UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FINAL DE CARRERA

EXPLORING THE BENEFITS OF MULTIPATH TCP IN WIRELESS NETWORKS

AUTOR: DAVID CEREIJO FREIRE

TUTOR: JOSE IGNACIO MORENO NOVELLA

5 de Enero de 2016

Acknowledgements

I would like to thank my thesis supervisor Dr. Jose Ignacio Moreno Novella for his support during the writing of this document and in the project itself.

I am also very grateful to T-Labs for their support and the opportunity they granted me to develop my thesis in such a healthy and passionate environment. Thanks in particular to Dr. Nico Bayer, whose expertise and ideas have lighten this work and to my colleague Roman Szczepanski for his daily support and cooperation.

Thanks to my mate Irene Romero, who worked side by side with me for six months in all the stages of this project. In what I am concerned, she is a part of this outcome just as important as I am.

Finally, a loving mention to my family for standing by my side for so many years, for they were there for me even more intensely in the bad times than in the good ones. I do not know whether this work of mine worth it to be read, but I do know it is because of them.

Abstract

The revolution of the information society has created a completely new situation in the telecommunications markets. As the average user data demands in today's society grow bigger, since users nowadays are demanding a faster, wider and more reliable communication service from the operators so they can watch more videos, listen to more music or access the Internet in general with a better quality, a lower latency and seamlessly to the network access they are using, the network operators face the challenge to fit this demands into their existing networks. This has forced the operators to think in terms of how optimal they are on providing their services if they want to fulfil the customer requirements in this new environment.

At the same time we need to keep in mind that simultaneously to this new user's habits smartphones revolution has created, it has also made it possible to have accessible communication devices which have the necessary hardware and horsepower to keep different network interfaces up, and so it has become a common thing to reach the Internet via different kind of networks along the day. Even more it has enabled a rich communications environment where different connection possibilities are available to the user at the same time.

In this context, the idea of multipath communication emerges. The idea of taking advantage of a dense wireless communication offer through the use of multipath (sending and receiving information through different network interfaces simultaneously) looks promising to overcome a situation where user's communications services demand grows and at the same time the mobile network load becomes stronger. The newfangled protocol Multipath TCP (MPTCP) is a technology which is enabling in practice this king of multipath communication, and it is the focus of this project to dig into possible benefits the protocol may bring to the table by defining a set of use cases, test-bed implementations and experiments with MPTCP which we present and analyse in this document.

Resumen

Introducción

La revolución de la sociedad de la información ha creado una situación que es completamente nueva en los mercados de telecomunicaciones. A medida que el usuario medio aumenta su demanda de datos, ya que hoy en día los hábitos de estos pasan por conexiones más rápidas y fiables que les permitan reproducir contenido (video, música, páginas web) con mejor calidad, menor latencia y transparentemente a la red que estén utilizando, los operadores de red afrontan nuevos retos a la hora de encajar estas expectativas del usuario dentro de las posibilidades que ofrece la red. Esto está forzando a los operadores a buscar una manera más óptima de gestionar el tráfico de sus clientes para así poder satisfacer la demanda de unos servicios de mayor calidad que estos realizan.

Al mismo tiempo hay que tener en mente que, de la misma manera que el impacto que esta esta revolución de los smartphones ha tenido en los hábitos de consumo del usuario ha creado nuevos y complejos problemas, también ha hecho posible que existan dispositivos económicamente accesibles para el público con el hardware y la capacidad de procesamiento necesarias para incorporar múltiples adaptadores de red, y esto a su vez ha llevado a al escenario actual en el que comúnmente coexisten en el mismo lugar diferentes posibilidades para conectarse a internet (típicamente Wi-Fi y conexión móvil, pero también podríamos nombrar tecnologías como el Bluetooth o la clásica conexión de Ethernet en ordenadores portátiles)

La transmisión multi-trayecto: Multipath TCP

Es en este contexto en el que surge la idea de la comunicación multi-trayecto. La idea de aprovechar un entorno con una densa pero heterogénea oferta de conexión a través del uso del multi-trayecto (enviar y recibir información a través de múltiples interfaces de red simultáneamente) aparece como una posibilidad prometedora para los operadores para mejorar la experiencia del usuario al mismo tiempo que se gestiona el tráfico en la red de una manera más eficiente.

El protocolo experimental Multipath TCP es una extensión del TCP clásico que hace posible este uso simultáneo de múltiples interfaces para la comunicación, y es objetivo de este proyecto diseñar, implementar y testear el protocolo en diferentes casos de uso en los que el multi-trayecto ofrece, a priori, algunas ventajas. En las siguientes páginas explicaremos que casos de uso hemos elegido para probar el protocolo y por qué, cómo hemos diseñado e implementado los bancos de pruebas y que resultados hemos obtenido en nuestro experimentos sobre el rendimiento del protocolo, realizando al mismo tiempo un análisis crítico de los resultados de los resultados.

Sobre el proyecto

Este Proyecto ha sido desarrollado en T-Labs Berlin bajo el programa Erasmus Placement, con la supervisión del Dr. Nico Bayer y la estrecha colaboración de nuestro compañero Roman Szczepanski del departamento de Seamless Network Control (SNC). T-Labs Berlin es una de las cuatro localizaciones que Detusche Telekom AG tiene para propósitos de innovación y desarrollo en diferentes lugares del mundo. Los otros laboratorios están en Bonn (Alemania), Be'er Sheva (Israel) y Mountain View (Estados Unidos). Durante nuestros seis meses de trabajo en T-Labs, nuestro equipo desarrolló las investigaciones con MPTCP que se recogen en este trabajo.

Un ejemplo para la reflexión

Cuando un usuario medio se despierta por la mañana en su casa y revisa la bandeja de entrada de su correo electrónico, probablemente lo esté haciendo conectándose a un punto de acceso Wi-Fi en su domicilio. Cuando sale de su casa para ir al trabajo, la señal que recibe irá degradándose a medida que se aleje del punto de acceso, hasta que llegará un momento en que la comunicación a través de Wi-Fi ya no será posible. Eventualmente su teléfono detectará esta situación, y en ese momento su Smartphone realizará una entrega de las conexiones existentes a la red móvil, de manera que el usuario pueda mantenerse online en todo momento. Cuando llegue a su puesto de trabajo, probablemente quiera que sus conexiones vuelvan a entregarse a un punto de acceso Wi-Fi de la empresa, para así optimizar su consumo de datos móviles, además de poder acceder a servicios de la red interna de la empresa. Incluso es posible que más de una red Wi-Fi esté disponible al mismo tiempo (por ejemplo que haya una para empleados y otra para visitantes) o que también sea posible acceder a través de una conexión cableada en el caso de su PC de trabajo. Lo que intentamos ejemplificar con este caso, es que hoy en día diferentes tecnologías de acceso conviven y cooperan para proveer a los usuarios del mejor servicio de comunicación posible en cada momento. Surge la pregunta: ¿Lo están haciendo de una manera óptima?

La proliferación de tantos tipos de tecnologías de acceso en el día a día, ha sido realimentado por el rápido desarrollo también de los interfaces de red necesarios en los terminales para acceder a ellas. Se ha vuelto fácil y barato incorporar múltiples interfaces de red en los dispositivos de los usuarios, y así es difícil encontrarse hoy un Smartphone en el mercado que no cuente con un interfaz móvil para conexiones 3G/4G además de interfaz Wi-Fi e incluso Bluetooth. Estos dispositivos multi-interfaz están por todas partes hoy en día. Los teléfonos móviles son ejemplos muy evidentes de ello, pero un portátil con interfaz Wi-Fi y tarjeta de Ethernet o un servidor web, que es accesible desde múltiples interfaces al mismo tiempo también son máquinas multi-interfaz.

Armónicamente a esta tendencia, la importancia del acceso sin cables a internet ha experimentado un crecimiento acusado en los últimos años, con un gran repunte tanto en el número de puntos de acceso Wi-Fi como en el de suscriptores de un plan de datos móviles. En general, estas dos tecnologías se reparten de manera natural las preferencias de tipo de conexión de los usuarios a la hora de acceder a internet, con un acuerdo implícito en que Wi-Fi es para de las necesidades de acceso estáticas/más locales mientras que el acceso móvil es para la calle. Además, si nos adentrásemos en el espectro de tecnologías Wi-Fi existentes, veríamos como dentro de las WLAN también existen varias tecnologías de acceso diferentes y que se pueden presentar al mismo tiempo (tenemos por ejemplo WLAN en la banda de los 2,4GHz y en la de 5GHz). La situación descrita ilustra la realidad de que existe una "oferta" diversa de métodos de acceso a la red para el usuario que, generalmente, puede acceder a internet a través de diferentes canales al mismo tiempo.

Al mismo tiempo, como ya explicábamos en el preludio, los usuarios demandan cada vez conexiones más rápida y fiables. Si tenemos un acceso Ethernet en nuestro ordenador portátil que ofrece una velocidad de 100 Mbps y también podemos conectarnos a una red Wi-Fi que da hasta 30 Mbps, ¿Por qué no deberíamos esperar que nuestra conexión a internet tuviera hasta un máximo de 130 Mbps? Si estamos realizando una video llamada vía 3G desde nuestro teléfono móvil y necesitamos un extra en la velocidad de la conexión para tener imagen en calidad HD, ¿Por qué no podemos complementar nuestra conexión con Wi-Fi para así obtener el ancho de banda necesario para ello? Todo esto manteniendo reglas en el lado del usuario que eviten que este haga un uso innecesario de la red móvil, más cara y habitualmente más congestionada. En este contexto, el concepto de traffic bundling resulta útil.

El traffic bundling

El traffic bundling es una técnica que consiste en realizar una redistribución de los paquetes correspondientes a un flujo, en una red de conmutación de paquetes, en diferentes sub-flujos. Típicamente se utilizará para repartir los paquetes que discurren a través de un camino entre varios. El concepto está muy relacionado con el de agregado de ancho de banda, que busca combinar la capacidad de diferentes enlaces en uno que sea la suma de todos. Dependiendo del punto en la pila de protocolos en el que se implementa, varía el protocolo o tecnología que lo realiza. Para el propósito de este trabajo, nos hemos centrado en el traffic bundling de nivel 4 o nivel de transporte, y la tecnología que lo implementa en este caso es MPTCP.

¿Por qué MPTCP?

La principal razón por la que se desarrolla el protocolo MPTCP es la limitación de TCP clásico a la hora de manejar múltiples flujos de datos. El protocolo original tan sólo permite una conexión por dupla TCP (dirección IP + puerto). Como el encaminamiento en redes IP se basa precisamente en la IP y, en algunas situaciones, también en el puerto, no es posible conseguir que un flujo de paquetes perteneciente a una misma conexión TCP fluya por varios caminos distintos simultáneamente y por tanto no es posible el agregado de enlaces. Además, TCP no provee de un sistema de señalización entre los pares, de tal manera que uno pueda notificar al otro de un cambio que se haya podido producir en su dirección IP, o la disponibilidad de una nueva

IP/interfaz de red para la comunicación en paralelo. Estos dos inconvenientes hacen que TCP clásico no soporte nativamente la transmisión multi-trayecto, y hacen necesaria su evolución para convertirlo en una opción óptima para la transmisión con dispositivos multi-interfaz. Esta evolución es lo que pretende ser MPTCP.

El modelo de MPTCP

MPTCP es un protocolo del nivel 4 o transporte, lo cual quiere decir que se sitúa entre el nivel de red (típicamente IP) y el de aplicación. De manera más abstracta, MPTCP se puede ver como una capa insertada directamente sobre TCP y bajo el nivel de aplicación. Así, una conexión MPTCP hará uso de múltiples conexiones TCP, que manejará internamente, para enviar y recibir segmentos de datos entre aplicaciones. Aunque en la mayoría de los casos se utilizará una conexión TCP por par de interfaces, en realidad el protocolo permite establecer tantas sub-conexiones por par de direcciones IP como se quiera.

Una conexión MPTCP requiere de entidades que soporten el protocolo en ambos extremos. Esto es así porque es este nivel el que implementa la inteligencia para reordenar los paquetes enviados a través de los diferentes sub-flujos y entregarlos a la aplicación. Esta condición implica que a la hora de desplegar el soporte MPTCP serán, en principio, necesarios cambios tanto en los servidores como en los dispositivos finales.

Objetivos de MPTCP

La actual implementación del protocolo se guía por tres objetivos, que son la mejora de la velocidad de transmisión con respecto a TCP clásico, no dañar el tráfico TCP que vaya a coexistir con transmisiones MPTCP y el compromiso con el balance de la congestión entre sus caminos activos. Además, la primera implementación oficial del protocolo se hizo de manera que el nuevo nivel MPTCP resultase transparente para el nivel de aplicación. Se intentaba con esto que la implementación fuese compatible hacia atrás y que no forzase a la reescritura del código de las aplicaciones que fuese a correr sobre máquinas con kernel MPTCP. La recomendación es que cualquier nueva versión del protocolo que se realice se haga siguiendo estas reglas. En la memoria de este proyecto está descrito de manera detallada el modo de operación de MPTCP.

Itinerancia con asistencia multi-trayecto

Como ya se ha dicho, el objetivo de este proyecto ha sido desarrollar diferentes entornos de pruebas para el testear el rendimiento de la transmisión multi-trayecto en general y del protocolo MPTCP en particular. Para ello se propusieron una serie de casos de uso, para cada uno de los cuales se implementó un banco de pruebas en el laboratorio, sobre los que a continuación se ejecutaron una serie de experimentos de interés.

