4

Opportunistic Delivery Services and Delay-Tolerant Networks

Sanjoy Paul

Abstract

The number of endpoints connected wirelessly to the Internet has long overtaken the number of wired endpoints, and the difference between the two is widening. Wireless mesh networks, sensor networks, and vehicular networks represent some of the new growth segments in wireless networking in addition to mobile data networks, which is currently the fastest-growing segment in the wireless industry. Wireless networks with time-varying bandwidth, error rate, and connectivity beg for opportunistic transport, especially when the link bandwidth is high, error rate is low, and the endpoint is connected to the network in contrast to when the link bandwidth is low, error rate is high, and the endpoint is not connected to the network. "Connected" is a binary attribute in TCP/IP, meaning one is either part of the Internet and can talk to everything or is isolated. In addition, connecting requires a globally unique IP address that is topologically stable on routing timescale (minutes to hours). This makes it difficult and inefficient to handle mobility and opportunistic transport in the Internet. Clearly we need a new networking paradigm that avoids a heavyweight operation like end-to-end connection and enables opportunistic transport. In addition to the these scenarios, given that the predominant use of the Internet today is for content distribution and content retrieval, there is a need for handling dissemination of content in an efficient manner. This chapter describes a network architecture that addresses the previously mentioned unique requirements.

4.1 Introduction

Caching[1,2] and Content Distribution Networks (CDNs)[3,4] have proven to be extremely useful on the Internet today. However, the mechanisms used to leverage the usage of caches on the Internet today are not very clean. For

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

example, to use an institutional proxy cache, typically, the browsers have to be *configured* to point to the proxy cache, or a special device like a Layer-4 switch has to be used to transparently redirect Web requests to the institutional cache, or some automated scripts are run to identify the proxy cache for the corresponding browser. Multiple mechanisms exist because each has its own pros and cons, and none of these techniques is a clear winner. Similarly, to redirect a user request to the nearest mirror server of a CDN, different CDN vendors use different mechanisms. Moreover, the details of the mechanism and signaling used by a CDN vendor like Akamai[3] and/or Limelight Networks[4] are *proprietary* even though we know it is most likely based on Domain Name System (DNS). Whereas the DNS-based redirection is best for CDN vendors like Akamai and Limelight networks who do not *own* the network, it may not be the best way out for Network Service Providers like AT&T, who own their network, for building a CDN. Once again, just as in the case of caching, multiple techniques are used in CDNs to redirect an end-user request to the "nearest" mirror server. In summary, *several* complex parallel *control and signaling* infrastructures have been built on top of the Internet to make use of the caches (or storage nodes). The question is, *if* we had the luxury of building a *clean-slate* next-generation Internet, would it make sense to maintain status quo or to design a simpler unified mechanism to leverage the well-proven benefits of caching (storage) in the network.

A parallel development has been happening in the Internet community in the context of Delay/Disruption-Tolerant Networking (DTN)[5] whose objective is to deal with *disruption* or *intermittent* connections on the Internet that the traditional TCP/IP paradigm cannot handle efficiently. Interestingly enough, DTN community recognized the need for hop-by-hop transport combined with *caching* as a way of mitigating the effect of disruption in communication. DTN community has proposed a different control and signaling mechanism on top of the Internet.

Yet another community, mostly driven by the researchers in the field of mobile communications and networking,[6,7,23] had realized the benefit of hop-by-hop transport in multihop wireless communications to improve performance of content delivery, and once again caching plays a central role. To take advantage of caching, this community is designing yet another control and signaling mechanism.

Given that caching is so central to multiple communities and that it is being used to serve a variety of needs, and given that due to the limitations of the current Internet design, each community has to come up with its own control and signaling mechanism, and also given the luxury of designing the next-generation Internet from scratch, there is tremendous benefit in designing a unified protocol for leveraging the caches to meet the needs of these diverse communities.

The architecture proposed in this chapter is not an alternative to what the CDN community has deployed, or DTN community has proposed, or mobility community has proposed; rather it is an attempt to leverage the best ideas from these communities and to put them together into a unified framework. In the context of the current Internet, this framework can be thought of as an overlay network on top of the Internet. In a clean-state design of the next-generation Internet, the unified framework may very well be integrated into the network itself.

4.2 Design Principles

There are several reasons why a new architecture is needed for opportunistic transport and delay-tolerant networking. First, the Internet architecture assumes that there exists an end-to-end path between the endpoints that need to communicate and exchange information. This is certainly not true for mobile endpoints that may not be within the range to communicate or for sensor nodes that wake up intermittently to communicate. Second, the Internet architecture computes a single path from the source to the destination for routing packets between the two endpoints. However, there are several scenarios where computing a path from the source to the destination is not possible ahead of time, especially when the source or the destination is not connected to the network. In addition, in the event of congestion along the precomputed path, packets get delayed. It may be a better approach to decide on the route dynamically as opposed to statically before the communication begins. Third, packet switching is assumed to be the most appropriate abstraction for interconnecting heterogeneous systems. However, when the end-users are mostly interested in content, the appropriate switching entities need not be packets, but rather messages or contents themselves. Fourth, the Internet architecture assumes that packet loss rate is small and the lost packets can be recovered through end-to-end retransmissions. However, when such assumptions fail, as in time-varying wireless links where packet loss rate could be significantly high from time to time, or in systems where an end-to-end path does not exist, the end-to-end performance suffers badly. In general, these shortcomings of the Internet Architecture need to be addressed for the following types of networks:

1. Hybrid Fixed and Mobile Networks
2. Military Ad hoc Networks
3. Vehicular Networks
4. Mobile Wireless Networks
5. Media Distribution Networks
6. Sensor Networks

All the previously mentioned factors lead to the design of a new architecture for opportunistic communication and delay-tolerant networking with the following characteristics:

1. Network elements should have *persistent memory* or *storage (cache*) integrated in them. This is important because the intended destination may be out of reach and the message may need to be stored at an intermediate network element until the intended destination gets connected. The intermediary carrying the message to the final destination could also be mobile and hence may need to hold on to the message until it gets back into the connected network or gets a chance to hand over the message to the destination. The side-effect of storing content in the network is the efficient delivery that can be achieved by virtue of delivering the content from the network itself as opposed to from a server outside the network.
2. The network should not be built on packet-switching technology but rather on message-switching technology where a message could be as big as the entire content file itself.
3. Messages should be transmitted between two successive intermediate network elements using a reliable *virtual link layer* protocol. The link between two successive network elements is called virtual because it consists of multiple hops in the underlying physical network but behaves as a single link between two nodes in the overlay network. The link layer protocol should be configurable so that it can be tuned to the characteristics of the virtual link.
4. *Routing* decisions should not be made at the source at the time of transmission but rather should be made at each intermediate network element as the message is transmitted hop by hop.
5. In addition to address-based routing, there is a need for *content-based* routing.
6. *Network layer* should support multiple classes of service so that some messages are treated with higher priority compared to others based on the urgency of message delivery.
7. *Naming and late binding* should be two of the most important support services in the network. Late binding is useful because resolving names upfront makes sense only when the routing needs to be decided at the source. However, when the destination may not even be connected to the network or the exact location of the destination is not known ahead of time, it makes sense to resolve names to addresses toward the end of the delivery process.
8. Semantics of *multicasting* needs to be defined differently because the members of a multicast group may not be online when the multicast session starts

or ends. Moreover, the source and/or destinations may be mobile leading to dynamic formation of the multicast tree.

9. *Transport layer* becomes minimal in this case because the network itself provides reliable transmission between network elements. Moreover, since the final destination may not be connected, it may be difficult, if not impossible, to have a timely end-to-end acknowledgment as in the case of TCP in the Internet.
10. Acknowledgment continues to make sense for the *Application layer* protocol. However, the semantics may vary depending on the circumstances.

4.3 Alternative Architectures

Several network architectures and associated protocols have been proposed to handle disruptive communication. However, the driving factors behind these architectures have been different and hence, despite significant functional commonality, these architectures evolved slightly differently as described next.

4.3.1 Delay and Disruption Tolerant Networking (DTN) (RFC 4838)

Delay and Disruption Tolerant Networking (DTN)[5,30] was the result of combining research in the fields of mobile and ad hoc networking (MANET), vehicular ad hoc networking, and the DARPA-funded research on Interplanetary Internet (IPN). The IPN architecture that was developed to cope with significant delays and packet corruption of deep-space communications laid the foundation of DTN architecture. However, it evolved significantly from the initial IPN architecture as the focus shifted from just Interplanetary Internet to more general concept of Delay and Disruption Tolerant Networking.

4.3.1.1 Architecture

DTN (RFC 4838) architecture consists of endpoints (source and destination) and intermediate nodes, some of which merely forward bundles (bundles are equivalent of packets in DTN architecture) and some, in addition to forwarding bundles, also store them for forwarding at an opportunistic moment some time in the future (such nodes are referred to as custodians). All nodes in the architecture have a common protocol layer, namely the bundle protocol layer that binds together all components of DTN architecture. Bundle protocol layer, as described later, is the equivalent of TCP/IP in the Internet architecture. Architectural highlights of DTN are presented next.

