration. Maintaining TCP compatibility while enabling a connection endpoint to move transparently to the other endpoint proved to be a hard problem we had to solve in M-TCP. This determined us to question the feasibility of a TCP-like migratory or mobile protocol in content-based networks.

A vast amount of work has been done in adapting the TCP to vagaries of wireless links, either in mixed wireless/wired or in all-wireless networks. Several feedback-based approaches rely on additional support from intermediate nodes to discriminate congestion from link failure: Explicit Loss Notification [4], Explicit Bad State Notification [3], TCP-F [8], TCP-BuS [14]. In some cases [14], they use ad hoc mechanisms for localized route repair (based on augmenting intermediate nodes with limited buffering functions). However, a sender is still responsible for error recovery as the main goal is to preserve TCP compatibility and, with it, the strong end-to-end semantics on which all TCP mechanisms are built. None of these schemes departs in a significant way from the end-to-end model.

Mobile-TCP approaches adapt TCP's behavior on the last one-hop wireless link between the mobile host and a base station connected to the wired network. Snooping protocols [5] monitor TCP traffic at the base station, build caches of unacknowledged data, and perform local recovery from wireless losses by link-layer retransmission. Split-connection protocols [2, 7, 24] split the TCP connection and handle traffic on behalf of the mobile host at the base station, thus breaking the end-to-end semantics. While the relaxed semantics of a split-connection Mobile TCP is an artifact of the adaptation solution, our split-connection protocols are explicitly built on this idea, applied throughout the network. In our ATP model, connections can be dynamically split, and a split may not be permanent. In our protocol, splitting can be used for more than just simple protocol adaptation: link failure recovery, error control, endpoint mobility, etc.

6 Conclusions

We have argued for a new approach to building autonomous transport protocols for content-based networks. This approach differs from current transport protocols such as TCP, where endpoints are bound to host addresses in the network, and the protocol enforces strong end-to-end semantics. We identify two key design principles for autonomous protocols: *(i)* an endpoint name space decoupled from the network host address space and *(ii)* relaxed end-to-end semantics, which allows intermediate nodes in the network to be dynamically involved in the transport.

We have used these principles to describe an autonomous transport protocol with dynamic connection splitting for reliable communication in content-based networks. To our best knowledge, this is the first attempt to discuss transport protocols in content-based networks. The approach we propose is by no means final, as many details were not discussed in this paper. We expect this paper, however, to challenge the current view on transport protocols and generate further research.

Acknowledgements

We thank Viraj Bhat for his contribution to the development of the ATP idea in its incipient stages.

References

[1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The Design and Implementation of an Intentional Naming System," *Proc. 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, Dec. 1999.

[2] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. 15th International Conference*

on Distributed Computing Systems (ICDCS '95), May 1995.

[3] B. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Improving Performance of TCP over Wireless Networks," *Proc. 17th International Conference on Distributed Computing Systems (ICDCS)*, 1997.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, Dec. 1997.

[5] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," *Proc. 1st ACM Int'l Conf. on Mobile Computing and Networking (Mobicom)*, nov 1995.

[6] C. Borcea, D. Iyer, P. Kang, A. Saxena, and L. Iftode, "Cooperative Computing for Distributed Embedded Systems," *Proc. 22nd International Conference on Distributed Computing Systems (ICDCS), July 2002. To Appear*, 2002.

[7] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks," *Computer Communication Review*, vol. 27, no. 5, pp. 19–43, Oct. 1997.

[8] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback Based Scheme For Improving TCP Performance in Ad-Hoc Wireless Networks," *Proc. 18th International Conference on Distributed Computing Systems (ICDCS)*, May 1998.

[9] J. Elson and D. Estrin, "Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks," *Proc. Twenty First International Conference on Distributed Computing Systems (ICDCS-21)*, Apr. 2001.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *Proc. 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Aug. 1999.

[11] M. Gritter and D. R. Cheriton, "An Architecture for Content Routing Support in the Internet," *Proc. 3rd USENIX Symposium on Internet Technologies and Systems (USITS-01)*, Mar. 2001.

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," *Proc. Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Nov. 2000.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. 6th Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, Aug. 2000.

[14] Dongkyun Kim, C.-K. Toh, and Yanghee Choi, "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks," *Journal of Communications and Networks*, vol. 3, no. 2, June 2001.

[15] J. Postel, "RFC 793: Transmission Control Protocol," Sept. 1981.

[16] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu, "Data-centric Storage in Sensornets," Submitted for review. February 1st, 2002.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM*, Aug. 2001.

[18] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329–??, 2001.

[19] A. C. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility," *Proc. 6th Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, Aug. 2000.

[20] K. Srinivasan, "M-TCP: Transport Layer Support for Highly Available Network Services," Tech. Rep. DCS-TR-459, Department of Computer Science, Rutgers University, Oct. 2001.

[21] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, Aug. 2001.

[22] F. Sultan, K. Srinivasan, and L. Iftode., "Transport Layer Support for Highly-Available Network Services," *The 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001, Extended version: Rutgers University Department of Computer Science Technical Report DCS TR-429.

[23] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode., "Migratory TCP: Connection Migration for Service Continuity over the Internet," *Proc. 22nd International Conference on Distributed Computing Systems (ICDCS)*, July 2002, To appear.

[24] K. Wang and S. K. Tripathi, "Mobile-End Transport Protocol: An Alternative to TCP/IP Over Wireless Links," *INFOCOM (3)*, 1998.

[25] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, Computer Science Division, Apr. 2001.