El primer caso de uso que se propone es la asistencia por una transmisión multi-trayecto a la itenerancia entre redes. Comúnmente en una situación de movilidad, un cliente necesita migrar sus conexiones desde un punto de acceso físico a otro que presenta una mejor cobertura. Este es el caso en telefonía cuando un usuario se mueve de una célula a otra, o en grandes despliegues Wi-Fi que usen muchos nodos para cubrir una cierta área. Incluso es posible también (y de hecho es probable) que un usuario necesite migrar sus conexiones desde una red WLAN a una red móvil o viceversa en lo que se conoce como entrega vertical. En transmisión mono-trayecto (TCP clásico), cualquiera de estos eventos implica una interrupción en el flujo de datos mientras el nivel de enlace se reasocia con un nuevo punto de acceso (entrega horizontal) o establece la conexión desde un nuevo interfaz (entrega vertical). Uno de los grandes retos en movilidad es reducir el impacto de este estado transitorio en la experiencia de usuario. En un escenario como el descrito, MPTCP puede ser utilizado para asistir el proceso de entrega, incluso cuando esta se produce entre tecnologías de red diferentes en una entrega vertical. Con la idea de probar esta posible aplicación del protocolo, se diseñaron dos bancos de pruebas en el laboratorio.

En una primera instancia se comparó el rendimiento de MPTCP con el de TCP en un despliegue Wi-Fi en una de las plantas del edificio de T-Labs. Para ello se establecieron dos redes Wi-Fi redundantes con varios puntos de acceso en esta planta, utilizando dos frecuencias distintas, 2,4 GHz y 5 GHz. A continuación, con un ordenador portátil con soporte MPTCP y dos interfaces Wi-Fi se inició una transmisión multi-trayecto en la que uno de los flujos utilizaba la red de 2,4 GHz y el otro la de 5 GHz, y se midió el rendimiento de la conexión general cuando nos movíamos a lo largo de los pasillos y del rango de cobertura de los diferentes puntos de acceso que formaban la red. La idea era que, gracias a la conexión redundante que nos ofrecía la transmisión multi-trayecto, las fisuras en la transmisión global se redujesen y la velocidad de transmisión media y máxima aumentasen. Para comparar ambos protocolos, un experimento similar se realizó con TCP clásico, utilizando por tanto una transmisión mono-trayecto. El resultado de los experimentos fue que, efectivamente, el uso de múltiples interfaces de red para la transmisión estabilizó la calidad de la conexión general, que tuvo, para el caso de MPTCP, un mayor ancho de banda medio y un menor número de fisuras.

Sin embargo durante los experimentos se detectó que este banco de pruebas era muy complejo, al utilizar muchos nodos de red y forzar durante las pruebas un gran número de eventos de reconexión WLAN. Se decidió que para caracterizar con mayor precisión el rendimiento de MPTCP, principalmente en lo que respecta a la ganancia en el tiempo de entrega con respecto a TCP, se necesitaba un banco de pruebas que fuese más estable y que dejase fuera de la ecuación la imprevisibilidad que afectaba a la anterior. Para ello se diseñó un experimento en el que el cliente estaría estático y permanentemente conectado con sus dos interfaces a dos puntos de acceso equidistantes, que estarían en distintas frecuencias. La situación de movilidad se simularía ahora en el segmento de la red que une a los puntos de acceso con el servidor, inutilizando periódicamente los caminos utilizados por las transmisiones correspondientes a cada uno de los dos interfaces de red. De esta manera pudimos emular con una estabilidad aceptable las situaciones de entrega suave (ambos caminos están disponibles y la entrega de los datos puede realizarse gradualmente), entrega dura (la transmisión deber ser redirigida inmediatamente desde un interfaz a otro) y agregado de acceso (ambos enlaces está disponibles y por tanto MPTCP debe agregar su capacidad). En el documento se encuentran detallados los resultados obtenidos para este experimento, que además se pueden comparar con los obtenidos en un experimento similar realizado con TCP clásico. Se podrá apreciar cómo MPTCP mejora ampliamente el rendimiento de TCP en la situación analizada en lo que respecta al tiempo de entrega, llegando a ser este de un mínimo de 50ms para una situación de entrega suave. En cambio en el agregado de capacidad el protocolo no muestra un comportamiento tan bueno como cabría esperar, empatando con TCP en este aspecto e incluso empeorándolo en algunos casos.

La descarga de la red móvil: Otra ventaja de usar MPTCP

Una aplicación distinta que hemos identificado del protocolo se deriva de una de las características de MPTCP, expresada en una de sus tres reglas básicas de diseño; ésta es su capacidad para hacer balance de la congestión presente a lo largo de sus sub-flujos. Cualquier implementación de MPTCP, y por descontado la implementación oficial que la de referencia en este trabajo, incluye uno o varios mecanismos de control de congestión que realizan balance de carga de la transmisión entre sus sub-flujos. Esta cualidad es, a priori, muy interesante para redirigir el tráfico desde la red móvil, habitualmente muy congestionada, hacia otras redes (el ejemplo más claro son las redes Wi-Fi) de una manera eficiente, rápida y transparente para el usuario. En un caso ideal, si un usuario que estuviese utilizando la red móvil entrase en rango de una red Wi-Fi, MPTCP detectaría inmediatamente que hay un canal para la comunicación que está menos congestionado y redirigiría gradualmente el tráfico hacia esa red, liberando la red móvil para usuarios que no tuviesen más opción disponible para el acceso a internet. Esta posibilidad sería beneficiosa tanto pare el operador de red, que vería como su red móvil alivia su congestión, como para el usuario, que se ahorraría así el coste asociado al uso de la red móvil, habitualmente bastante caro y vería racionalizado su gasto. Con el propósito de comprobar cuan eficiente sería MPTCP en estos casos, se diseñó el siguiente banco de pruebas.

Para el testeo del rendimiento de MPTCP en el balance de carga entre red móvil y WLAN, se propuso un banco de pruebas consistente en un Smartphone Android con el kernel experimental de MPTCP que realiza diferentes descargas a través de la red móvil 4G. Ocasionalmente y de manera controlada, hacemos disponible una red Wi-Fi en cuya área de cobertura también se encuentra el cliente. Medimos con que intensidad y rapidez es MPTCP capaz de desviar el tráfico hacia Wi-Fi cuando este está presente. Los resultados obtenidos no fueron del todo satisfactorios, puesto que aunque descubrimos que MPTCP era bastante rápido en detectar la disponibilidad del nuevo canal de transmisión, la cantidad de tráfico que se redirigía hacia la red WLAN nos parece insuficiente. MPTCP tarda del orden de 2-3 segundos en reaccionar a la aparición de una nueva vía para la transmisión, lo cual es más que aceptable, pero encontramos que, en nuestro caso, en presencia de Wi-Fi la carga sobre la red móvil tan solo se reducía en torno a un 40%, cuando el canal Wi-Fi tenía la capacidad suficiente para que ésta fuese descargada por completo. Sin embargo, a pesar de estos resultados somos optimistas, ya que sospechamos que esta limitación en la práctica para desviar una cantidad de tráfico mucho mayor tiene que ver con la elección del algoritmo de control de congestión para MPTCP. Tal y como se explica en detalle más adelante en el documento, MPTCP utiliza diferentes algoritmos para ligar el estado de congestión de cada uno de los sub-flujos de una conexión y así realizar balance de carga. Estos algoritmos hacen que la tasa de transmisión a la que convergen cada uno de los sub-flujos se una función tanto del RTT para ese camino como de las pérdidas del enlace. En ninguno de esto dos aspectos Wi-Fi es claramente dominante sobre LTE, y como resultado el algoritmo de control de congestión vuelca por completo el tráfico sobre Wi-Fi. Sin embargo, creemos que esta situación se puede subsanar incorporando nuevos criterios al algoritmo de control de congestión de MPTCP, que den más relevancia al RTT que a la tasa de pérdidas, por ejemplo, o que introduzcan control de congestión con tipos de interfaz prioritarios (en este caso Wi-Fi sobre LTE).

MPTCP como tecnología posibilitadora de la Comunidad DSL

Una tercera aplicación analizada surge como fruto de una idea original de T-Labs Berlin llamada la Comunidad DSL. La velocidad de transmisión que proporciona la típica línea DSL del bucle de abonado es muy sensible a la distancia a la central. Por culpa de esto, en zonas rurales y suburbanas, donde esta distancia suele ser relativamente alta, a menudo la velocidad de las conexiones individuales en estas zonas es limitada. En Reino Unido, la velocidad media de la línea DSL en zonas urbanas es de 28 Mbps, mientras que para zonas rurales es sólo de 10 Mbps, lo cual lleva una experiencia de navegación de baja calidad para los usuarios de estas líneas. Por otro lado, la el uso de las líneas DSL típicamente bajo (entendido como la cantidad de tiempo que se utiliza sobre el total) y el tráfico a rachas que genera el uso para navegación de estas líneas crea una situación contradictoria en la que pareciese que se está desperdiciando recursos de transmisión cuando estos son ya de por si escasos. El concepto de la Comunidad DSL de T-Labs busca agregar los accesos DSL de varios residentes de una misma zona mediante la creación de una malla de nodos Wi-Fi que distribuyan el tráfico de la comunidad entre todos los accesos de una manera más óptima. De esta forma, cuando un miembro de la comunidad DSL no esté utilizando su línea, esta podrá ser asignada a otro usuario que sí quiera conectarse a internet en ese momento y viceversa. Gracias a su capacidad para dividir la transmisión en varios sub-flujos, MPTCP se presenta como una tecnología capaz de hacer posible esta Comunidad DSL. Por este motivo se diseñó también un entorno de pruebas emulando una Comunidad DSL y se realizaron experimentos con MPTCP sobre él.

A grandes rasgos, el entorno de pruebas que se diseñó e implementó consistió en la instalación de una malla simple de nodos Wi-Fi que creaban hasta tres posibles caminos para la transmisión, cada uno de los cuales ofrecía una capacidad máxima de 10 Mbps. Con la configuración de rutas adecuada, el tráfico proveniente de un cliente MPTCP conectado a uno de los nodos se enrutaría a través de los tres caminos simultáneamente, permitiendo que éste aprovechase la capacidad agregada de los enlaces para la transmisión (30 Mbps). Los resultados obtenidos hay que separarlos para los dos sub entornos de pruebas realizados en la práctica.

Igual que para la primera aplicación que se testó (Itinierancia entre redes asistida por multi-trayecto), la realización del experimento para MPTCP sobre Comunidad DSL se dividió en la práctica en dos etapas. En una primera parte, la conexión entre los nodos pertenecientes a la malla de puntos de acceso Wi-Fi se realizó en frecuencias distintas, y la conexión entre el cliente y la Comunidad DSL directamente con cable Ethernet. Con esta implementación se buscaba aislar lo más posible los caminos de transmisión, para tener unos primeros resultados tan libres de interferencias como fuese posible. Para esta primera instalación, los resultados fueron satisfactorios, alcanzándose los esperados 9 Mbps en el cliente cuando sólo se utilizaba un camino, aproximadamente 15 Mbps cuando se usaron dos caminos y 26 Mbps cuando se transmitió simultáneamente por los tres caminos disponibles.

Sin embargo, la presunción de que en la Comunidad DSL los nodos de la red de malla puedan estar conectados en frecuencias distintas y el cliente vaya a tener acceso cableado a la comunidad en todo momento. La situación que más probablemente tendremos en la realidad será que todos los nodos Wi-Fi de la red de malla que forman la Comunidad DSL estará conectados en la misma frecuencia e incluso el mismo canal (probablemente en modo "mesh" o "ad hoc") y los clientes se conectarán a la comunidad utilizando también un enlace Wi-Fi, aunque en este caso el enlace si podría ir en un canal o incluso frecuencia distinta. Para probar esta situación, se modificó el entorno de pruebas para que todos los nodos de la Comunidad DSL estuviesen en la misma frecuencia (2,4 GHz) y el cliente se conectó a través también de un enlace Wi-Fi en la banda de los 5 GHz. Desafortunadamente en esta nueva implementación los resultados observados no fueron tan buenos, ya que en ningún caso se superó los 10 Mbps correspondientes a la capacidad de un camino. El motivo, analizado más en detalla en la memoria, es un problema de nodo escondido en la red de malla. La situación de nodo escondido es una problema típico de las redes WLAN en el que el protocolo de compartición del medio físico para la transmisión falla, y se producen interferencias en la red que afectan drásticamente al rendimiento de las conexiones. En este caso se detectó, y esta documentada, una situación severa de interferencias causadas por el problema de nodo escondido en los flujos que discurrían por la red mesh de la Comunidad DSL, tanto en flujos de subida como de bajada. Este es un problema del nivel de enlace inherente a una situación como la propuesta, y creemos que puede ser una limitación técnica importante a la hora de llevar la Comunidad DSL a la práctica.

Para concluir

Como conclusión final, aclarar que los conceptos e ideas analizadas en el documento son propuestas de desarrollo de T-Labs, y que nuestro trabajo como estudiantes consistió en la definición de los entornos de pruebas concretos, su implementación utilizando los medios proporcionados y la ejecución de los experimentos y obtención de las medidas y resultados que se aquí se presentan. Durante nuestro trabajo hemos encontrado que le protocolo analizado muestra ventajas, algunas muy amplias, con respecto a TCP clásico en nuestros tests. Es especialmente destacable la capacidad de MPTCP para realizar entrega vertical del tráfico con un impacto tan pequeño en fisuras de la conexión como 50 ms, lo cual lo hace una alternativa sólida para utilizar en aplicaciones de tiempo real en situaciones de movilidad. El protocolo ha mostrado también un buen rendimiento en cuanto a los máximos de velocidad de conexión observados, si bien ha sido inestable en tasa de transmisión y esto ha llevado a que el comportamiento en media no haya sido todo lo bueno que se esperaba. Sobre las ventajas del protocolo a la hora de contribuir a la descongestión de la red móvil, podemos afirmar que es posible, aunque su rendimiento en este sentido no fue el óptimo. Como tecnología de Comunidad DSL, MPTCP ha mostrado también un rendimiento cercano al ideal cuando las limitaciones técnicas al problema que ya se explicaron no estaban presentes. El protocolo deber ser una alternativa a considerar para utilizar sobre este tipo de redes cuando si estos problemas pueden ser solventados.