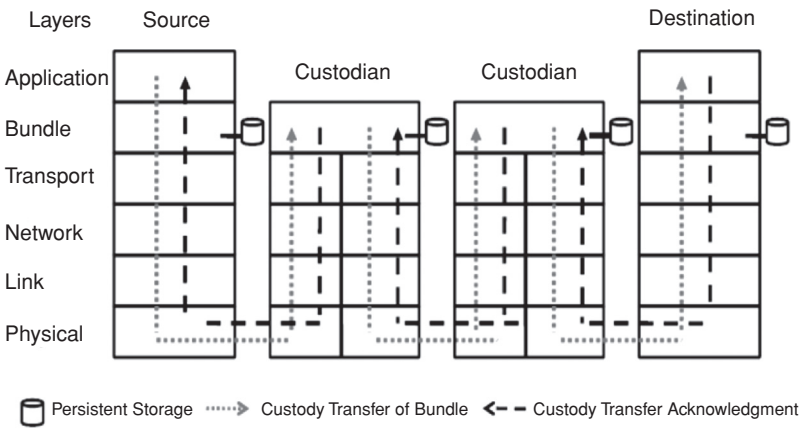


Figure 4.1. DTN architecture (RFC 4838).

Hop-By-Hop Delivery

Fundamental paradigm used in DTN networks is "store and forward" where storing is persistent and not transient as in IP networks. Furthermore, the unit of storage and forwarding in DTN networks is a "bundle" as opposed to a "packet." A bundle is formed by adding relevant header information to an Application Data Unit (ADU) so that the ADU can be routed to the right destination by the bundle layer. A bundle header consists of the original source and final destination endpoint identifier (EID), so that each intermediate node in the DTN network knows where the bundle originated from and where it is headed. Each intermediate node forwards the bundle based on the EID. However, all intermediate nodes are not the same. Some of them simply forward the bundle toward the final destination, whereas some others take on custody of the bundle. Taking custody of a bundle means taking on the responsibility of "reliably" transferring the bundle to the next custodian or to the final destination, whichever may be closer. Reliable transmission requires the custodian to figure out if the bundle has been successfully delivered to the next custodian or not, and if not, retransmit it until the bundle reaches the desired custodian and/or the final destination.

Naming and Late Binding

Endpoints in DTN architecture are identified using EID that follows the syntax of Uniform Resource Identifier (URI) (RFC 3956). Each EID may refer to either a single destination endpoint or a set of destination endpoints. The latter is applicable to anycast and multicast.

Binding refers to mapping an EID to the next-hop EID or the lower-layer address for transmission. For example, in the context of the Internet, the binding happens at the source where the name is mapped into an IP address using DNS. However, in case of DTN architecture, EIDs may be reinterpreted at each

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

intermediate node because the final destination may not be connected to the network or its location in the network may not be known. Thus, DTN nodes perform "name-based" routing with late binding as opposed to "address-based" routing.

4.3.1.2 Protocols

Virtual Link (Bundle Delivery) Layer

In DTN networks, "virtual" link (bundle delivery) layer protocol is responsible for transferring a "bundle" from one DTN node to the next DTN node just as the link layer protocol is responsible for transferring a packet from one router (host) to the next router (host) in the TCP/IP protocol stack. The "virtual" link (bundle delivery) layer in DTN networks rides on top of traditional transport layer protocols (TCP and UDP).

In contrast to the TCP/IP protocol stack where the link layer is usually best effort (no guarantee of delivery, for example in Ethernet), the bundle layer in DTN supports both best-effort as well as reliable delivery mechanisms. Best-effort delivery happens between two nodes when the next-hop DTN node is not a "custodian." However, between two "custodian" nodes, the delivery is expected to be "reliable."

Virtual Network (Bundle Forwarding and Routing) Layer

In DTN networks, "virtual" network (bundle forwarding and routing) layer protocol is responsible for computing the route of a "bundle" from the original source to the final destination. DTN node does the forwarding of a "bundle" to the next-hop node. In fact, the "virtual" network (bundle forwarding and routing) layer resides on top of traditional transport layer protocols (TCP and UDP).

Bundle header contains the original source EID, final destination EID, current custodian EID, and report-to EID in addition to some other fields. Forwarding decisions are made based on the final destination EID, and reports, such as return receipt, among others, are sent to the report-to EID.

Routing is tricky in DTN because the capacity and delay in DTN links vary with time. If link characteristics are known ahead of time, forwarding decisions can be made in an intelligent manner. However, many a time, such information is not available, and then routing becomes challenging. In general, the links could be persistent (DSL line), on-demand (dial-up modem), scheduled intermittent (low-orbiting satellite), opportunistic intermittent (unscheduled low-flying aircraft), or predictive intermittent (based on a previously observed pattern). Different routing protocols are appropriate for different types of links.

DTN architecture supports routing and forwarding of anycast and multicast traffic in addition to that of unicast traffic. However, the semantics of multicast routing in DTN is tricky, because a member of the multicast group might express

interest in a content that might have already been delivered to other members of the multicast group. This requires support for storage and forwarding at intermediate nodes for delivery at a later point of time.

Virtual Transport (Bundle Flow Control and Congestion Control) Layer
In DTN networks, "virtual" transport (bundle flow control and congestion control) layer protocol is responsible for ensuring that the average rate at which a sending node transmits data to a receiving node does not exceed the average rate at which the receiving node is prepared to receive data (flow control), and the aggregate rate at which the senders inject traffic into the network does not exceed the maximum aggregate rate at which the network can deliver data to the destinations over time (congestion control). In addition, there are various acknowledgment schemes to guarantee end-to-end delivery. Because the "virtual" transport protocol for DTN network rides on top of transport layer protocols (TCP and UDP), it can leverage both the flow/congestion control of TCP and the acknowledgment scheme of TCP for its own "equivalent" functions at a higher level.

Application Layer
Applications interface with the DTN architecture asynchronously and that is the most appropriate mechanism in long/variable delay environments. Usually the applications register callback actions when certain triggering events occur (such as arrival of an ADU). The application layer protocol generates ADUs and uses the bundle layer for forwarding and delivery.

4.3.2 BBN's SPINDLE

BBN's SPINDLE[8] program was driven by DARPA with the objective of transforming U.S. military into an agile, distributed network-centric force. To achieve that goal, it was critically important to have access to mission-related information even under temporary disruptions to connectivity in the Global Information Grid (GIG). DARPA's Disruption Tolerant Networking (DTN) program, with the above goal in mind, has been developing technologies that enable access to information when stable end-to-end paths do not exist and infrastructure access cannot be assured. DTN technology makes use of persistence within network nodes, along with the opportunistic use of mobility, to overcome disruptions to connectivity. That is the genesis of BBN's SPINDLE architecture.

BBN's SPINDLE architecture is designed on the principle of extensibility with the goal of leveraging the same architecture for serving a variety of next-generation networking needs. A DTN application that focuses on delivering a bundle to the destination in an intermittently connected network would have

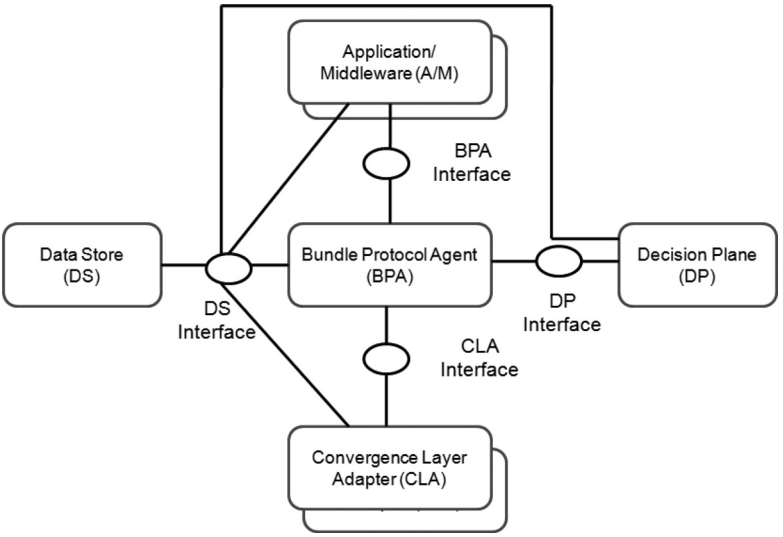


Figure 4.2. BBN's SPINDLE architecture.

different needs compared to a content discovery-and-retrieval solution. However, BBN's SPINDLE network is designed to meet the needs of both of these seemingly disparate types of applications through its extensible architecture. The details of the architecture are described further in this chapter.

4.3.2.1 Architecture

The core of SPINDLE architecture consists of Bundle Protocol Agent (BPA) that implements the main functionality of bundle protocol (RFC 4838). For example, BPA implements forwarding a bundle to the next-hop DTN node, performs delivery of a bundle to the applications, implements custody-transfer mechanism, and so on. However, the routing and forwarding functions, the implementation of reliable delivery of bundle, and such are decoupled from the basic forwarding functionality of the bundle protocol and are designed as separate components. In fact, the other components of the SPINDLE architecture are Data Store (DS), Decision Plane (DP), Convergence Layer Adapter (CLA), and Application/Middleware (A/M). These components are coupled with the core BPA component through Inter Component Communication Protocol (ICCP).