Contents

1. Introduction **1**

1.1. Motivation 2

1.1.1. Control Plane Policies 4

1.1.2. Data Plane Policies 5

1.2. Goals and project phases 6

1.3. Document Structure 7

2. State-of-the-Art **8**

2.1. Traffic Bundling and Multipath 9

2.2. Multipath TCP 10

2.2.1. Model 11

2.2.2. Operation 13

2.2.3. Congestion 15

2.2.4. Configuration 17

2.2.5. Present status 18

2.3. Wireless Networks 20

2.3.1. WLAN Networks 20

2.3.2. Mobile Networks 24

2.4. MRAT 25

2.5. Monitoring tools 26

2.5.1. Data collection and analysis tools 26

2.5.2. Collection procedures 29

3. Test-bed description **31**

3.1. Multipath-Assisted Roaming (MAR) . . . 32

3.1.1. Dynamic MAR . . . 33

3.1.2. Static MAR . . . 35

3.2. Mobile Traffic Offloading . . . 40

3.3. Capacity Aggregation in Mesh Networks . . . 42

4. MPTCP Performance Evaluation **47**

4.1. Dynamic MAR . . . 47

4.1.1. Results . . . 47

4.2. Static MAR . . . 49

4.2.1. Results . . . 49

4.2.2. Conclusions . . . 51

4.3. Traffic Offloading . . . 53

4.3.1. Results . . . 53

4.3.2. Conclusions . . . 54

4.4. Mesh Capacity Aggregation . . . 54

4.4.1. Conclusions . . . 56

5. Personal conclusions and Future Works **58**

.1. Summary . . . 58

.2. Personal Conclusions . . . 59

.3. Future Work . . . 60

A. Budget **63**

A.1. Project planning . . . 63

A.2. Equipment expenses . . . 64

A.3. Human resources expenses . . . 65

B. Equipment **66**

B.1. Hardware . . . 66

B.1.1. Network Nodes . . . 66

B.1.2. Wireless Adapters . . . 67

B.2. Software 68
B.2.1. OS Platforms 68
B.2.2. Drivers 68

List of Figures

1.1. T-Labs Berlin. 1

1.2. There exists a wide network access offer today. 3

2.1. Traffic bundling technique. 9

2.2. The MPTCP model. 11

2.3. MPTCP-in-the-middle architecture. 12

2.4. A multipath connection should not take up capacity form a coexisting single path connection. 13

2.5. The MPTCP congestion control tries to move away traffic from congested to uncongested links [16]. 13

2.6. MPTCP establishing and joining mechanisms. 14

2.7. MPTCP announcing mechanism. 15

2.8. L2 Wi-Fi Handover 22

2.9. L3 Wi-Fi Handover 23

2.10. The MRAT network. 25

2.11. Node 75 on start-up. 26

2.12. The bwm-ng tool can monitor the physical layer throughput of each interface. . . . 27

2.13. Speedometer provides physical layer throughput monitoring with a better visualization. 27

2.14. Output of an iperf test. 28

2.15. The wget client can be used to download files using MPTCP. 28

3.1. MPTCP-assisted roaming. 32
3.2. Dynamic MAR Set-up. 34
3.3. Static Spiderman Set-up. 36
3.4. Emulated mobility in Static MAR 38
3.5. Offloading Spiderman Setup. 41
3.6. The DSL Community aims to provide a better user experience by using cooperating Wi-Fi mesh networks [47]. 43
3.7. Nodes choice for the Capacity Aggregation in Mesh Networks test-bed. 44
3.8. The Capacity Aggregation in Mesh Networks sub-scenario one. 45
3.9. The Capacity Aggregation in Mesh Networks sub-scenario two. 45

4.1. Dynamic MAR results 48
4.2. Throughput test in Static MAR set-up with MPTCP. 50
4.3. Comparison of two different Recovering Flow events. 52
4.4. Static Spiderman results 52
4.5. Traffic Offloading results 53
4.6. Throughput offered by the DSL Community for the central node only, the lateral nodes only and all nodes available. 55
4.7. Sub-scenario two was potentially a Hidden Terminal situation. 56
4.8. Hidden node transmission impact in performance. 57

A.1. Gantt diagram of the project. 64

B.1. Fujitsu Laptop 66
B.2. Acer Desktop 67
B.3. Network Node 67
B.4. Google Nexus 5 67
B.5. External wireless card. 68

Chapter 1

Introduction

This project was developed inside T-Labs Berlin under an Erasmus Placement Programme stay, with the supervision of Dr. Nico Bayer and in close collaboration with our mate Roman Szczepanski of the Seamless Network Control department. T-Labs Berlin is one the four different locations worldwide Deutsche Telekom AG owns for innovation and development purposes, others are in Bonn (Germany), Be'er Sheva (Israel) and Mountain View (U. S.). During our six months time in T-Labs our team took forward the research and experiences with the Multipath TCP protocol which are presented in detail in this work.

Deutsche Telekom
Laboratories

Figure 1.1: T-Labs Berlin.

1.1. Motivation

When a user wakes up in the morning at his place and check his e-mail, he is probably doing so by connecting to a domestic Wi-Fi access point. As he walks out and moves away from it, the signal will decade and eventually be lost therefore data connections will no longer be able to take place over Wi-Fi. In such case, the user's smartphone will generally detect this drop in the primary WLAN quality and react by turning on the mobile interface and connecting to the mobile network to stay on-line. The moment he arrives to his workplace, he will most probably want his data connections to be handed over to an enterprise Wi-Fi network in the building to avoid unnecessary use of his mobile plan. It is even possible that more than one Wi-Fi network is available (for instance one for employees and one for guests) or that access is available via different Wi-Fi technologies (A, G and N). What this example is trying to show is how heterogeneous networks are cooperating today in delivering communication services to the users. One question raises: Are they doing it optimally?

This proliferation of types of networks in the daily scene, has been followed by a fast development of the commercial availability of wireless access technologies for the average user, meaning it has become cheap and easy to incorporate different types of wireless interfaces into devices. Today it is difficult to find a smartphone in the market which does not provide Wi-Fi and Bluetooth interfaces in addition to that of mobile communications. This multi-homed devices, which have multiple network interfaces and thus are able to connect to different networks at the same time are everywhere today. Smartphones are a clear example of multi-homed devices but they are not the only case; for instance a laptop with both a Wi-Fi and Ethernet card and a server on the Internet which is accessible through different interfaces (as it is the case most of the times) are also multi-homed devices.

Harmonically to this trend, the importance of wireless access to the Internet have experienced a dramatic growth along the past years, with expectations of the total number of mobile subscribers for 3G and 4G services passing form 2.6 billion in 2014 to 6.1 billion in 2020 [1] and it can also be noted how the number and density of Wi-Fi access points is increasing both in the domestic and professional environment (For instance, the number of public Wi-Fi networks have grown 270% in the past two years, according to iPass watcher [2]). This context shows the importance and penetration of wireless access to the Internet today and it is also showing how two different wireless technologies, such as Wi-Fi and mobile, are friendly splitting the share

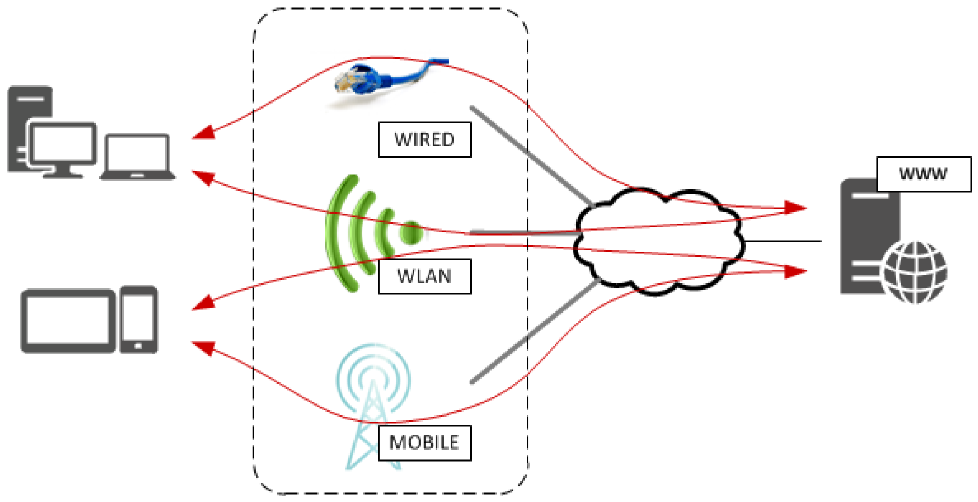


Figure 1.2: There exists a wide network access offer today.

of the Internet access, with both players agreeing on Wi-Fi taking up local, more static, access needs whilst mobile networks are for the road. Moreover, if we step in Wi-Fi access solely, different specifications are available as well (IEEE has defined standards for WLAN support in 2.4 and 5 GHz and the steps for a new standard in the 60 GHz frequency band have been taken [3]). Such high wireless access diversity and penetration have made it very common that a user have at the same time different ways or paths to reach one peer in the Internet. It could be a mobile connection and a Wi-Fi connection, different Wi-Fi access points available on different (or the same) frequency channel or even conventional wired access: It stands out that there is a diverse access "offer" existing in daily life today and, in general, multiple paths available for connecting network hosts.

At the same time, the users are increasingly demanding faster and trustful communication services. As users, we want the best possible performance regarding connectivity and Internet access. If we have Ethernet access in our desktop system delivering a speed of 100 Mbps and we can also connect to an access point topping 30 Mbps, why shouldn't we expect an overall maximum download speed of 130 Mbps? If a video call in our phone is taking place over Wi-Fi, but we need an extra boost in the connection speed to maybe have an HQ video call, why can't we use our 3G connection to complement Wi-Fi and provide a better overall quality? All this by keeping user side rules to avoid unnecessary usage of the more expensive and usually more congested mobile connection. It is in this context that the traffic bundling concept becomes useful.

Traffic bundling enables a path-aware allocation of the data streams generated by the user applications. It allows distributing traffic among the available paths so link diversity can be exploited. Remembering the example above, this technique would allow to send/receive part of the traffic in the Wi-Fi link (one sub-flow) and part in the mobile link (other sub-flow), and recombine them at the end into a single, faster one which is presented to the user. In general, in order to achieve this, two kinds of adjustments are required in the end devices: we need to implement Control and Data plane bundling policies.

1.1.1. Control Plane Policies

The control policies are those mechanisms in the system aimed to maintain the network access pool ready to use. Some example of software already doing so is the Network Manager, present in most of platforms today, which is responsible for the network connections of a system. The way the Network Manager usually operates is by assigning increasing priorities to different types of interfaces (mobile, wireless LAN and wired access). Whenever Wi-Fi becomes available, it disconnects from mobile network and connects to Wi-Fi and if an Ethernet connection is available it uses this one first. Some Network Managers are actually able to maintain different network connections at the same time, but they will still not be able to properly configure the required routing tables for a multipath connection to take place. This is because these managers are aimed for single-path transmission, where only one path is going to be used for transmission in a given time. One integral solution for multipath communications would require the Network Manager to bear a hand in the multipath managing.

However, not even the best possible network manager we could think of solves the issue of traffic bundling by itself. To do this, we need to go into the data plane to develop policies which allow us to split traffic along those links which the control plane have provided for us. Although, as we will see later on, control plane policies and data plane policies performance are correlated, deeper insight into date plane are beyond this project and we are not going further into them (although we will refer to them a few times along the document). It is, however, within the scope of this work to go deeper into the data plane policies.

1.1.2. Data Plane Policies

The classic Internet protocol suite (also called the TCP/IP model) was conceived as a network architecture to provide reliable end-to-end connectivity over IP networks [4] and since the very first steps, the suite has become de fact standard in the present Internet. Designed to single-path transmission, the Transmission Control Protocol(TCP), which the architecture bases on, is the dominant protocol inside the Internet. Due to this network architecture was never meant to host multi-path, most of Layer 4 protocols in the Internet sticks to the single-path TCP/IP model rules: One sending interface, one receiving interface, one packet flow in between them. This is a limitation when what we want to do is precisely having several sending interfaces, several receiving interfaces and consequently several packet streams flowing in the middle.

The development of Multi-path TCP (MPTCP) was a reaction to this limitation of the TCP protocol. It enables traffic bundling by defining stream level rules so the traffic can be split among all available links or paths in one end and reconstructed at the other end to be used. To do so, multipath-aware entities at both sides handle multiple connection establishments, queues and sending policies. In summary, MPTCP extends classic TCP features to make it possible to have multipath transmission on the actual TCP/IP architecture.

Of all the advantages that traffic bundling can bring to the table, we have identified three which can be especially useful in the context of a diverse wireless access offer as described above. These are:

Bandwidth Aggregation The combination of the individual capacity available in different paths is the first (and obvious) advantage a multipath connection can accomplish. Successful bandwidth aggregation capabilities would allow us to combine our 3G and Wi-Fi connections into a more powerful and faster link.

Inter-technology traffic handover A multipath approach can also bring benefits to roaming along different networks, specially between heterogeneous networks. By using multipath it should be possible to smoothly switch traffic, for instance, from mobile to Wi-Fi, or form 2.4 GHz Wi-Fi to 5 GHz Wi-Fi with minor impact in performance.

Cellular traffic offloading This last advantage is somewhere in between the other two. Due to MPTCP capacity to adapt transmission load according to different network parameters, we believe it has some potentials as well on traffic offloading from Mobile Networks to

WLAN.

1.2. Goals and project phases

In this work we explore benefits and limitations of MPTCP in wireless environments. This is why the core objective of the forthcoming research is developing different test-beds to realize MPTCP performance tests in terms of bandwidth aggregation, inter-technology traffic handover and cellular traffic offloading. This will give us the necessary outlook of the protocol's potentials in these situations. For each of these hypothetical advantages of MPTCP we implement a test-bed in a laboratory network which will allow us to test it in reality and analyse the protocol's behaviour. Therefore, for each of the possible MPTCP benefits, we do:

1. Define a test scenario which allows to observe this behaviour.
2. Implement the defined scenario in a real test-bed.
3. Take performance measurements in the test-bed.

As for the execution of the project, the work was divided into different phases. These covered a contact stage, the test-bed definition, the execution of the experiments and the evaluation of the obtained results.

Hands-on stage Research about multi-path transmission, with a focus on MPTCP protocol, as well as about relevant wireless access technologies (namely WLAN and mobile) and the necessary tools to test MPTCP in wireless.

1. Technical analysis of MPTCP operation: How the protocol works and how it is properly installed and configured.

2. Context of Wi-Fi and mobile technologies and how MPTCP enhance their performance.

3. Software and hardware tools which can be used to test bandwidth aggregation, handover time and traffic offloading in multipath.

Test-bed implementation Definition of those scenarios where multipath gain is expected and implementation of the corresponding test-beds which allow us to observe this gain experimentally.

4. Definition of the scenario.

5. Implementation on a test-bed for measurements.

6. Running the corresponding performance tests and collecting the measurements.

Performance evaluation Evaluation of the protocol's performance on the different scenarios and documentation.

7. Process the measurements for visualization and analysis.

8. Documentation of the obtained results. Conclusions.

1.3. Document Structure

In Chapter 2 of this document the MPTCP protocol is presented, together with a small review of WLAN, mobile and other technologies which are going to be used in the test-bed implementation phase. Chapter 3 covers the definition of several scenarios which are relevant to observe MPTCP behaviour, as well as a translation of these scenarios into physical test-beds for actual experimentation. Chapter 4 presents the obtained results on the test-bed as well as an evaluation of how the protocol has performed. Finally, the last chapter contains our final conclusions regarding the protocol potentials, applications, limitations and future improvements.

Chapter 2

State-of-the-Art

In this chapter we give a perspective of the traffic bundling technique, positioning MPTCP on the spectre of possibilities for its implementation. A detailed description of the MPTCP protocol is thereafter provided basing on both research of previous work and practical experience. We also explain how cooperation of wireless access technologies with multipath can bring benefits in some cases and present the tools which we will be using to actually test MPTCP performance in next chapters. A summary of the most relevant wireless access technologies (roughly WLAN and cellular) as used in the test-beds is also included here.

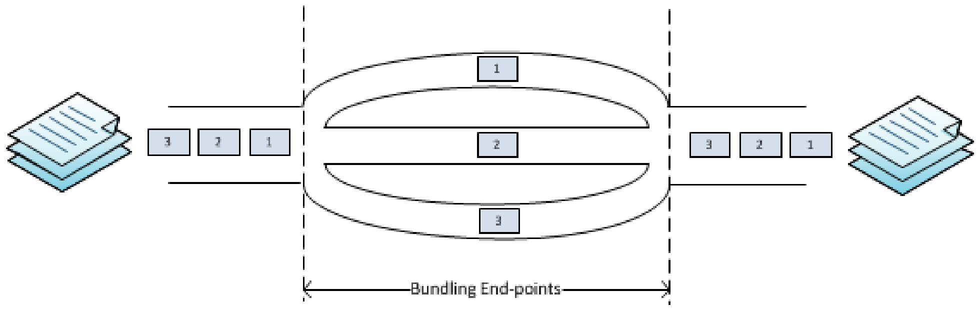


Figure 2.1: Traffic bundling technique.

2.1. Traffic Bundling and Multipath

The traffic bundling is a technique in packet switching networks that uses scheduling to split the traffic belonging to one flow of packets into different sub-flows. Typically, these sub-flows will thereafter travel through different physical paths in the network, although this is not necessarily always de case. The concept is closely related to that of bandwidth or capacity aggregation, which looks to combine different available physical links of a certain capacity into one whose capacity is the summation of all (the idea is also often known as port trunking, link bundling, link aggregation or NIC bonding/teaming). In general, in this document we refer to bandwidth or capacity aggregation as a consequence of using traffic bundling in some scenarios. When traffic belonging to the same logical stream flows through different physical paths due to bundling we call that a multipath stream. Depending on the layer of the protocol stack where the end-points are implemented, different technologies exist providing traffic bundling, each of them with their pros and cons.

In the Layer 2 we have the Link Aggregation Control Protocol (LACP), which was first proposed in 2005 by IEEE on its standard 802.3ad [5]. It defines a way to bond multiple physical links into a single one, which can be operated from the upper layers. It is meant for network devices such as hubs and switches and it only supports aggregation of point to point Ethernet links; this means the protocol only supports aggregation of links of the same technology (and even there with limitations). There are some proprietary solutions providing link layer aggregation, such as Cisco's EtherChannel, Juniper's Aggregated Ethernet and others[6] [7] [8] [9]. Similarly, in the context of DSL lines, the Point-to-Point (PPP) protocol also provides an extension to allow bandwidth aggregation [10] by means of the PPP Multilink Protocol (MP), which is defined in the RFC1717[11], but again with the limitation of only supporting DSL to

DSL bonding.

If we step up for Layer 3 bundling technologies a few other options are available. The Linux bonding diver defines a virtual interface for traffic bundling. This interface internally re-schedules the traffic along different interfaces to achieve bundling. Other solutions involve the usage of proxies in the middle to split and bond streams [12] or make usage of both IP tunnelling and more complex schemes [13]. All of them rely one way or another in architectures which need changes in the network, auxiliary components or/and software packages installed in both the server and user side. Implementing the bundling end-points in the network layer makes the solution independent from underlying link technologies and therefore enables aggregation of heterogeneous links.

Transport layer bundling has two major candidates to take on these duties in the following years: The Stream Control Transmission Protocol (SCTP), as it is defined in the RFC4960 [14], is a transport-layer protocol which was first defined in 2000. It was designed for PSTN signalling, but capable of reliable connection oriented transmission. It is an answer to some limitations of TCP protocol such as strict ordering in the segments delivery, its stream-oriented nature sometimes forcing applications on the top to use `TCP_PUSH` facility to make sure messages are delivered in time and some security issues. The protocol provides support for multi-path by setting the rules for multiple to multiple IP communication. The other choice here is MPTCP.

2.2. Multipath TCP

As it was explained earlier, the first motivation to develop such a protocol is the original TCP limitation in handling multiple data streams flowing through different paths. In this kind of environments the disadvantages of TCP are clear:

- It only allows one connection establishment per IP tuple. This is, one connection per [IP—port] pair. Since packets in data networks are forwarded basing on its destination address, it is not possible to get those packets to flow through different paths with TCP. Although the path packets are flowing through in a TCP connection may change in time (due to link failures or routing readjustment) it is not possible with TCP to hold simultaneous streams going through different paths, i. e., having a multipath flow, and it would have to rely on a L2-L3 bundling technique to do so.

- TCP does not provide mechanisms so end entities can communicate. This lack of a signalling scheme prohibits single TCP from notifying an eventual migration form one IP to another or new interfaces or IPs getting available for its peer to interact with.

These are two major drawbacks, leading to TCP not being an optimal choice in many scenarios where multiple communication paths between hosts are possible, and thus it is not adequate in those cases. We explain how MPTCP solves this problem, achieving a faster and more reliable transmission scheme.

2.2.1. Model

MPTCP is a Layer 4 protocol, which means it infiltrates in between the application and the IP layer. More precisely, MPTCP can be seen as an extra layer inserted right above the TCP layer and under the application layer. By standing in the middle of them, an MPTCP connection can make use of multiple TCP connections, which it handles internally. Although in most of cases only one connection per interfaces pair is used, MPTCP allows us to use as many sub-connections as desired in multiple connection schemes. Despite it is implemented on the top of TCP, an MPTCP connection requires an MPTCP-enabled entity running in both ends. This is mandatory because it is this level which provides the intelligence to reorder the received segments across all sub-flows.

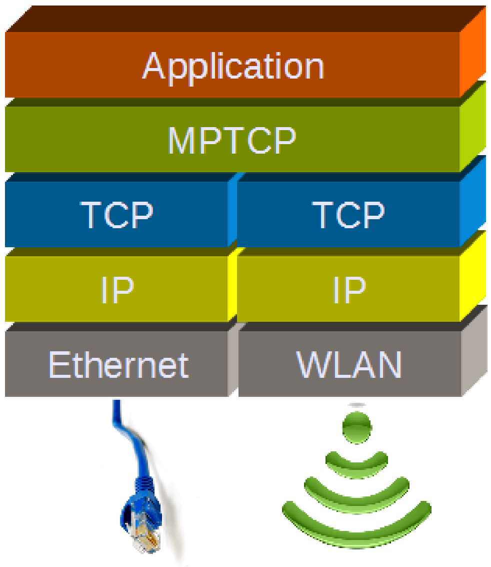


Figure 2.2: The MPTCP model.

Again, this approach comes at the disadvantage of requiring changes in the end devices (both client and server), since new software and configuration is needed. It is because of this problem that an architecture with a middle proxy can be proposed (Figure 2.3). In this scheme the proxy is assuming those changes that otherwise the user's end device would have to assume, relying in this proxy for the multipath managing and preventing from any change or update to be needed in the user side. This is how the MPTCP-in-the-middle approach can be used as a network side solution to aggregate paths. If it is only one side who is not supporting multipath, a semi MPTCP-in-the-middle approach can be used instead to deal with multipath incompatibility in one side only [15].

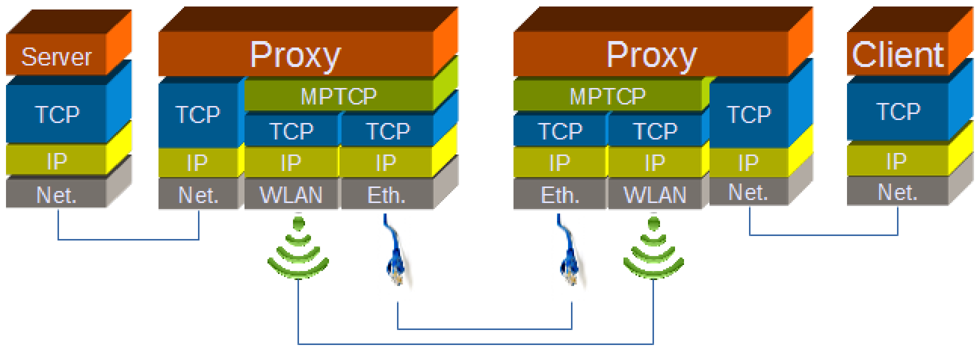


Figure 2.3: MPTCP-in-the-middle architecture.

For the purpose of this work, a simple MPTCP architecture with MPTCP entities running at both sides will be enough.

Rules

MPTCP was designed to fulfil three basic rules, which are intended to ensure proper behaviour from different points of view. The protocol needs to stick to this rules in order to be consider a viable alternative to TCP:

Improve Throughput A multipath connection must achieve throughput which is at least as high as the best single path connection would achieve. This is an obvious rule, because if it was not the case it would not worth it to upgrade from TCP.

Do not harm A multipath flow should not use more capacity from any of the links his subflows are going through than if it were a single TCP connection using only one of them. This is, MPTCP should gain performance form the use of multipath and not from stealing capacity from TCP.

Balance congestion Distributing and adjusting the load among the different paths is also an important goal. The protocol must adapt the rate on each flow to the available capacity, thus those less congested paths will be used more than more congested paths.

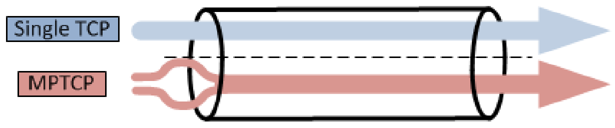


Figure 2.4: A multipath connection should not take up capacity form a coexisting single path connection.

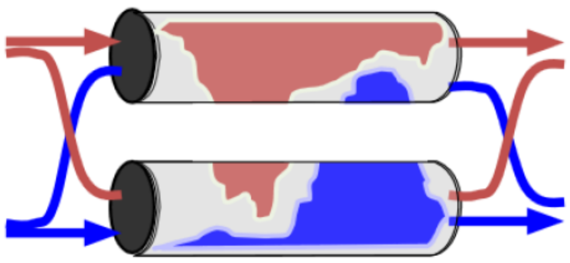


Figure 2.5: The MPTCP congestion control tries to move away traffic from congested to uncongested links [16].

There is another recommendation when it comes to actually implement the protocol, which is the compromise of doing it in such way that it is transparent to the application layer. This will make it compatible with already available applications and will not force them to rewrite their code if they want to benefit from the use of multipath. The way it has been suggested to achieve this is by overwriting those functions that are being used in the application layer, namely the TCP Application Program Interface (API). An MPTCP implementation overwrites the TCP API, internally adding the multipath capabilities and keeping its usage exactly the same as it was before. There exist, however, a complementary API providing further capabilities in case programmers are interested on developing MPTCP aware applications.