Bundle Protocol Agent (BPA)
Bundle Protocol Agent offers the services of bundle protocol (BP). It executes the procedures of BP and that of bundle security protocol (BSP) in cooperation with other components of the architecture. For example, even though BPA is responsible for implementing the mechanisms of the BP, such as reading,

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

creating, and updating the fields in the bundle header, it has the flexibility of leveraging the DP component of the SPINDLE architecture for any key decisions, such as those related to policy or optimization.

Main functions of BPA are:

1. Forwarding a bundle to the next-hop DTN node, whether it is for unicast, anycast, or multicast. However, the next-hop computation is done by the DP and passed on to BPA.
2. Doing fragmentation and reassembly of bundle payload as needed to adapt the delivery of payload over a link with time-varying capacity.
3. Implementing custody-transfer mechanisms in the bundle header, such as sending custody acknowledgment. However, whether to accept or reject custody is determined by the DP again.
4. Delivering a bundle to a "registered" application.
5. Discarding and deleting a bundle. Once again, it is the DP that decides whether a bundle should be discarded or not.
6. Implementing all security functions, such as authentication, confidentiality, and data integrity.

In addition to the functionalities on the list, BPA implements agent interface that can be accessed by applications and it uses the interfaces exposed by other components of the architecture.

Data Store (DS)

DS implements persistent storage used to store not only the bundles, but also the bundle metadata, network state information, and application state information. Network-state information includes, among others, routing tables, content metadata, and policies, whereas application-state information includes registration information, application metadata, and so on.

Data Store implements a full Data Base Management System (DBMS) to enable basic database functions such as query processing.

Knowledge Based (KB) systems can also be integrated with DS to enable advanced inferencing based on execution of rules.

Decision Plane (DP)

If BPA is the heart of the system, DP is the brain. Specifically, DP is responsible for routing information dissemination, route computation, routing table updates, late binding and name resolution, policy handling, content caching and replication decision, content search, and other decisions. DP consists of several modules:

1. *Routing information dissemination module*: This module is responsible for exchanging routing-related information among the network elements.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

Specifically, this module decides what information will be shared with whom and when. In addition to disseminating the information, this module also collects the routing-related information in incoming bundles and updates the relevant entries in the knowledge base.

2. *Routing module*: This module is responsible for computing routes for unicast and multicast, for updating the routing table entries, for generating next hops for bundles, for scheduling the bundles, for making decisions about whether to take custody of a bundle or not, and so forth.
3. *Policy module*: This module is responsible for interpreting policies, enforcing policies, dispatching events based on policies, enabling users to add/delete policies, and for subjecting bundles to policies as they pass through the DTN node.
4. *Naming and late binding module*: This module is responsible for resolving names of DTN nodes and feeding the information to the router module so that the right decision about forwarding a bundle can be taken. Usually, this module is invoked and used when the bundle is close to the final destination or close to the care-of address of the final destination where it will be stored for opportunistic delivery to the final destination.
5. *Content module*: This module is used for content-based access, specifically for content search, content caching and replication, content routing, and other content-related functionality.

Convergence Layer Adapter (CLA)
Convergence Layer Adapter is responsible for actual transport of the bundles. CLA leverages whatever transport functionality is available from the underlying network. Status of links (available or not), schedule (for opportunistic delivery), and quality-of-service (QoS) parameters are all monitored by CLA, and the relevant information is passed on to the relevant modules of the architecture for their efficient functioning.

Application/Middleware (A/M)
Application/Middleware module is responsible for sending and receiving bundles based on application needs. This module leverages the services exposed by BPA.

4.3.2.2 Protocols

Virtual Link Layer
In BBN's SPINDLE network, "virtual" link layer functionality is implemented by CLA. The beauty of CLA is that it is not limited to using TCP and UDP, rather it can potentially use any custom protocol (such as, CLAP [23]) that might be available at the corresponding DTN node for use in a specific type of network (for example, CLAP may be available at a DTN node and it may be

the best-suited protocol for wireless links with highly time-varying bandwidth, delay, and error characteristics).

Virtual Network Layer
In the SPINDLE architecture, the "virtual" "network layer" functionality is implemented by the Decision Plane (DP). The beauty of DP is that it is not limited to using any specific protocol; rather it allows usage of any protocol to disseminate and assimilate routing information. Moreover, the routing information is also customizable, meaning the information that will be distributed will depend on the type of routing. For example, routing information to be disseminated for content-based routing could be very different from the routing information needed for traditional address-based routing. The SPINDLE architecture supports this flexibility. Specifically, policy-based routing, content-based routing, late binding, and rich naming, among others, are supported in an extensible manner by DP in The SPINDLE architecture.

Virtual Transport Layer
The "virtual" transport layer protocol is responsible for ensuring flow control and congestion control and is implemented by DP. This is because the network state information, including congestion, is available to and disseminated by the DP module.

Application Layer
Applications interface with BBN's SPINDLE network architecture asynchronously, and that is the most appropriate mechanism in long/variable delay environments. In the SPINDLE architecture, the application-layer functionality is implemented by A/M module. A/M module also provides multiplexed DTN communication service to non-DTN applications running on the node.

4.3.3 KioskNet

KioskNet[9,10] started at the University of Waterloo with the goal of providing very-low-cost Internet access to rural villages in developing countries using the principles of Delay-Tolerant Networking. KioskNet system uses vehicles, such as buses, to ferry data between village kiosks and Internet gateways in nearby urban centers. The data carried by the buses from the rural areas are reassembled at an Intermediary (or Proxy Server) for interaction with legacy servers.

4.3.3.1 Architecture

KioskNet consists of a set of kiosks from which *ferries* (buses) carry data to a set of *gateways* that communicate with a *proxy* on the Internet. The ferries not

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

only carry data from the kiosks, but they also carry data to the kiosks. The main architectural components of KioskNet are described in more detail further in the chapter.

Kiosks

Each kiosk is equipped with a server referred to as kiosk controller, from which one or more PCs can boot off. Kiosk controllers have WiFi connectivity to allow users to connect to them wirelessly. In addition, although Kiosk controllers could have different types of backhaul, such as dial-up, GPRS, or VSAT, the most interesting one from the perspective of DTN is the mechanical backhaul, such as ferries (buses, cars, motorbikes, etc.). The kiosk is expected to be used by two types of users. First type of users use a PC that boots over the network from the kiosk controller and can then access and execute application binaries provided by the kiosk controller. The second type of users uses their own devices such as smart phones, PDAs, and laptops to connect to one or more kiosk controllers or a bus directly and use them as wireless hot spots that provide store-and-forward access to the Internet.

A KioskNet *region* consists of a set of kiosks in the same geographic area administered by the same entity. This means all entities within the region are certified by the same certificate authority. In addition, from the networking perspective, all data bundles are flooded within a region. Figure 4.3 shows a system with two regions, which can be managed either by different administrative entities or by a single administrative entity.

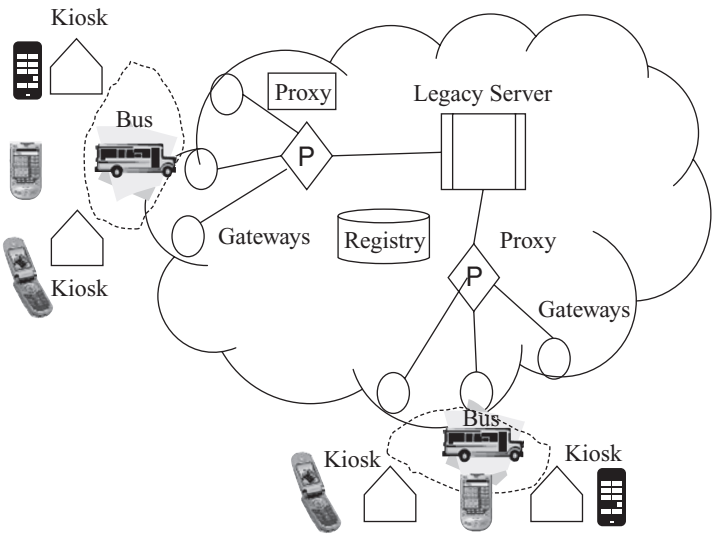


Figure 4.3. KioskNet architecture.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

Ferries
Ferries provide internet connectivity to the kiosks via a mechanical backhaul. Examples of ferries include cars, buses, motorbikes, or trains that pass by a kiosk and an Internet gateway. A ferry has a PC with 20–40GB of storage and a WiFi network interface and is powered by the vehicle's own battery. The PC communicates opportunistically with the kiosk controllers and Internet gateways when it comes within their coverage area. During an opportunistic communication session, which may last up to several minutes, hundreds of MB of data can be transferred in each direction. This data is stored and forwarded in the form of self-identifying *bundles*. Ferries upload and download bundles opportunistically to and from an Internet gateway.

Gateways
A gateway is just a PC with WiFi network interface and a broadband (DSL or Cable) Internet access. A gateway collects data opportunistically from a ferry and holds it in local storage before uploading it to the Internet through the proxy. A region may have one or more gateways.