2.2.2. Operation

MPTCP is an extension of TCP to support multipath transference. To extend the functions of TCP, it makes use of the options field in the header to provide some signalling that can be used to do so. When an MPTCP-capable device initiates a connection (because an application is requesting so), it announces in the `SYN` segment multipath support by activating the flag `MP_CAPABLE` in the header extension in the options field. If the other end supports multipath as well it will acknowledge activating the same flag `MP_CAPABLE` in the replied `SYN\ACK`. Otherwise it will reply with the flag off in the `SYN\ACK`, so the initiator will understand it has to fall back the

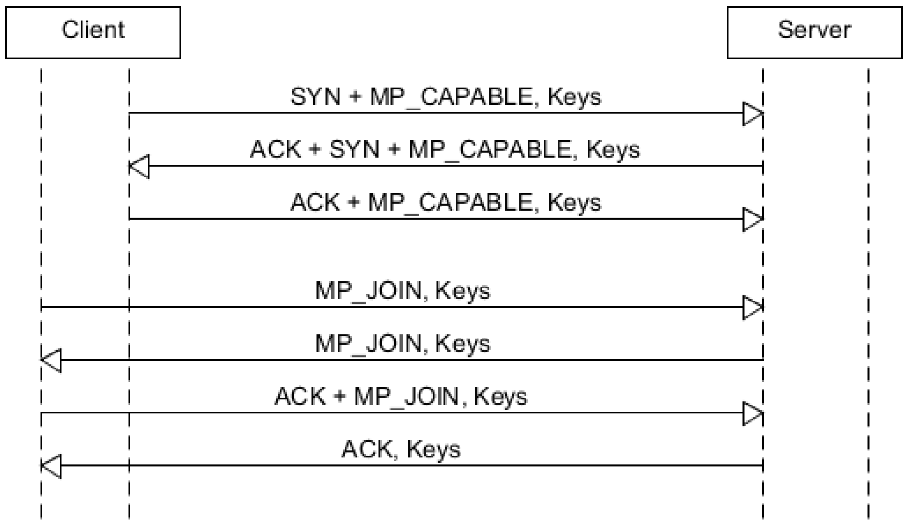


Figure 2.6: MPTCP establishing and joining mechanisms.

connection to regular TCP. This mechanism makes MPTCP backwards compatible and avoids problems in communications between hosts supporting different versions of the protocol.

Besides the `MP_CAPABLE` flag, there is another field of interest that is being added in the handshake: the connection keys. We will not go into detail in this document regarding the security mechanisms of MPTCP, which are the goal of this keys. Enough saying that they are one key per peer, and they will be used later when it comes to add up new subflows in order to guarantee security. We therefore not include those keys in the following figures showing the operation of an MPTCP handshake and the JOIN mechanism. Further details regarding security mechanism in MPTCP can be obtained in the RFC6824 [17].

The JOIN process, which takes place every time a new sub-flow joins the transmission, follows a different mechanism. Depending on the path manger policy we have selected [2.2.4], this process can vary, but typically it will be the client triggering a JOIN from every other available IP immediately after the main sub-flow has successfully established connection or every time a new IP is announced to him (we explain later this mode). In any case, the mechanism will follow the next sequence: The client sends a `SYN` with the `MP_JOIN` flag activated along with a multipath connection identifier. At this point the server will acknowledge the join and an authentication process will start. In this process the keys already exchanged in the handshake stage are being used. After successfully authenticating the join request, a new path will be

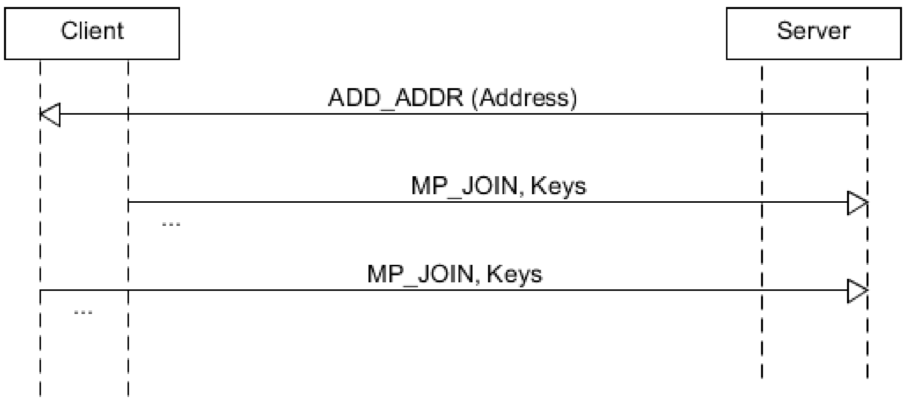


Figure 2.7: MPTCP announcing mechanism.

available to send data.

MPTCP allows that both sides can trigger a JOIN at any time during a multipath connection. It is, however, recommended that only the initiator (typically the client) will do so. This is because of simplicity. By activating the `ADDR_IN` flag one end (typically the sever) can signal to the other end that a new IP address has become available. The client can now choose whether to initiate or not a JOIN to that IP address and which IP addresses initiate from. There exists as well MPTCP flags to mark sub-flows as only backup sub-flows so they will not be used unless some main/more priority flows fail (see Section 2.2.4).

In Figure 2.7 the server announces a new IP to the client, who then tries to initiate the corresponding new sub-flows. In this example the client is triggering one sub-flow per interface. If we assume there were two other sub-flows already going on between client and server, we have now four sub-flows in total. This establishment policy where sub-flows are created from every IP in the client to every IP in the sever is actually one of the possible operation modes of the path manager, which is called `'fullmesh'` mode. Nevertheless, other operation modes are available. The whole signalling set of the protocol can be consulted in the RFC [17].

2.2.3. Congestion

The congestion control algorithms in TCP are based on a congestion window and rules about how to increase or decrease that window. TCP look up for congestion in the flow by detecting different congestion events (what is considered to be a congestion event is different from one congestion algorithm to another; for instance in TCP Reno only packet loss will trigger the congestion control procedures, whilst in TCP Vegas are both packet loss and RTT variations that do so). Meanwhile congestion is not detected, the congestion window is increased up to its maximum, which is the announced transmission window[1]. The moment congestion appears in the network and is detected, the size of congestion window is readjusted to adapt to the network status. In case TCP congestion control mechanisms are not clear at this point we recommend reading RFC2581 where this mechanisms are described [18]. We have already explained how MPTCP signals sub-flow adding and removing as well as how it solves backward compatibility. Rules one to three, regarding throughput, fairness and load balancing, require some different mechanism to be solved, which will involve coupling the congestion windows on individual sub-flows.

MPTCP tries to couple congestion windows of the individual sub-flow so congestion status in a path can affect the others. This way the multipath streaming can adapt to the overall congestion status. The complete coupled congestion control algorithm in MPTCP is covered in the RFC6356 [19]. The algorithm does not, in principle, introduce any change on how congestion is detected or how the decreasing rule applies and just applies the same rules as single TCP CCA to do so. It does, however, redesign the increasing rule[2] in its congestion avoidance stage. MPTCP holds one congestion window per sub-flow and relate them through a coupled increasing rule. For each ACK received on sub-flow i, it increases the size of the congestion window for that sub-flow (cwndi) by:

$$min(\frac{alpha}{cwnd_{total}}, \frac{1}{cwnd_i}) \tag{2.1}$$

MSSs[3], with i indicating sub-flow index. The cwndtotal variable roughly represents the aggregated congestion window along al sub-flows, although accurate description of how this parameter must be computed should be better consulted in the official algorithm definition in the RFC 6356 [19]. Let us first ignore the effect of alpha and explain how this scheme is achieving rule two.

The increasing rule for single TCP increments the size of the congestion window by one (MSS) every time acknowledge is received. This is, it is increased by a factor of 1/cwnd. The

[1]The transmission window would be the available buffer memory announced by the receiver.

[2]More precisely, the congestion avoidance stage rule.

[3]Maximum Segment Size of TCP

first argument in the min expression will be usually more restrictive than the second one, but in case it is not, equation 2.1 makes sure the increment in the congestion window is never more aggressive than that of a single TCP in the same situation. This will prevent MPTCP from unduly damaging other coexisting flows.

The objective of alpha is achieving rule one and three by setting an increasing parameter such that the aggregate throughput of the multipath flow is at least as high as the throughput a TCP flow would get if it ran on the best path and at the same time push away traffic from congested to uncongested links. The total throughput of a multipath flow depends on different factors like the loss rate, capacity or RTT on its path. That is why the following rule suggested to be used to compute the value of alpha on the fly [19].

$$alpha = cwnd_{total} \frac{max_i(\frac{cwnd_i * mss_i^2}{RTT_i^2})}{(\sum_i \frac{cwnd_i * mss_i}{RSS_i})^2} \tag{2.2}$$

This algorithm, called the Linked Increases Algorithm (LIA), is the default algorithm in the UCLouvain linux implementation of MPTCP and the reference congestion control algorithm during this work, but since version v0.89 the implementation offers as well other CCAs choices, such as:

- The Opportunistic Linked-Increases Algorithm or OLIA [20]
- The Balanced Linked Adaptation Congestion Control Algorithm or BALIA [21]
- The Delay-based Congestion Control Algorithm or WVEGAS [22]

2.2.4. Configuration

Path Manager

The path manager decides how the sub-flows are established across the available IP addresses on client and server. If a path manager were not present, the host would not be able trigger the creation of new sub-flows, nor advertising alternative IP-addresses. You have the choice between:

- 'default': This path-manager actually does not do anything. The host will not announce different IP-addresses nor initiate the creation of new sub-flows. However, it will accept the passive creation of new sub-flows.

- ’fullmesh’: It will create a full-mesh of sub-flows among all available IP addresses.
- ’ndiffports’: This one will create a certain number sub-flows across the same pair of IP-addresses, modifying the source-port.
- ’binder’: The path-manager using Loose Source Routing from the paper [23].

Scheduler

The scheduler decides how to allocate/send the packets over the available sub-flows. You have the choice between:

- ’default’ The default scheduler will first send data on sub-flows with the lowest RTT until their congestion-window is full. Then, it will start transmitting on the sub-flows with the next higher RTT.
- ’roundrobin’: This scheduler will transmit traffic in a round-robin fashion.

One additional option available for an MPTCP connection which also related to scheduling, is the Backup Mode. It is possible to set one interface (not necessarily the same as setting one sub-flow) as a backup interface, so if the connection fails on the main interface/sub-flows, the transmission is immediately relayed on the backup interface. Sub-flows in backup mode are not being used unless all the other sub-flows fail.

2.2.5. Present status

Some theoretical background about the feasibility of multipath streaming was released in 2005 [24] and 2006 [25] [26]. The first publication containing architecture and primal implementation for of MPTPC was released in 2011 by Sebastien Barre, as a PhD thesis [27] in the Univeristé Catolique de Louvain (UCLouvain). He had already been working with the concept of multipath since 2008, when he released his research on the Shim6 protocol [28]. Another big contributor to MPTCP development is the University College of London, with several publications in the field. The protocol development goes on presently in the Internet Engineering Task Force’s (IETF) Multipath TCP working group. The have already released five RFC describing different aspects of the protocol among other interesting publications [29]. The latest definition of the protocol was released in the RFC6824 in January 2013. In July 2013 the MPTCP work group published a survey showing up to five different implementations of the protocol so far. Today these are:

1. The Linux implementation from UCLouvain (the reference implementation)[30].
2. The FreeBSD implementation from [31].
3. Networks BIG-IP LTM [32].
4. Citrix Netscaler [33]
5. A privative implementation from Apple Inc., which was reported to be using MPTCP on its virtual assistant Siri [34].

In this thesis we will focus on the main implementation from UCLouvain.

So far two promising applications have been investigated for MPTCP. In this paper we can see how MPTCP can be used to switch traffic between 3G and Wi-Fi in a phone, which can lead to gain in performance, battery use and data consumption [35]. There have also been works about how MPTCP would help to load balancing in data centres exchange by making use of multiple interfaces usually available in this kind of deployments since MPTCP perfectly meets the bandwidth and security and redundancy requirements for data centres [36]. This work intends to be a new contribution to the MPTCP experimentation.

Deployment

For these work we are going to focus on the MPTCP implementation from UCLouvain (mentioned before). Guides about how to install the latest version on different Linux platforms can be found in their reference webpage [37]. In Debian and Ubuntu 64 bits, in order to have MPTCP the installation of a kernel patch is needed. Detailed explanation of how to install such patch can be found as well in the webpage. After installation, different MPTCP parameters such as the path manager, scheduler or the congestion control algorithm can be tuned with a set of commands.

- The Path Manager: To select one of the compiled path-managers just set the

  'net.mptcp.mptcp_path_manager'

  system variable. Choices are 'default', 'full_mesh', 'ndiffports' and 'binder'. Under 'ndiffports' mode, to control the number of subflows (X), you can set the

’/sys/module/mptcp_ndiffports/parameters/num_subflows’

variable to the desired value.

- The Scheduler: You can select one of the compiled schedulers through the

  ’net.mptcp.mptcp_scheduler’

  system variable. Options are ’default’ and ’roundrobin’

- The Congestion Control Algorithm: Select between different congestion control algorithms with

  ’net.ipv4.tcp_congestion_control’

  variable set to either ’lia’, ’olia’, ’balia’ or ’wvegas’. These are the four coupled congestion control algorithms available in the latest v0.90 release, every other CCAs will work in uncoupled mode.