Proxy
Most likely communication between a kiosk user and the Internet would be for existing services such as e-mail, or for accessing back-end systems that provide government-to-citizen services. Legacy servers that provide such services typically are not designed to handle either long delays or disconnections, and most importantly, they cannot be easily modified. Therefore, the architecture requires a disconnection-aware proxy that hides end-user disconnection from legacy servers. A proxy is assumed to exist in every region.

The proxy is resident in the Internet and has two halves. One half establishes disconnection-tolerant connection sessions with applications running on the kiosk controller or on mobile users' devices. The other half communicates with legacy servers on behalf of disconnected users. When a proxy receives application data from a legacy server, it transfers the data to an appropriate gateway that eventually forwards it to a passing ferry. The ferry delivers the data to a kiosk, which in turn passes it to kiosk users.

In the opposite direction, when a kiosk user wants to send data to the Internet, it uses a ferry to transport the data to a gateway that in turn transfers it to a proxy. The proxy passes received data to the legacy Internet servers.

4.3.3.2 Protocols

Virtual Link Layer
In KioskNet, TCP is used as the "virtual" link-layer protocol and is responsible for transferring a "bundle" from one DTN node to the next DTN node. Note that

the mobile device (cell phone), kiosk controller, ferry, gateway, as well as the proxy are considered DTN nodes in KioskNet architecture.

Virtual Network Layer

In KioskNet architecture, "virtual" network layer protocol is responsible for routing a "bundle" from the original source to the final destination. Routing within a disconnected region of KioskNet is different from routing from Internet to Kiosk.

Routing within a Disconnected Region

A routing algorithm allows a kiosk to decide which ferry to use to send data to the Internet, and for a gateway to decide which ferry to use to communicate with a particular kiosk. However, ferries may fail and ferry trajectories are not always known beforehand. Therefore, routing in KioskNet is a hard problem. Fortunately, a ferry can transfer several tens of megabytes of data to and from a kiosk as it passes by, and it can store tens of gigabytes of data on its hard drive. Based on these observations, routing is done using flooding, thereby trading off over-the-air bandwidth and storage for reliability and ease of routing. This means, in KioskNet, a kiosk or a gateway transfers all its data to every ferry that passes by and accepts data from every ferry. Clearly, this redundancy maximizes the probability of bundle delivery while eliminating routing decisions altogether. An added benefit is that with flooding, communication between kiosk users in the same region does not require a bundle to go to the proxy. Finally, flooding requires fewer configurations at deployment time, making KioskNet easier to deploy.

KioskNet eliminates the inefficiencies commonly associated with naïve flooding using two optimization techniques. First, before any data is transferred from a kiosk controller to a ferry and vice versa, bundle metadata is exchanged so that each side knows what bundles the other side has, and as a result can avoid accepting bundles it already has.

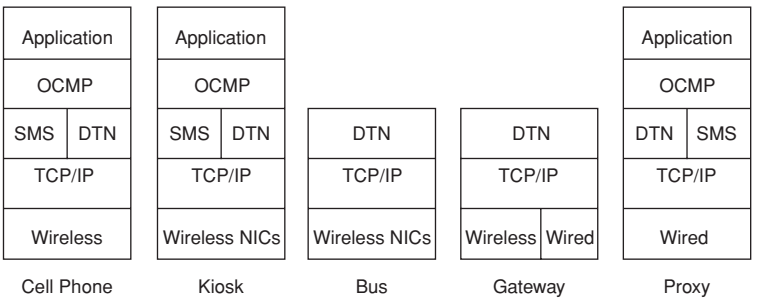


Figure 4.4. KioskNet protocol stack.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

Bundle metadata exchange happens as follows:

- The kiosk controller tells the ferry the user GUIDs registered at the kiosk.
- The ferry informs the kiosk controller of the bundle IDs on the ferry belonging to these users.
- The kiosk controller determines the missing bundles and requests them from the ferry.
- The ferry transfers these bundles to the kiosk controller. No metadata exchange is required in the other direction: A kiosk transfers all its bundles to every passing ferry.

In addition, although bundles sent from a kiosk destined to legacy servers on the Internet are flooded to all reachable gateways in the same region, and these gateways accept all bundles from all kiosks, these gateways coordinate with each other to make sure that each bundle will be sent to the proxy by one and only one gateway. This avoids wasting bandwidth on the link between the gateways and the proxy.

With these two optimizations, despite flooding, KioskNet resources, namely kiosk-to-bus communication link and the gateway-to-proxy link, are not unnecessarily wasted.

Routing of Internet-to-Kiosk Bundles

Data from legacy servers destined to kiosk users is first buffered at the responsible proxy, then sent to gateways that transfer bundles to ferries. After a bundle is sent to a gateway, it is flooded to reach its destination kiosk (i.e., handed to all ferries passing by that gateway).

Proxies are located in bandwidth-rich data centers, but gateways are connected to the Internet typically using slow dial-up or DSL links. Given that the link between the gateways and the proxy is the bottleneck, ideally the proxy should choose only *one* gateway in the region to send each bundle to, rather than flooding it to all the gateways in the region.

If the schedules of ferries are known to the proxy, a routing and scheduling algorithm can be used at the proxy that can choose the best gateway for each bundle and decide the order in which they are sent so as to minimize the overall delay. Moreover, this algorithm can also enforce arbitrary bandwidth allocation among kiosks. If bus schedules are not known, then the proxy has no choice but to flood it to all the gateways.

Virtual Transport Layer

In KioskNet, these capabilities are provided by the opportunistic connection management protocol (OCMP) that runs on top of DTN and other available network connections. OCMP can be viewed as a disconnection-tolerant and

policy-driven session layer that runs over both DTN and standard links. Each type of available communication path is modeled as a connection object (CO) within OCMP. For instance, the DTN mechanical backhaul path is a CO, just as a TCP connection over WiMAX or dial-up is.

OCMP allows a policy manager to arbitrarily assign bundles to transmission opportunities on COs. This scheduling problem is complex because it has to manage many competing interests: reducing end-to-end delay while not incurring excessive cost, and maximizing transmission reliability.

Application Layer

Applications, residing on mobile device (cell phone), kiosk controller, and the Proxy, interface with the KioskNet architecture asynchronously, which is the most appropriate mechanism in long/variable delay environments. Usually the applications layer protocol generates ADUs and uses the bundle layer for forwarding and delivery.

4.4 Converged Architecture

The previous section described several alternative architectures for dealing with disruption-tolerant networking. However, each architecture was designed to solve a slightly different problem and hence evolved differently. For example, the DTN architecture (RFC 4838) has been designed primarily to deal with significant delays, including long interruptions, in communications. BBN's SPINDLE architecture evolved from the need to provide access to information to military field force where stable end-to-end paths do not exist and infrastructure access cannot be assured. KioskNet was designed with the goal of providing very-low-cost Internet access to rural areas. DieselNet's[11,12,13] goal has been primarily to deal with the challenges of vehicular DTN. PocketNet[14,15] was designed to enable communications via storage and networking purely at the end-hosts.

Looking back at Section 4.2 in this chapter, shortcomings of the Internet architecture need to be addressed for a variety of networks including Hybrid Fixed and Mobile Networks, Military Ad hoc Networks, Vehicular Networks, Mobile Wireless Networks, Media Distribution Networks, and Sensor Networks.

Whereas DTN architecture primarily addresses the requirements of hybrid fixed and mobile networks, BBN's SPINDLE mostly focuses on military ad hoc networks, KioskNet deals with hybrid fixed and mobile networks, and DieselNet primarily explores DTN in vehicular networks. There are isolated efforts to deal with the time-varying characteristics of mobile wireless networks[23,31,32] and the existence of content delivery networks (CDNs) to address the needs of media distribution.[3,4]

A deeper look into the entire problem space exposes an underlying commonality in the basic building blocks of a converged network architecture that addresses all the previously mentioned problems in a uniform manner. The converged network architecture is referred to as Cache and Forward (CNF) Network Architecture.

4.4.1 Cache and Forward (CNF) Network Design Goals

Cache and Forward (CNF) architecture[16,17] evolved at WINLAB, Rutgers University, in order to solve four main problems: (1) efficient delivery and retrieval of video, (2) improving throughput in multihop wireless network, (3) improving content delivery in a mobile network where the mobile nodes may be intermittently connected to the wired infrastructure, and (4) improving communication in sensor networks. These issues are briefly discussed further in this chapter.

(1) Efficient delivery and retrieval of video (*challenges of media distribution networks*): Video will be driving the need for improved communications infrastructure in the foreseeable future, as is evident from the phenomenal rise of YouTube,[18] Revver,[19] and other video sharing sites in addition to the rise of Internet television, where specialized sites provide niche television content over the Internet. The uniqueness of video as content is the huge size of files that are several orders of magnitude larger than music (audio) files. P2P networking is helping scale the distribution of video, but the P2P delivery mechanism, by itself, cannot optimize the bandwidth usage in the underlying network. Moreover, the existing TCP/IP networking paradigm is not exactly suitable for video retrieval because the TCP/IP paradigm expects the application to figure out through an out-of-band mechanism (such as a search engine) the name/IP address of the server where a given video is hosted and then connect to the server to fetch the desired content, as opposed to allowing the application to query the network for a given video and retrieve it from the network, all operations being done in-band.
(2) Improving throughput in multihop wireless networks (*challenges of mobile wireless networks*): When TCP/IP is used over wireless links, performance is often degraded due to transport layer timeouts, and in-network solutions such as indirect TCP have been proposed in earlier work.[20] In addition, when TCP is used over multiple wireless hops (an increasingly common scenario), the so-called self-interference effect in which packets from the same flow (specifically the data and acknowledgment packets belonging to the same flow but traveling in opposite directions), contending for the same radio resources, can further degrade end-to-end performance.[21,22] For multihop wireless networks, the probability of impairment or disconnection in at least

one radio link can be quite high as the number of hops, n, increases. It can be shown that the probability of failure before the file transfer is completed is increased by a factor of n^2 over the probability of a single hop failure. This is almost an order-of-magnitude increase for $n = 3$ hops and is two orders of magnitude increase for $n = 10$ hops.