It is also possible that you set one of the available interfaces as a backup interface, so it is used only when sub-flows on other interfaces fail. For that it is necessary to install the multipath extension to ip link tool in Linux by following the directions in the webpage [37]. Once it has been successfully installed, you can set it up like with ’ip link set dev ¡if¿ multipath backup’. It also possible to remove one interface for multipath consideration by doing ’ip link set dev eth0 multipath off’

Since the aim of this thesis is showing how MPTCP performs over wireless access links, we also include a review of the two most remarkable technologies in wireless access, such Wi-Fi and 3G.

2.3. Wireless Networks

Since in this project we are going to develop different test-beds for performance tests of MPTCP over wireless networks we give a review here of the two main technologies we are going to be using in the set-ups, namely WLAN and mobile.

2.3.1. WLAN Networks

Although the term Wi-Fi is often being used to name any wireless access technology inside the local range (WLAN), Wi-Fi are, technically speaking, those technologies which were tested and approved by the commercial organization Wi-Fi Alliance. This is the organization in charge of coordinating the compatibility and quality of these technologies worldwide and what most of companies providing wireless access systems are involved with it. Wi-Fi is the commercial name receiving a subset of specifications of IEEE 802.11 approved by the Wi-Fi Alliance [38].

IEEE 802.11 is a set of specifications for the Physical (PHY) and Media Access Control (MAC) layer of Wireless Local Area Networks or WLAN. It describes technology standards for the radio access in the frequency bands of 2.4, 3.6, 5 and more recently 60 GHz. Below we name some of the most relevant. All of them are approved as Wi-Fi and commercialized all over the world in such way.

- 802.11a: It is a modification of 802.11 standard released in 1999 which integrates the use of ODFM in the physical layer. It was originally design to occupy the 5-6 GHz band, which was the unlicensed in the United States. Thenceforth the standard became very popular and suffered several updates. The latest version of the standard is using the 4.9-5.9 GHz frequency band and delivers a throughput of 54 Mbps on the physical level.

- 802.11b: This is another modification of the original standard IEEE 802.11 which increases the bitrate up to 11 Mbps. It is allocated in the 2.4 GHz band.

- 802.11g: 2003 standard adding OFDM capabilities to 802.11b physical layer and extending the speed to 54 Mbps. It occupies the 2.4 GHz band.

- 802.11n: This recently proposed specification incorporates MIMO (Multiple Input Multiple Output) to the standard. It works on both the 2.4 GHz and 5 GHz frequency bands and it can theoretically deliver a throughput of 600 Mbps. In practice typical values are from 50 Mbps to 100 Mbps.

- 802.11ac: It is the latest review of the standard and its commercial availability is still

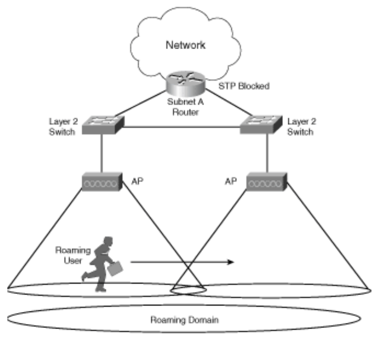


Figure 2.8: L2 Wi-Fi Handover

reduced. It increases the channel bandwidth from 40 MHz to 160 MHz and the modulation order from 64-QAM to 256-QAM, thus multiplying performance and making it possible to obtain bit rates up to 1300 Mbps in both 2.4 GHz and 5 GHz.

A wireless link in any of the previously described specifications requires an association process between base stations and, most of the times, authentication as well since 802.11 requires a mobile device to establish its identity with an AP or a broadband wireless router. Once authentication is complete, mobile devices can associate (register) with an AP/router to gain full access to the network. Association allows the AP/router to record each mobile device so that frames may be properly delivered. A station can only associate with one AP/router at a time. [39]

Association Process

After the mobile device authenticates to an AP/router, it sends an Association Request which the AP/router processes. AP/router vendors may have different implementations for deciding whether or not a client request should be allowed. When an AP/router grants association, it responds with a status code of 0 (successful) and the Association ID (AID). Failed Association Requests include only a status code and the procedure ends [39].

Association occurs at the MAC layer and it hoards the link until it is finished. During roaming (Fig 2.8), the L2 handover time in WLAN may vary a lot depending on the vendor and product, but it is usually over 200ms [40]. This small gap will affect ongoing TCP connections over the links undertaking handover, because they will perceive this situation as a sign of congestion an therefore unnecessarily penalize these flows. This is a problem for environments

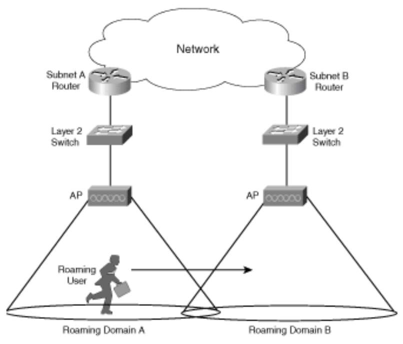


Figure 2.9: L3 Wi-Fi Handover

where applications which are have critical latency requirements such as VoIP or video streaming are relaying on a Wi-Fi infrastructure, because in many of cases a Wi-Fi client will fail on handover connection from one AP to another in a reasonable time, taking up to several seconds to do so [41]. Some research [42] talks about 250ms to 1s as observed practical layer 2 handover delay during Wi-Fi roaming, however application layer delay can be much higher when TCP is involved.[4]. This can be explained by a couple of reasons:

- Congestion control in TCP treats packet loss as a sign of network congestion, which in a situation of poor link quality where a lot of packets are being lost will cause the throughput to drop rapidly before the hand-off process had even started.

- TCP is affected by the internal timers that rule probing for changes in the congestion of the flow[5]. These are not aware of layer 2 events such as a new association event and therefore delay transmission continuation.

If the situation is such that a L3 handover is needed too (i. e. not only the client is re-associating to a new physical AP but IP address need to change as well) ongoing TCP connections will be completely wasted 2.9. As we will see later in Section 3.1, multi-path can be applied to this scenario to improve perspectives in WLAN handover.

[4]Practical experience showed TCP delay in Wi-Fi roaming can be up to 7-10 s [41]

[5]TCP stops transmitting if network is too congested and checks for changes in the state of the network congestion every certain time. Even if the network becomes uncongested back up transmission will not restart until TCP realizes so.

2.3.2. Mobile Networks

Unlike WLAN technologies, which almost completely consist of different versions of the 802.11 standard forming Wi-Fi, when we talk about Mobile Networks we usually refer to a wide range of coexisting Radio Access Technologies which are used today to reach the telephone providers network from our mobile phones. Telecommunications providers usually label their services as 2G, 3G or 4G... which refer to the generation of access technologies where they are included.

- 2G groups the second generation of mobile communication technologies, of which GSM is the most relevant [43].

- 3G groups the third generation of mobile communication technologies, which are based on the IMT-2000 standard [44]. A variety of systems complying with the IMT-2000 are coexisting today (EDGE, CDMA2000, UMTS, LTE...)

- 4G groups the fourth generation of mobile technologies which fulfil the ITU standard IMT-Advanced. Although many providers have usually labelled LTE as 4G, this technology does not completely match the requirements to be consider such. LTE-Advanced systems however do so, although few carriers have managed to deploy this ”true 4G” technology as of today.

All these systems have defined mechanisms to perform tolerable handover between stations but the situation where handover is needed from a mobile network to WLAN is again not contemplated. This is called inter-technology handover or vertical handover and is another field for improvement with multipath, since a vertical handover normally requires all the existing TCP connections taking place over one access technology to die in the handover process. This causes increased delay in and also requires the application to deal with new connection establishment over the new interface, making up for possible packet losses in the way and securely opening a new session to continue communication. All this issues can be improved by using a multipath communication underneath.

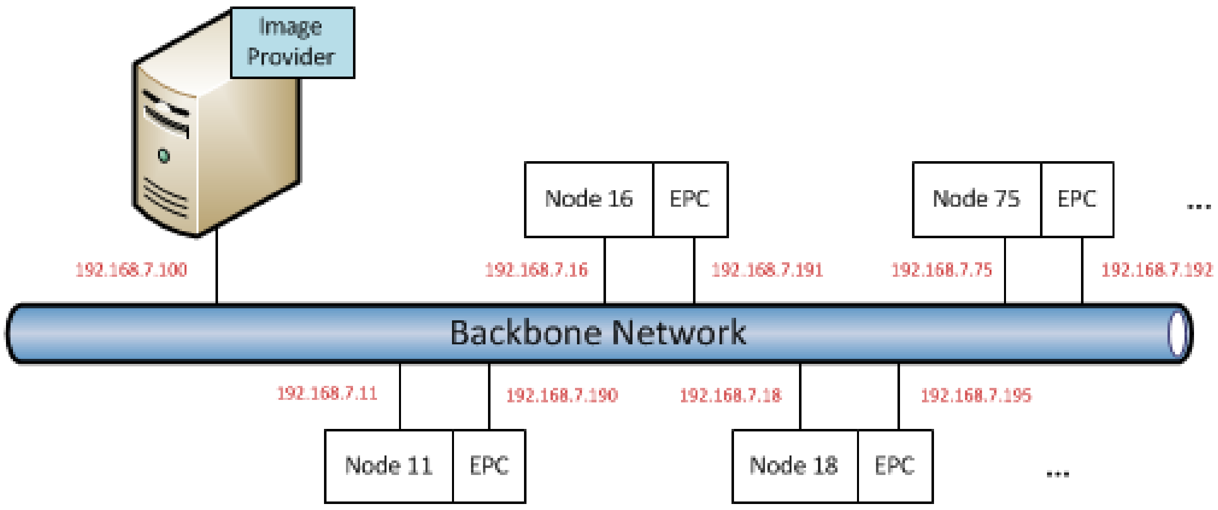


Figure 2.10: The MRAT network.

2.4. MRAT

The MRAT network is the name of the general purpose network which the Seamless Network Control team has deployed in T-Labs Berlin to nest a wide range of networks tests. It consists of a set of devices nodes attached all over a few floors in the building which can be remotely configured and turn on and off using an internal web application. Due to MRAT being an actual fully-configurable remote-controlled network of nodes it becomes very simple to bring up and down a Wi-Fi network, simulate link failures, monitor status, track down and fix configuration problems. All nodes in the testbed are identified by a unique local IP and can be rebooted remotely via an Energy Power Controller (EPC) which can be accessed in a web service on its local IP. On startup, the node loads a system image and configuration scripts residing on a central entity, which offers web application as well. By pre-setting these system image and scripts in the image provider it is possible to get the nodes to boot to the desired state. Node's configuration is realized by selecting three things:

1. **System Image:** Different choices of bootable system images are available in the Image Provider entity, such as Debian or OpenWRT and a web interface can be used to set the default image each node will boot on start-up. The Debian image was always enough for our purposes along the experiments in this work, and therefore this must me assumed to be our choice on the set-ups, although other distributions were also possible.

2. **common.sh:** This is the common configuration script which all nodes will load and

execute on booting so general configuration commands must be placed in this script. It can be edited as will in the Image Provider web interface. We have used different common.sh scripts along the project to realize the test-beds.

3. **[IP].sh:** This script contains the specific configuration commands for one node. It is identified as [IP].sh, where [IP] is the last number of the node's IP address in the dot notation. This script can be different from one node to another in the same set-up.

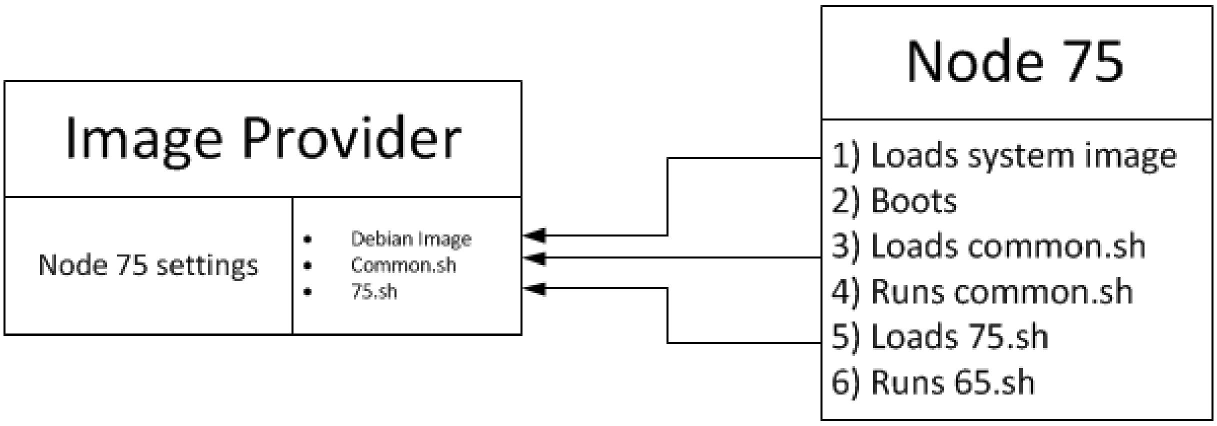


Figure 2.11: Node 75 on start-up.