(3) Improving content delivery in mobile networks where mobile nodes may be intermittently connected to wired network (*challenges of hybrid fixed and mobile networks, military ad hoc, and vehicular networks*): The existing TCP/IP architecture embraced the concept of mobile IP to reach mobile hosts when the point of attachment of the mobile host (with the wired network) changes due to its mobility. However, the scope of mobile IP is limited to the case when mobile node is not disconnected from the wired network for a significant amount of time (longer than the lifetime of a typical Internet session). At the same time, research has shown that if content is temporarily stored in the network when the destination node is not connected to the wired network, and is ferried via "mobile nodes" to the destination node, the capacity of the wireless network increases substantially.[24,25]

(4) Improving communication in sensor networks (*challenges of sensor networks*): Internet applications involving sensors are expected to grow rapidly in the next ten years. Sensor scenarios have unique networking requirements,[33] including the ability to deal with disconnections due to wireless channel impairments as well as sensor hardware sleep modes. In addition, sensor applications tend to be data-centric and are thus more interested in content-aware services (e.g., querying data) than in connecting to a specific IP address.

CNF architecture was designed to address these issues in an efficient manner.

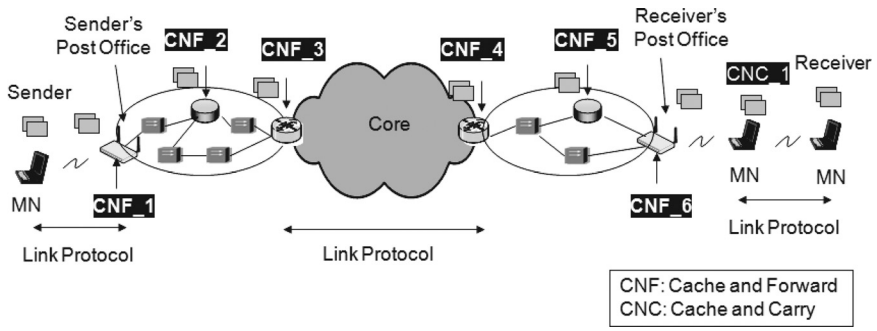


Figure 4.5. Cache-and-Forward (CNF) architecture.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

4.4.2 Architecture

The main concepts of CNF architecture are listed in this section:

Post Office (PO): The CNF architecture is based on the model of a postal network designed to transport large objects and provide a range of delivery services. Keeping in mind that the sender and/or receiver of an object may be mobile and may not be connected to the network, we introduce the concept of "Post Office" (PO) that serves as an indirection (rendezvous) point for senders and receivers. A sender deposits the object to be delivered in its PO and the network routes it to the receiver's PO, which holds the object until it is delivered to the final destination. Each sender and receiver may have multiple POs, where each PO is associated with a point of attachment in the wired network for a mobile endpoint (sender/receiver). In the context of DTN network and BBN's SPINDLE, a PO is nothing but a special type of custodian node, whereas in the context of KioskNet, PO is equivalent of a Gateway.

Cache and Forward (CNF) Router: The CNF Router is a network element with persistent storage and is responsible for routing packages within the CNF network. Packages are forwarded hop-by-hop (where a hop refers to a CNF hop and not an IP hop) from the sender's PO to the receiver's PO using forwarding tables updated by a routing protocol running either in the background (proactive) or on demand (reactive). In the context of DTN network and BBN's SPINDLE, a CNF Router is nothing but a DTN node that may or may not be a custodian node, whereas in the context of KioskNet, a Kiosk as well as a Gateway is a CNF Router.

Cache and Carry (CNC) Router: The CNC Router is a mobile network element that has persistent storage exactly as in a CNF Router, but is additionally mobile. Thus a CNC router can pick up a package from a CNF router, another CNC router, or a PO and carry it along. The CNC router may deliver the package to the intended receiver or to another CNC router that might have a better chance of delivering the package to the desired receiver. In the context of DTN network and BBN's SPINDLE, a CNC Router is nothing but a mobile DTN node that may or may not be a custodian node, whereas in the context of KioskNet, a Ferry is a CNC Router.

Content Identifier (CID): To make content a first-class entity in the network, we introduce the notion of persistent and globally unique content identifiers. Thus if a content is stored in multiple locations within the CNF network, it will be referred to by the *same* content identifier. The notion of a CID is in contrast to identifiers in the Internet, where content is identified

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

by a URL whose prefix consists of a string identifying the *location* of the content. CNF endpoints will request content from the network using content identifiers. Since none of the described alternative architectures in Section 3 considered content as a first-class citizen of the network, there was no need to have a specific content ID which is a "network" level ID rather they continued to use the "application" level ID, such as URLs as in the case of traditional Internet. However, CID is a fundamentally important concept in the converged network architecture.

Content Discovery: Since copies of the same content can be cached in multiple CNF routers in the network, discovering the CNF router with the desired content that is "closest" to the requesting endpoint must be designed into the architecture. We discuss this in more detail in the next section. Once again, since none of the described alternative architectures in Section 3 considered content as a first-class citizen of the network, there was no need to discover content within the "network"; rather they continued with the traditional Internet model whereby a search engine is expected to be used to locate the node holding the content and once the node is located, traditional network-based routing is used to access the content. One exception is BBN's SPINDLE architecture where the concept of content module has been conceived as a part of the Decision Plane module for enabling content-based access, specifically for content search, content caching and replication, content routing, and other content-related functionality. However, in the converged architecture, since content is a first-class citizen of the network, content discovery is part of the network layer functionality.

Type of Service: To differentiate between packages with different service delivery requirements (high priority, medium priority, low priority), a Type of Service (ToS) byte will be used in the package header. The ToS byte can be used in selecting the cache replacement policy and in determining the delivery schedule of packages at the CNF routers. This concept exists with both DTN architecture as well as BBN's SPINDLE architecture.

Multiple Delivery Mechanisms: A package destined for a receiver would be first delivered to, and stored in, the receiver's PO. There are several ways in which the package can be delivered from the PO to the receiver:

- A PO can inform the receiver that there is a package waiting for it at the PO and it (the receiver) should arrange to pick it up. The receiver can pick up the package when in range of that PO. Otherwise, it may ask its new PO and/or a CNC router to pick up the package on its behalf.
- A receiver can poll the PO to find out if there is a package waiting for pick up. If it is, and the receiver is within range of the PO, it can pick

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

up the package itself. Otherwise, it may ask its new PO and/or a CNC router to pick up the package on its behalf.

- A PO can proactively *push* the package to the receiver either directly or via CNC routers.

Routing mechanisms are not prescribed in either DTN architecture or in BBN's SPINDLE architecture as the architecture is expected to provide flexibility for choosing variety of routing techniques, especially at the edge of the wired network. KioskNet, however, does talk about intelligent flooding from the Gateway (equivalent of a PO) to the final destinations (PCs and/or mobile phones) as a way of routing. Nonetheless, the previously mentioned techniques in the converged architecture cover the entire gamut of routing from the wired edge node to the mobile or wirelessly connected end nodes (or final destinations).

Details of protocols used in CNF network are described next.

4.4.3 Protocols

The cache-and-forward architecture represents a set of new protocols that can be implemented either as a clean-slate implementation or on top of IP.

Virtual Link Layer: Virtual Link Layer in CNF architecture uses a reliable link layer protocol referred to as CNF LL in Figure 4.6. CNF LL protocol is used for reliable delivery of packages (bundles) between two adjacent CNF nodes that could be either CNF/CNC routers or CNF hosts. In the traditional TCP/IP network paradigm, two adjacent CNF nodes could be separated by multiple IP router hops or could be connected by a wireless link with highly time-varying bandwidth, delay, and loss characteristics. Because of this diversity, the link-layer protocol used in CNF is *configurable* to suit the characteristics on the underlying link. For example, if the virtual link in CNF consists of a few wired IP router hops, TCP may be the best virtual link-layer protocol for reliable delivery between two adjacent CNF nodes. On the other hand, if the virtual link

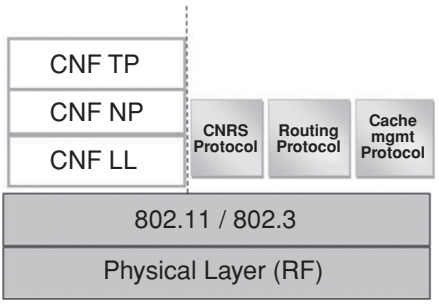


Figure 4.6. Cache-and-Forward protocol stack.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

in CNF consists of a wireless link with highly time-varying bandwidth, delay, and loss characteristics, then some proprietary protocol (such as CLAP) may be the best virtual link-layer protocol for reliable delivery between two adjacent CNF nodes.

Virtual Network Layer: Virtual Network Layer in CNF architecture uses a network layer protocol referred to as CNF NP in Figure 4.6. Each node in the CNF network, as described earlier, is assumed to have a large storage cache (∼TB) that can be used to store packages (files/file segments) in transit, as well as to offer in-network caching of popular content. CNF routers may either be wired or wireless, and some wireless routers may also be mobile. The basic service provided by the network is that of file delivery either in "push" or "pull" mode, that is, a mobile end-user may request a specific piece of content, or the content provider may push the content to one (unicast) or more (multicast) end-users. Each query and content file transported on the CNF network is carried as a CNF packet data unit or *package* in a strictly hop-by-hop fashion. The package is transported reliably between data stores at each CNF router before being prepared for the next hop toward its destination. The CNF network assumes the existence of a reliable link-layer protocol between any pair of CNF routers, and this protocol can be customized to the requirements of each wireless or wired link in the network. Packages are forwarded from node to node using opportunistic, short-horizon routing and scheduling policies at CNF nodes that take into consideration factors such as package size, link quality, and buffer occupancy. Alternative routing techniques may also be used opportunistically to deal with congestion or link failure. Caches in the network can create more complex scenarios. To retrieve any content, a host would send a query to the network with the location-independent CID, and the query would then be routed to the nearest CNF router using a *content routing* procedure, and the content would then be routed back to the host using the conventional routing capability mentioned earlier.

One unique aspect of CNF network layer protocol is the concept of "query routing" whereby an application may trigger query for a specific content object that may be stored "within" the CNF network. Note that this is possible as content is cached within the network itself. The query is routed from the originating node through the network. Just as a traditional router in the Internet maintains a routing table with IP address of destination networks/hosts, a CNF router maintains a routing table with CID and indicates next hops to follow in order to reach the desired content object stored within the network.

When the query is routed to the CNF node that has the desired content cached, the network layer triggers what is called response routing. Response is routed to the node that originated the query; the response forwarding is similar to traditional TCP/IP routing where packets are routed to a given destination IP

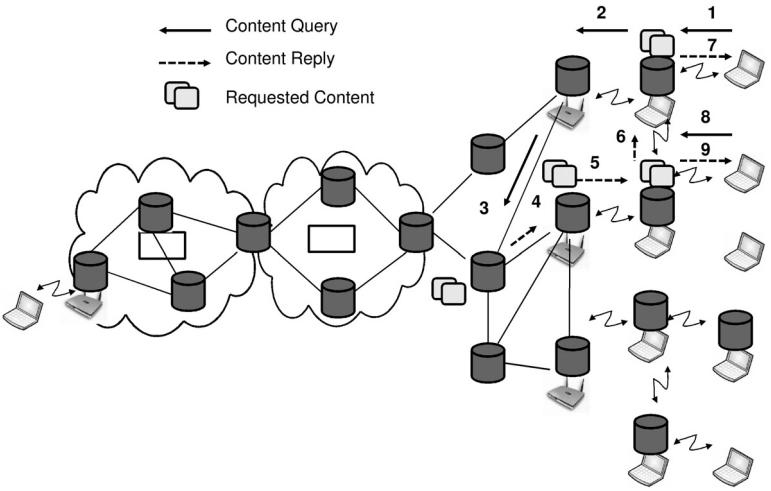


Figure 4.7. Routing of queries and content in the CNF network.

address. Perhaps the only difference is that the response (desired content object) is cached at each CNF router en route to the destination.

Figure 4.7 shows how content queries and responses are routed. Specifically, the content query originated at the top right laptop is routed through the CNF network (steps 1–3) until the query reached a CNF node with the desired content. Content is routed back (steps 4–7) where steps 5–7 are over wireless links that may or may not be mobile (if the link is a mobile wireless link, the corresponding node would be considered a cache and carry [CNC] router). As the content is routed through the CNF network, it is cached at intermediate nodes, and the benefit of caching shows up when another CNF host queries for the same object (step 8), which is now cached at the first hop (thanks to the previous query and content response routing and caching), and the content is immediately sent back to the originating CNF host (step 9).

Virtual Transport Layer: Virtual Transport Layer in CNF architecture uses a transport layer protocol referred to as CNF TP in Figure 4.6. The purpose of virtual transport layer protocol is to provide an end-to-end acknowledgment or notification for the delivery of content where the ends are defined as the original source and final destination. Because of reliable link-layer delivery between CNF nodes, transport layer also includes "intermediate" level acknowledgment and notification, which helps diagnose delivery problems in the same way as the tracking system does in FedEx or UPS delivery networks. In addition, it is the virtual transport layer that needs to deal with congestion and flow control as contents are transported across the CNF network from multiple sources to multiple destinations.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

4.4.3.1 Support Services

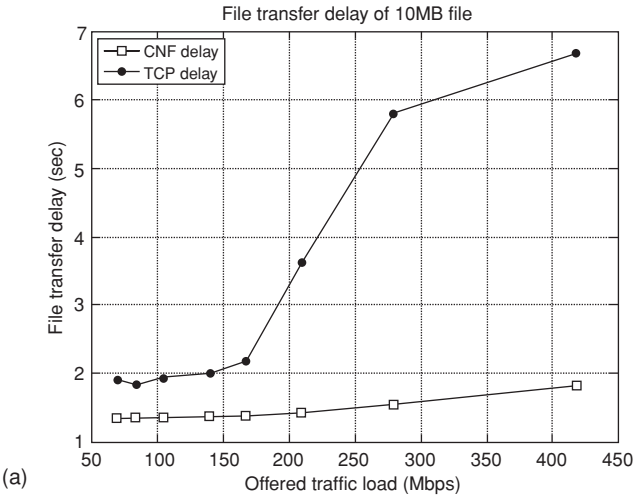
Content Name Resolution Service (CNRS): Because CNF network is designed to support efficient distribution and retrieval of content, and it allows applications to "query" for content cached in the network (see the virtual network layer), it is useful to have the IDs of CNF routers corresponding to a given file (or CID) that have the corresponding content cached. Since there would be potentially millions of objects, constantly updating the CNRS server for each content would not scale. Hence the idea in CNF is to update the CNRS server with the IDs of caches where a "popular" is cached. This information may be used by the CNF hosts (when originating a query) and/or by the CNF routers when forwarding the query in order to optimize routing.

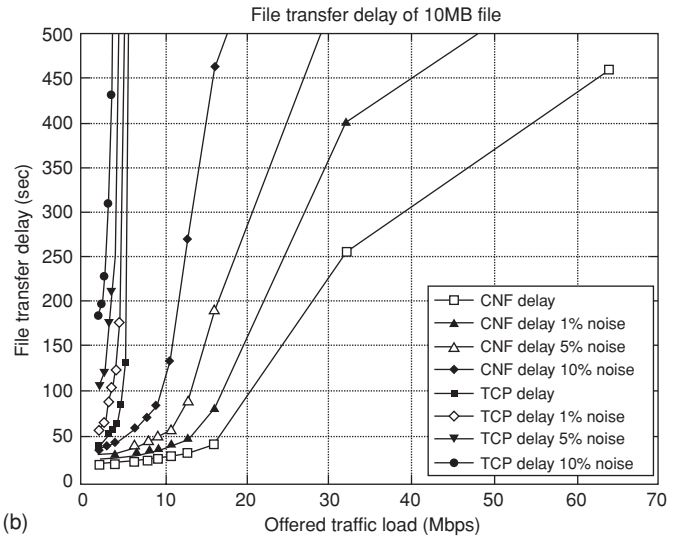
4.4.4 Performance of Protocols in CNF Architecture

4.4.4.1 CNF and TCP/IP Based Internet in Mobile Content Delivery

The goal of this section is to compare the performance of the proposed converged architecture (referred to as CNF architecture) with that of the traditional TCP/IP-based Internet architecture when it comes to delivery of content, especially large-size content, such as video, files when the sender or the receiver or both are mobile. To compare the performances of these two networks, a 24-node transit-stub network is considered, and the time taken in transferring a file from a source to a destination is computed under varying offered load where Offered Load = arrival rate × file size × no. of source nodes. Specifically, three scenarios are considered: (1) client and server nodes are wired, (2) client nodes are wireless but server nodes are wired, and (3) both client and server node are wireless. The results are shown in Figure 4.8 and are taken from Liu, Zhang, and Raychaudhuri.[26]

For CNF traffic, the transmission delay depends on the number of hops and the bandwidth of each hop. For TCP traffic, the transmission delay depends on the bandwidth of the bottleneck links. If all the links have the same bandwidth, TCP-based data transfer performs better than CNF-based data transfer because there is no store and forward delay in TCP and it is able to take full advantage of "streaming" data. However, if the bottleneck bandwidth is much smaller than that of the other links, TCP throughput is significantly reduced because it is limited by the bottleneck bandwidth. From the plots in Figure 4.8, it is clear that the file delivery time (referred to as file transfer delay in the figure) for TCP/IP network shoots up at much lower offered load compared to the case of CNF networks. Specifically, if the throughput is defined as the offered load with delay limit of 100s, the throughput of TCP is less than 2 Mbps in the case where both clients and servers are wirelessly connected, while CNF throughput is 2–7 Mbps.