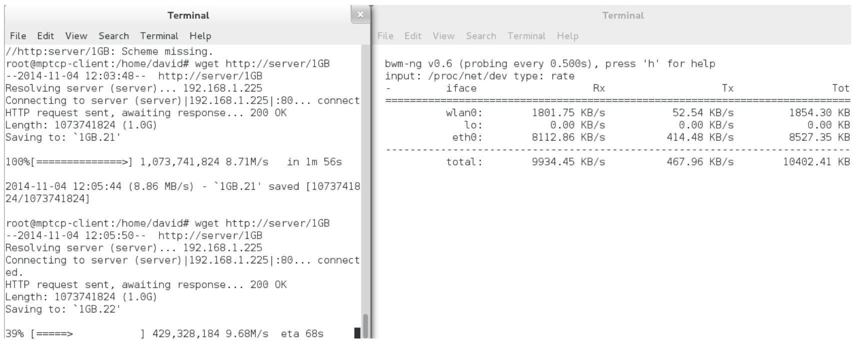
The boot up process is described on figure 2.11 for node 75 as an example. Along with different set-ups implemented in the MRAT, a set of software tools where used to monitor the network and run throughput tests. These are covered in the next section.

2.5. Monitoring tools

2.5.1. Data collection and analysis tools

- **bwm-ng** It monitors the instantaneous physical throughput on every network interface in the system, although it does not provide functionality to record the values by itself. Mainly used during evaluation to have a quick view of which interfaces are being used for transmission and with what intensity. Available for at least for Debian and Ubuntu.

- **speedometer** Similarly to bwm-ng, this tool monitors the throughput at the physical layer. It provides a nice visualization framework to observe throughput in real time. Install it by typing



```
//http:server/1GB: Scheme missing.
root@mptcp-client:/home/david# wget http://server/1GB
--2014-11-04 12:03:48--  http://server/1GB
Resolving server (server)... 192.168.1.225
Connecting to server (server)|192.168.1.225|:80... connect
HTTP request sent, awaiting response... 200 OK
Length: 1073741824 (1.0G)
Saving to: `1GB.21'

100%[==============>] 1,073,741,824 8.71M/s   in 1m 56s

2014-11-04 12:05:44 (8.86 MB/s) - `1GB.21' saved [10737418
24/1073741824]

root@mptcp-client:/home/david# wget http://server/1GB
--2014-11-04 12:05:50--  http://server/1GB
Resolving server (server)... 192.168.1.225
Connecting to server (server)|192.168.1.225|:80... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 1073741824 (1.0G)
Saving to: `1GB.22'

39% [=====>           ] 429,328,184 9.68M/s  eta 68s
```

```
bwm-ng v0.6 (probing every 0.500s), press 'h' for help
input: /proc/net/dev type: rate
-         iface                   Rx                   Tx                Tot
============================================================================
          wlan0:         1801.75 KB/s          52.54 KB/s         1854.30 KB
             lo:            0.00 KB/s           0.00 KB/s            0.00 KB
           eth0:         8112.86 KB/s         414.48 KB/s         8527.35 KB
----------------------------------------------------------------------------
          total:         9934.45 KB/s         467.96 KB/s        10402.41 KB
```

Figure 2.12: The bwm-ng tool can monitor the physical layer throughput of each interface.

```
sudo apt-get install speedometer
```

in both Debian and Ubuntu. You can configure all kinds of different formatting and displaying options by using parameters.

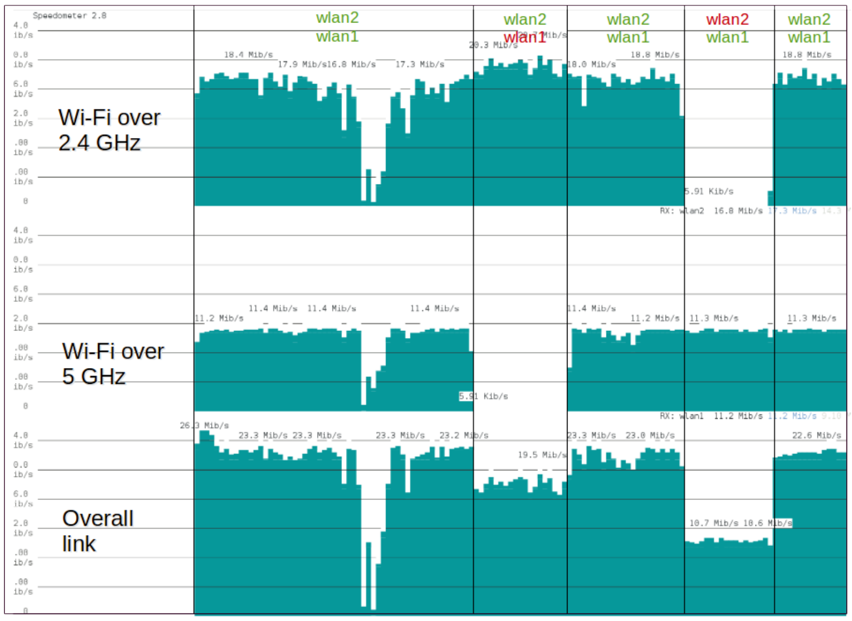
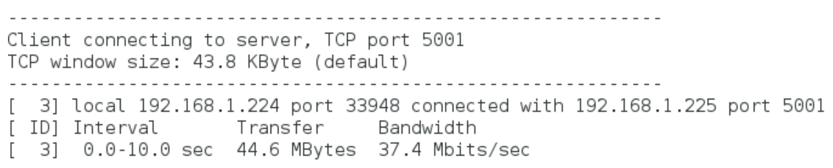


Figure 2.13: Speedometer provides physical layer throughput monitoring with a better visualization.

- **iperf** Iperf is a bandwidth monitoring tool which provides measurements for throughput tests between hosts. It comes with a variety of options to do bandwidth tests over TCP, UDP and SCTP. It is also possible configuring iperf to produce the output in a comma-separated fashion, directly in a file of our wish. This is why iperf is the preferred tool to realize the tests in most of the occasions in this work. To perform throughput tests with

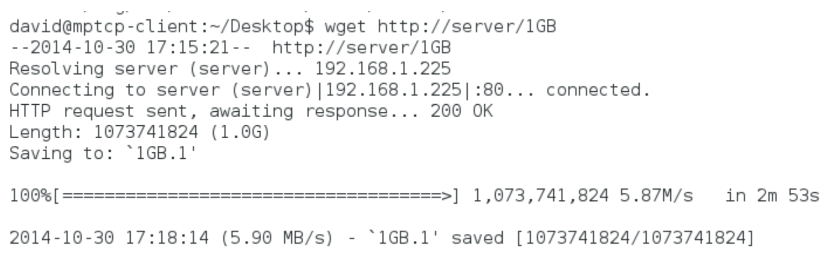


```
------------------------------------------------------------
Client connecting to server, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.224 port 33948 connected with 192.168.1.225 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  44.6 MBytes  37.4 Mbits/sec
```

Figure 2.14: Output of an iperf test.

this tool, it is necessary to have iperf entities at both sides, i. e. we need to start an iperf server in the peer prior to beginning the test. After this, if we run the iperf client from the other peer to the server, it will perform a bandwidth test of the specified duration from the iperf client to the server. Note that by doing so we will get the speed of the link from the client to the server.

- **wget** This is a popular HTTP download client for Linux environments. It provides a set of download statistics on the fly, such as instantaneous throughput or download time and progress. In order to use this tool to the download speed from a server we need to set-up first the HTTP server on the machine of interest. In some of our test we used wget sessions to an apache server in a MPTCP capable machine where we previously stored big files for testing purposes.



```
david@mptcp-client:~/Desktop$ wget http://server/1GB
--2014-10-30 17:15:21--  http://server/1GB
Resolving server (server)... 192.168.1.225
Connecting to server (server)|192.168.1.225|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1073741824 (1.0G)
Saving to: `1GB.1'

100%[===================================>] 1,073,741,824 5.87M/s   in 2m 53s

2014-10-30 17:18:14 (5.90 MB/s) - `1GB.1' saved [1073741824/1073741824]
```

Figure 2.15: The wget client can be used to download files using MPTCP.

- **tcpdump** This is a packet analyser for Linux able to capture, filter and display information about the packets being transmitted over the network. This tool can be used to record extensive information about underlying TCP connections which can be later processed for deep analysis in off-line time. Tcpdump will be used in our measurements procedures whenever we need detailed-high resolution analysis of the transmission flows.

- **captcp** Command-line tool for Linux to process PCAP files as those produced by in tcpdump session. It is used in this work to provide detailed performance analysis whenever needed.

- **hostapd** This is a tool for controlling the setting up of access points in Linux. Provided with a configuration file with detailed information about the interface, operation mode, standard and so on, this tool is able to use a wireless interface to create a Wi-Fi network and control it by running in the background as a daemon.

2.5.2. Collection procedures

We will see later how tests in this work are based on throughput measurements. We here describe the procedures we followed to collect these measurements.

As listed before, a variety of Linux tools can be used to take and process throughput measurements. In our experiments we focused on Layer 5 and Layer 4 throughput measurements by using two different tools:

Layer 5 To measure top level throughput iperf can be used, since it can monitor throughput as observed at the endpoint of the multi-flow connection. Despite being a useful tool to know the application level speed and thus the user perception, iperf does not support per-flow bit rate inspection. This is because iperf, just like any other bandwidth monitoring tool, is meant for single-path transmission and simply does not realize it might be multiple flows underneath. On the top of that, iperf provides a minimum granularity[6] of 500ms, which is too high for our purposes in sometimes. This is why sometimes we have not use iperf other than to generate the bandwidth tests or to have a picture of how throughput performs on the application level. In many occasions throughput is being measured in the transport level (L4).

Layer 4 In order to measure per-flow speed, we needed tools to analyse TCP level flows. For that we used captcp which can compute per TCP flow speed with higher resolutions when provided with a tcpdump capture. For accurate measurement of the handover time we set the resolution to 50ms so we can see communication drops longer than that. This should be enough to find transmission gaps which would cause problems in real-time applications[7]

[6]Granularity in this context is the minimum amount of time the analyser can compute the average throughput over.

[7]The ITU says a one-way delay of 150 milliseconds is tolerable in VoIP applications [45]

and at the same time keep us away from the discrete essence of a packet transmission. In other occasions, when the granularity was not so critical we used set captcp to provide granularities between 200ms and 1000ms.

Since the throughput computation by captcp only takes into account the payload of each TCP packet, the only real difference we may find in the measurements is derived from the TCP level not delivering the data right away after reception but waiting either to complete a reception buffer or to receive the packets in the right order. In any case, the difference between L4 and L5 measurements will be zero in average, and small in reception/sending time.

To ease the data collection process, different Linux shell scripts were created using the software collection tools listed before in the described procedures.

As it is clear from this chapter, the use of traffic bundling, and more precisely L4 traffic bundling with MPTCP can bring big performance improvements to the present situation of different coexisting network access technologies. In the next chapters we explore these improvements by setting up different test scenarios involving Wi-Fi and mobile access with MPTCP, running tests on them and analysing the results.

Chapter 3

Test-bed description

In order to characterize MPTCP performance in reality three different test scenarios are going to be proposed in this chapter to cover performance tests in bandwidth aggregation (1), inter-technology traffic handover (2) and mobile traffic offloading (3). The first test scenario, which we called the Multi-path Assisted Roaming (MAR), will show MPTCP's ability to achieve 1 and 2 in a situation of heterogeneous access availability, which we represent by using WLAN networks of different Wi-Fi technologies. The second one, called Mobile Traffic Offloading (MTO) scenario will try to show MPTCP's performance in distributing the load among a Wi-Fi and a mobile link when operated in a smartphone. Finally, the third tested scenario, called Capacity Aggregation in Mesh Networks (CAMN) is somewhat showing 1 again, but this time in a more specific approach called Community Networks which will be explained in the corresponding section. The first scenario is actually split up into two different set-ups, one for covering overall performance measurements in a mobility situation and the one in static conditions with simulated mobility for in-depth measurements. Similarly, the test-bed realization for the CAMN scenario is also divided into two sub-setups so a clearer performance outlook can be obtained.

All the test-beds for these scenarios are implemented in the MRAT multi-purpose test-bed in T-Labs, with the exception of the Mobile Traffic Scenario, which used an LTE and Wi-Fi capable phone running tests directly through the mobile network and a Wi-Fi access point to the Internet in the office. This chapter covers the design and realization of the corresponding test-beds in the laboratory by providing detailed explanations of the scenarios, design decisions and implementation.

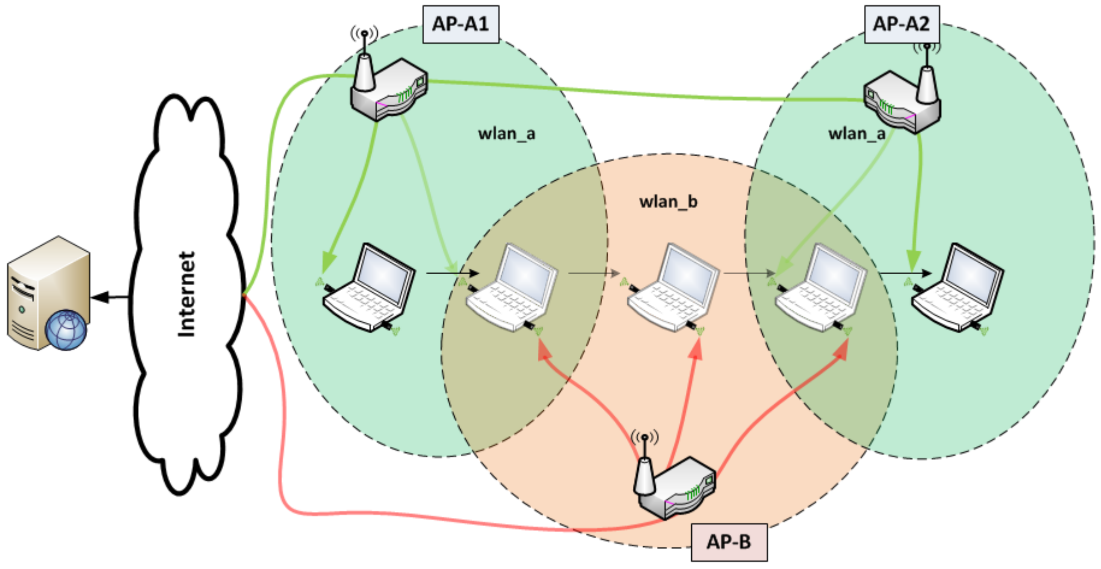


Figure 3.1: MPTCP-assisted roaming.