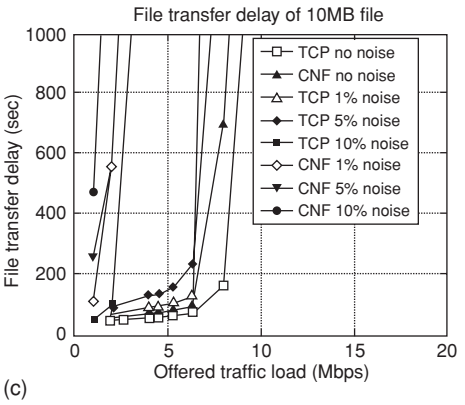


Figure 4.8. (a) Both clients and servers are wired nodes (b) Clients are wireless, servers are wired (c) Both clients and servers are wireless nodes.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

4.4.4.2 CNF and TCP/IP-Based Internet in Content Retrieval

The goal of this section is to compare the performance of the proposed converged architecture (referred to as CNF architecture) with that of the traditional TCP/IP-based Internet architecture when it comes to content retrieval. To compare the performances of these two networks, a 12-node transit-stub network is considered, the time taken to retrieve a specific content is computed when the network has the intelligence (as in the converged network architecture) versus when the network is dumb (as in the traditional Internet), and the intelligence of locating content resides outside the network at the application level.

Two schemes are compared in Lijun, Yanyong, and Sanjoy[27]: (1) Server Only, or SO (meaning that the content resides only on the servers) and (2) Cache and Capture, or CC (meaning content gets cached in the network as it transits through the network and hence can be retrieved from the network as opposed to from the server only) under three different scenarios: (a) small network, many requests, (b) large network, few requests, and (c) large network, many requests.[27]

It is clear from Figure 4.9 that whereas caching helps in every scenario, request number has a bigger impact on the caching effect than network size. In both scenarios (a) and (c), where a node makes a large number of requests, integrated caching can improve the performance by more than 52 percent, whereas in scenario (b), the improvement is only 24 percent. This is because more requests are served off the cache in cases (a) and (c) compared to that in case (b).

In the histograms shown in Figure 4.10, the first bin corresponds to the number of requests that were satisfied by a one-hop neighbor of the requester, either because that neighbor is the server hosting the content or because the neighbor has a copy of the named content in its cache, and the same explanation holds for the ith bin. Figures 4.10 (a)–(c) show that Caching and Capture (CC) as proposed in the CNF architecture can satisfy many more requests by nodes that are within a small number of hops from the requester than Server Only (SO) where the content is located only at the server outside the network as in the traditional TCP/IP-based Internet architecture. This clearly demonstrates

Scenario	(a)	(b)	(c)
SO (seconds)	0.369	0.737	0.635
CC (seconds)	0.175	0.561	0.304
Improvement	52.6%	23.9%	52.4%

Figure 4.9. Comparison of content retrieval schemes in CNF networks.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

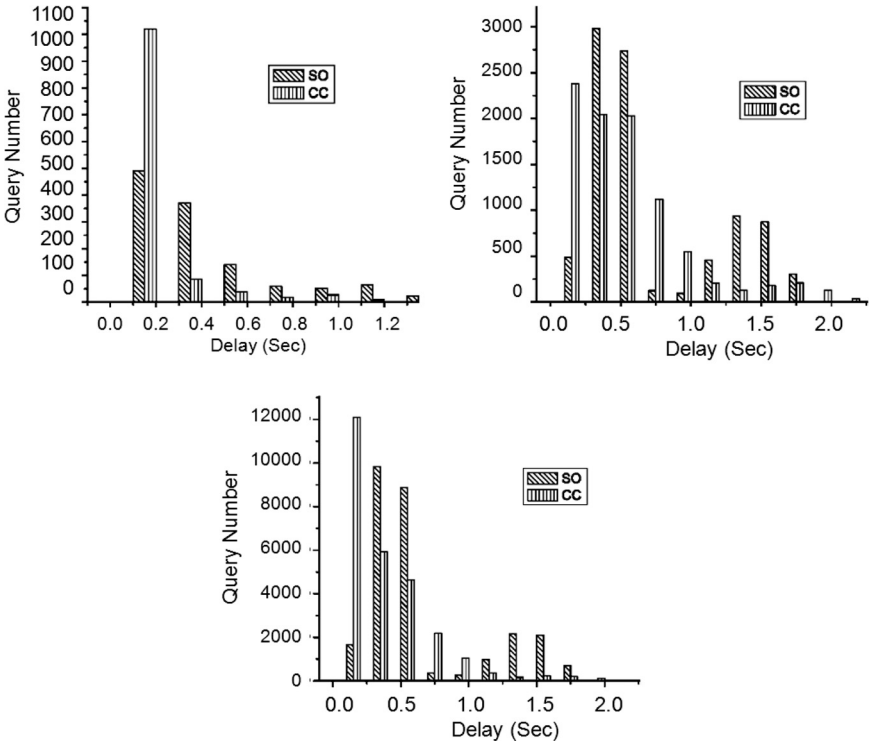


Figure 4.10. Histogram of content retrieval latency (scenarios a, b, and c).

the benefit of integrated caching and routing that is a core part of the proposed converged architecture.

4.4.4.3 CNF and TCP/IP-Based Internet in Routing

The goal of this section is to compare the performance of the proposed converged architecture (referred to as CNF architecture) with that of the traditional TCP/IP-based Internet architecture when the converged architecture uses storage-aware intelligent routing. To compare the performances of these two networks, Storage Aware Routing (STAR) scheme proposed for CNF architecture in Paul et al. [16] is compared with the traditional OLSR routing protocol in a 25-node network in two cases: (1) static network in 500 m-by-500 m grid with intermittently failing links, and (2) network in which the nodes move according to Truncated Levy Walk in $2,500 \times 2,500$ area.[28]

Figure 4.11 shows the number of files transmitted and delivered, average file transfer delays, file streaming throughput, and overall network throughput. File streaming throughput represents the average physical data rate used to transfer

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

(a)

Protocol	OLSR			STAR		
Mean Inter-arrival time (seconds)	50	10	1	50	10	1
Files transmitted/ Delivered	149/ 147	320/ 265	1016/ 691	165/ 164	468/ 438	1147/ 862
Average File Delay (seconds)	0.678	1.047	2.936	1.634	10.883	5.518
File Stream throughput (Mbps)	3.868	3.187	2.305	3.546	3.755	3.281
Network throughput (Mbps)	0.302	0.548	1.417	0.337	0.897	1.765

(b)

Protocol	OLSR		STAR	
Mean Inter-arrival time (seconds)	10	50	10	50
File Delivery Fraction	72.34	66.67	89.66	79.17
Average File Delay (seconds)	100.65	109.65	92.28	38.66
File Stream throughput (Mbps)	1.09	1.39	1.7	1.59

Figure 4.11. (a) Static network with intermittently failing links (b) Performance with Levy mobility model.

the file at every hop. Unlike the network throughput, streaming throughput does not include delays incurred in queues and storage.

From Figure 4.11(a) showing results for the static case (case 1), it can be observed that the STAR protocol is able to deliver a larger percentage of files that are admitted into the network, and this is because STAR selects faster paths for transmission. The average file delays and streaming throughput in STAR are larger showing the preference for storage over using slow transmission channels.

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

Although file delays are high, the cache-and-forward concept improves the overall network throughput.

From Figure 4.11(b) showing results for the mobile case (case 2), it can be observed that the file delivery fraction achieved by STAR is 17 percent and 33 percent more when the mean interarrival times are 10 and 50 seconds, respectively. The average file delays are much lower (or equivalently, the file stream throughputs are much higher) in STAR compared to OLSR. These results indeed justify the benefits of STAR in DTN like mobility models.

4.4.4.4 CNF and TCP/IP-Based Internet in Multihop Mobile Wireless Networks

The goal of this section is to compare the performance of the proposed converged architecture (referred to as CNF architecture) with that of the traditional TCP/IP-based Internet architecture when the network consist of multiple wireless hops in an end-to-end connection. To compare the performances of these two networks, a 49-node grid topology was considered under variety of traffic and noise patterns as shown in Figure 4.12 to observe the effect on average file transfer delay.[7]

Since TCP's performance over wireless without MAC layer reliability only becomes worse, the network capacity achieved by TCP/IP without MAC level reliability is not shown. It can be seen that for case 1, the network capacity offered by CNF Link Layer (LL) with MAC level ACKs is about 70 percent higher than that offered by TCP. Disabling the MAC level ACKs increases the CNF capacity gains to 140 percent over TCP. For case 2, the client server model,

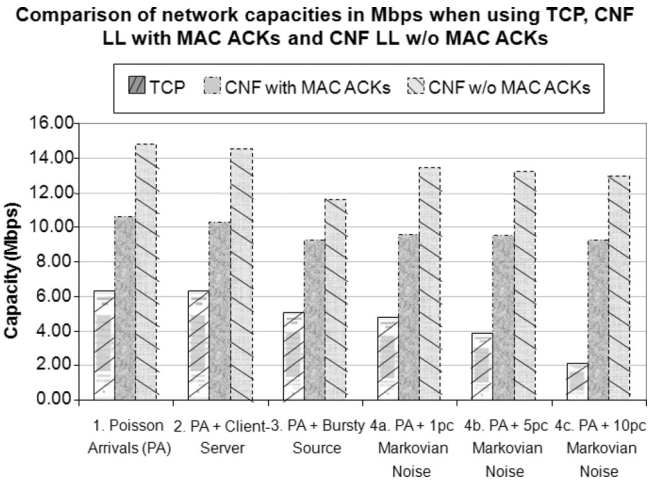


Figure 4.12. Comparison of network capacity (TCP vs. CNF LL).

https://doi.org/10.1017/CBO9780511921117.005 Published online by Cambridge University Press

the capacities achieved remain approximately the same for all three strategies, that is, TCP, CNF LL with MAC ACKs, and CNF LL without MAC ACKs. Hence the capacity gains of CNF over TCP also remain the same. For case 3, the bursty source model, it can be seen that noticeable reductions occur in the capacity achieved by the CNF LL protocol with and without MAC ACKs. This is because burstiness in traffic causes congestion at nodes. Link Layer queue sizes increase and the average file delay becomes longer. Capacity gains of 85 percent and 130 percent can be seen for CNF LL with MAC ACKs and CNF LL without MAC ACKs, respectively, over TCP. For case 4, where Markovian noise was introduced in the links, it should be noted that CNF is much more noise resilient as compared to TCP. For example, in 10 percent Markovian noise, TCP shows a two-third reduction in capacity whereas CNF LL without MAC ACKs suffers from 13 percent reduction in capacity. The capacity gains achieved by CNF LL over TCP in this case become about 650 percent.

4.5 Concluding Remarks

This chapter is motivated by networking scenarios driven by intermittent connectivity, content, and mobility that are not effectively handled by the traditional TCP/IP-based Internet. The proposed converged architecture to deal with these networking scenarios consists of in-network persistent storage and hop-by-hop reliable transport in the data plane, as well as name resolution, late binding, and routing in the control plane. Whereas these architectural components can be built as an overlay on the core networking (TCP/IP in the Internet) infrastructure, in a clean-slate network, these should be built into the core network itself. Given the exponentially dropping cost of processing/MIPS and storage/GB, it is highly conceivable that the network itself will consist of network elements (or future routers) with significant amount of persistent storage, and significant amount of processing power so that the route would be computed in real time at each node (rather than being computed in the background and the forwarding table used in real time for forwarding packets) based on multiple dimensions, such as congestion in the network, available storage in the network elements, availability of cached content, and so forth. Thus it makes a lot of sense for the next-generation clean-slate network architecture to encompass the support for intermittent connectivity, content, and mobility right in the network fabric because these themes together have a very broad scope in the overall area of networking.

References

[1] http://www.squidcache.org
[2] http://www.appliansys.com
[3] http://www.akamai.com

[4] http://www.limelightnetworks.com
[5] Delay-Tolerant Networking Architecture, IETF RFC 4838.
[6] Gopal, S., and Paul, S. 2007. TCP Dynamics in 802.11 Wireless Local Area Networks. *IEEE International Conference on Communications*, June.
[7] Saleem, A. 2008. Performance Evaluation of the Cache and Forward Link Layer Protocol in Multihop Wireless Subnetworks. Master's Thesis, Rutgers University, WINLAB.
[8] Krishnan, R., Basu, P., Mikkelson, J. M., Small, C., Ramanathan, R., et al. 2007. The SPINDLE Disruption-Tolerant Networking System. *IEEE Milcom*, 29–31 Oct., pages 1–7.
[9] Guo, S., Falaki, M. H., Oliver, E. A., UrRahman, S., Seth, A., Zaharia, M. A., and Keshav, S. 2007. Very Low-Cost Internet Access Using KioskNet. *ACM Computer Communication Review.* http://blizzard.cs.uwaterloo.ca/tetherless/images/c/c0/Kiosknet.pdf
[10] Guo, S., Falaki, M. H., Oliver, E. A., UrRahman, S., Seth, A., et al. 2007. Design and Implementation of the KioskNet System. *International Conference on Information Technologies and Development*, December.
[11] Balasubramanian, A., Mahajan, R., Venkataramani, A., Levine, B. N., and Zahorjan, J. 2008. Interactive WiFi Connectivity for Moving Vehicles. *Proc. ACM SIGCOMM*, pages 427–438.
[12] Banejree, N., Corner, M. D., Towsley, D., and Levine, B. N. 2008. Relays, Base Stations, and Meshes: Enhancing Mobile Networks with Infrastructure. *Proc. ACM Mobicom*, pages 81–91.
[13] Soroush, H., Banerjee, N., Balasubramanian, A., Corner, M. D., B. N., and Lynn, B. 2009. DOME: A Diverse Outdoor Mobile Testbed. *Proc. ACM Intl. Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet)*. Article No. 2.
[14] Hui, P. Chaintreau, A. Gass, R., Scott, J., Crowcroft, J., and Diot, C. 2005. Pocket Switched Networking: Challenges, Feasibility and Implementation Issues. *Proceedings of the Second IFIP Workshop on Autonomic Communications 2005*.
[15] Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., and Diot, C. 2005. Pocket Switched Networks and the Consequences of Human Mobility in Conference Environments. *Proceedings of the SIGCOMM 2005 Workshop on Delay Tolerant Networking*, pages 244–251.
[16] Paul, S., Yates, R., Raychaudhuri, D., and Kurose, J. 2008. The Cache-and-Forward Network Architecture for Efficient Mobile Content Delivery Services in the Future Internet. *Proceedings of the First ITU-T Kaleidoscope Academic Conference on Innovations in NGN: Future Network and Services*, pages 367–374.
[17] Dong, L., Liu, H., Zhang, Y., Paul, S., Raychaudhuri, D. 2009. On the Cache-and-Forward Network Architecture. *IEEE International Conference on Communications*, pages 1–5.
[18] http://www.youtube.com
[19] http://www.revver.com
[20] Bakre, A., and Badrinath, B. R. 1995. I-TCP: Indirect TCP for Mobile Hosts. *Proc. 15th Int'l Conf. on Distributed Computing Systems*, pages 136–143.
[21] Gopal, S., and Raychaudhuri, D. 2005. Experimental Evaluation of the TCP Simultaneous-Send Problem in 802.11 Wireless Local Area Networks. *ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, pages 17–22.
[22] Gopal, S., Paul, S., and Raychaudhuri, D. 2005. Investigation of the TCP Simultaneous-Send Problem in 802.11 Wireless Local Area Networks. *Proceedings of the IEEE Computer and Communications Conference (ICC)*, **5**, 3594–3598.
[23] Gopal, S., Paul, S., and Raychaudhuri, D. 2007. Leveraging MAC-Layer Information for Single-Hop Wireless Transport in the Cache and Forward Architecture of the

Future Internet. *The Second International Workshop on Wireless Personal and Local Area Networks (WILLOPAN)*, pages 1–6.

[24] Zhao, W., Ammar, M., and Zegura, E. 2004. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. *Proceedings of ACM Mobihoc 2004*, pages 187–198.

[25] Zhao, W., and Ammar, M. 2003. Message Ferrying: Proactive Routing in Highly-Partitioned Wireless Ad Hoc Networks. *Proceedings of the IEEE Workshop on Future Trends in Distributed Computing Systems*, pages 308–314.

[26] Liu, H., Zhang, Y., and Raychaudhuri, D. 2009. Performance Evaluation of the Cache-and-Forward (CNF) Network for Mobile Content Delivery Services. *IEEE International Conference on Communications (ICC) Workshops 2009*, pages 1–5.

[27] Lijun, D., Yanyong, Z., Sanjoy, P. Performance Evaluation of In-network Integrated Caching. Forthcoming.

[28] Shinkuma, R., Jain, S., Yates, R. 2009. Network Caching Strategies for Intermittently Connected Mobile Users. *IEEE PIMRC*, pages 1771–1775.

[29] Jain, S., Saleem, A., Liu, H., Zhang, Y., and Raychaudhuri, D. 2009. Design of Link and Routing Protocols for Cache-and-Forward Networks. *IEEE Sarnoff Symposium*, pages 1–5.

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

[31] Acharya, A., Ganu, S., and Misra, A. 2006. DCMA: A Label Switching MAC for Efficient Packet Forwarding in Multihop Wireless Networks. *IEEE JSAC Special Issue on Wireless Mesh Networks*, pages 1995–2004.

[32] Wu, Z., Ganu, S., and Raychaudhuri, D. 2006. IRMA: Integrated Routing and MAC Scheduling in Wireless Mesh Networks. *Proceedings of the Second IEEE Workshop on Wireless Mesh Networks, (WiMesh)*, pages 1–8.

[33] Akyildyz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. 2002. A Survey on Wireless Sensor Networks. *IEEE Communications Magazine*, pages 102–114.