3.1. Multipath-Assisted Roaming (MAR)

It is a very common situation in mobility that a client needs to hand-off its traffic from one AP to another due to a coverage drop. That is the case in cell telephony when a user moves from one cell to another or in large Wi-Fi deployments which use many APs to cover a wide area. It is even possible (and likely), that traffic need to be redirected from Wi-Fi to mobile or vice versa in a vertical handover. In single-path transmission, a handover implies transmission has to be interrupted for a certain amount of time while either the underlying layers re-associates with a new AP (horizontal handover) or the connection is killed in one path (say Wi-Fi) and re-established in the other (for instance mobile) in a vertical handover. One big challenge in mobility is keeping low the impact in performance during this process. In this scenario MPTCP can be used to reduce off-line time by assisting the handover process, even when an inter-technology handover is mandatory. In Figure 3.1 a client using ”wlan_a” Wi-Fi network relies on a second interface connecting to ”wlan_b” to save a discontinuity in Wi-Fi coverage between AP-A1 and AP-A2. In this example, the auxiliary network is another Wi-Fi network, but MPTCP can potentially use any kind of network to assist the connection handover, which makes it an interesting choice to provide seamless connectivity in roaming conditions. To test the advantages of this application, we have divided the experiments into two different set-ups.

For this first one, we are going to focus on performance as experienced by a client who is moving along a Wi-Fi deployment consisting of a distributed network of APs which he will have to hand his connection to as he moves. APs will generate networks in two frequency bands, representing two different access technologies and therefore giving a view of the protocol's performance when inter-technology cooperation is present. This is the aim of the first set-up, which we called the Dynamic MAR set-up (Section 3.1.1).

Now, when a client moves along a Wi-Fi network the signal quality gets affected by a set of factors. Distance to the access point, shadowing and/or traffic from other clients in the network reduce the quality of the wireless link which leads to packet loss and thus affecting performance of the upper layers. We also wanted to check out how MPTCP performs when we take as many mobility factors as possible out of the equation, so we could measure pure handover response time and bandwidth aggregation performance of the protocol. This is why a second, more specific, scenario is proposed where we make sure none of these factors are affecting our measurements. This is covered in the Static MAR set-up (Section 3.1.1).

3.1.1. Dynamic MAR

This set-up consists of two physically overlapping Wi-Fi networks in 2.4 and 5 GHz implemented in the MRAT to test the situation in Figure 3.1. For that we used a special Wi-Fi deployment in the third floor in T-Labs Berlin (see Fig. 3.2) which is described below in the Configuration section. As the client, who is equipped with two identical wireless cards connected to each network (one to the 2.4 and the other to the 5 GHz network), moves along his path (marked in yellow), the L2 controller will re-associate whenever a better AP for that network is available. Since there is Wi-Fi coverage in both 2.4 and 5 GHz bands all along the testing path and re-association events are unlikely to happen at the same time for both links, we should always have at least one available for the whole duration of the test. Namely, whenever an interface undertakes L2 handover, the other one should make up for any transmission interruption it may suffer due to the re-association process. Even more, the two of them should be available most of the time enhancing capacity in the overall. Under these circumstances, it should be possible to provide increased speed as well as decreased off-line time in the application level in roaming conditions with a multi-path transmission. By the use of MPTCP we expect the client will be able to exploit these path diversity to smoothly jump from one access point to another, keeping a low impact in performance.

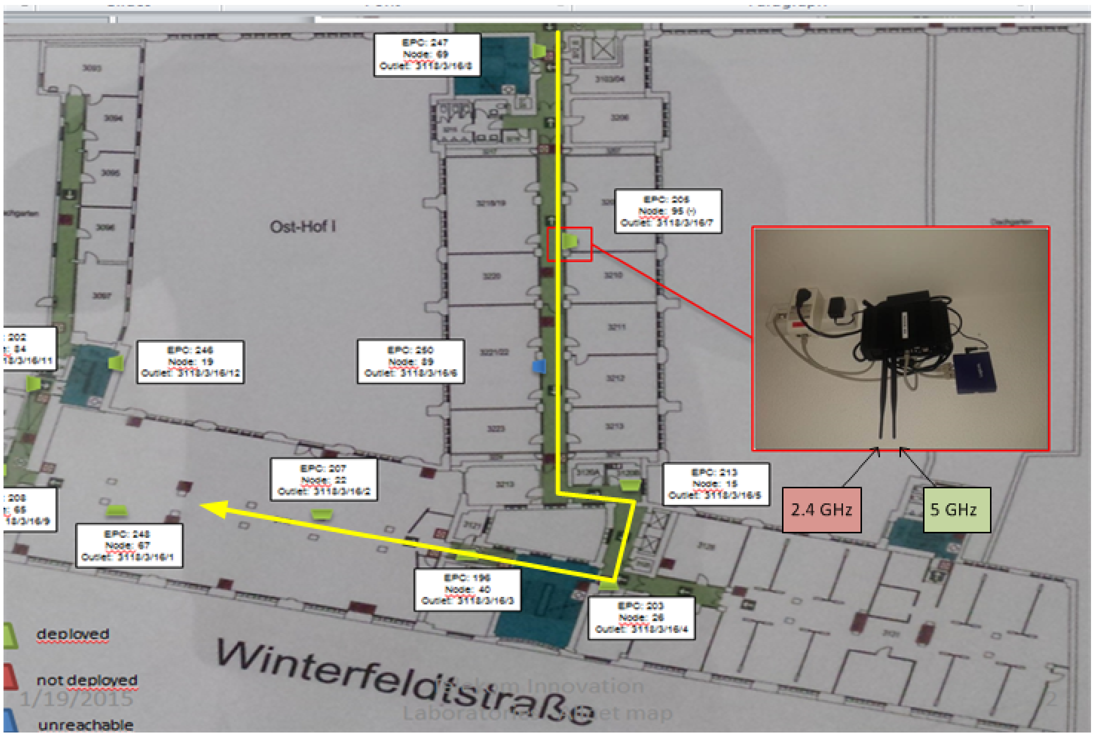

Ost-Hof I
EPC: 247 Node: 69 Outlet: 3118/3/16/8
EPC: 205 Node: 95 Outlet: 3118/3/16/7
EPC: 246 Node: 19 Outlet: 3118/3/16/12
EPC: 250 Node: 89 Outlet: 3118/3/16/6
EPC: 207 Node: 22 Outlet: 3118/3/16/2
EPC: 213 Node: 15 Outlet: 3118/3/16/5
EPC: 248 Node: 67 Outlet: 3118/3/16/1
EPC: 196 Node: 40 Outlet: 3118/3/16/3
EPC: 203 Node: 26 Outlet: 3118/3/16/4
2.4 GHz
5 GHz
deployed
not deployed
unreachable
Winterfeldtstraße

Figure 3.2: Dynamic MAR Set-up.

Configuration

Network A Wi-Fi deployment such as the one in Figure 3.2 was implemented in the MRAT for these set-up. This set-up creates two independent Wi-Fi networks, in the same physical spot by setting up access points in both 2.4 and 5 GHz in nodes 69, 95, 15, 26, 40, 22 and 67. This way we get Wi-Fi coverage in the two bands all along the test path. The APs are created by running two hostapd instances on each node, one for 2.4 and the other one to 5 GHz. Finally, to complete the set-up, the Wi-Fi and Ethernet ports were bridged.

Server The server, attached to the backbone network in the MRAT, was an Acer desktop running a Debian system as the one detailed in B. IP configuration consists of one IP address for the Ethernet interface in the backbone network: 192.168.7.230 . As for MPTCP specific configuration, the path manager is set to `'full_mesh'` and the scheduler, just like every other network parameters, is the default one. CCA is also the default LIA. In addition to this we start an iperf service on port 5001.

Client The client here was the Fujitsu laptop running Ubuntu where we disable the integrated wireless card and use instead two identical wireless adapters (see B). Those are bind to each of the present Wi-Fi networks, one to "wlan_a" and the other to "wlan_b" for the whole duration of the test. IP is manually assigned to be 192.168.7.225 and 192.168.8.225 and IP routes are configured accordingly. The MPTCP configuration is again `'full_mesh'` for the path manager and the default for the scheduler. This means two sub-flows are going to be generated, one going through 2.4 GHz ("wlan_b") and the other through 5 GHz ("wlan_a"). From now on we will refer to these sub-flows as sub-flow B and sub-flow A respectively. Similarly, the paths will be called path B and path A.

Design Decisions

- Decision about using different frequency bands in 2.4 and 5 GHz in the Wi-Fi set-up and not just different channels in the same band was due to previous experience in a similar set-up, which showed that even completely separated channels inside the same band interfered with each other in a multi-path transmission, resulting in poor performance. Choosing different frequency bands took this effect out of the equation.

- Decision of assigning IPs in different subnets to the wireless interfaces was also due to previous experience with MPTCP which showed that the protocol might prevent from mesh sub-flows establishment when IP addresses belonged to the same subnet. Although little conventional, this is not an issue for the development of the tests, provided the routing table in the server is slightly adjusted and the L2 bridging in the APs does not implement any kind of filtering.

Measurements

For these experiment measurements were taken in a L5-fashion (see 2.5.2), with a granularity of 1 second and in upload direction (from the client to the server). 120 seconds long tests were done for both a multi-path and a single-path transmission for comparison. Walking speed, testing time and testing path was kept constant all along the tests. Results can be consulted in Section 4.1.

3.1.2. Static MAR

As we explain later in Section 4.1, L2 performance had a big impact in the tests for Dynamic MAR. To have an insight of pure MPTCP potentials, we looked to isolate our measurements

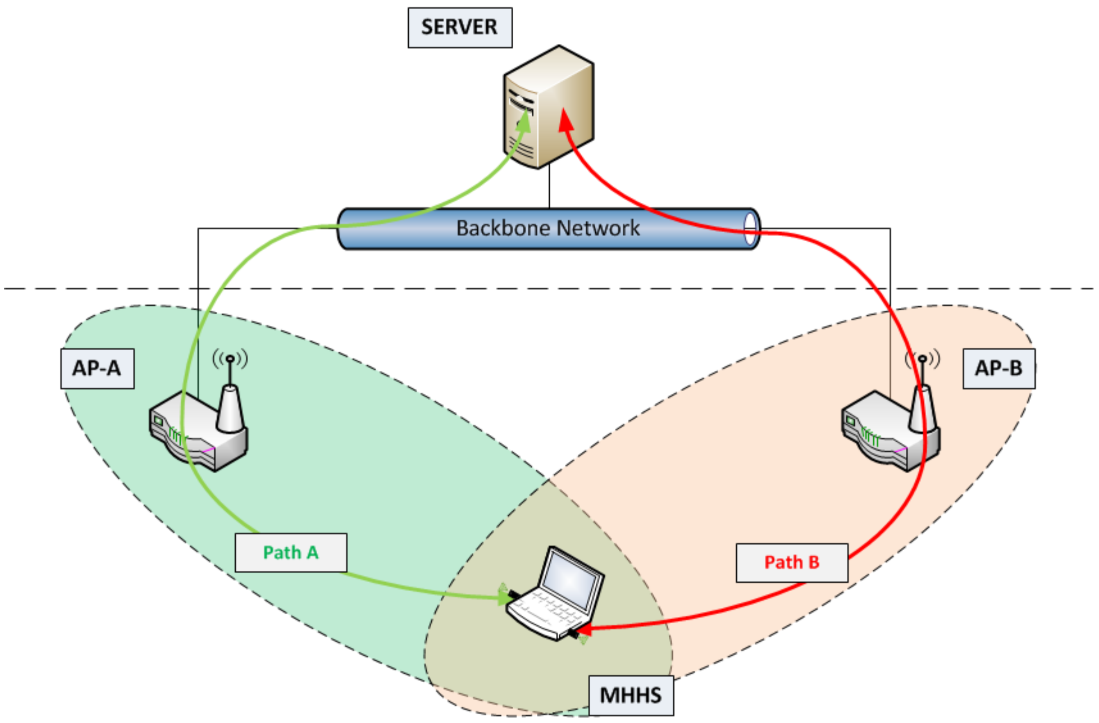


Figure 3.3: Static Spiderman Set-up.

from this L2 performance. For that we have conducted a similar like Dynamic MAR but in a static environment.

In figure 3.3, a client is holding a multi-path transmission with a server. The multi-path flow is formed by one sub-flow going through Path A and another one going through Path B. In such scenario, three situations may occur to the client.

1. **Bandwidth coupling (BC)** Both paths A and B are usable and traffic can be split among them. This situation is ideal to show bandwidth aggregation since MPTCP should be able to push traffic into both links combining their capacity.

2. **Soft-Handover (SH)** Despite the corresponding interface being perfectly associated with the AP, one of the two paths (say path A) is unavailable and thus the client is using only one of them (path B) for transmission. Suddenly path A becomes available back up and the client starts sending traffic on its associated sub-flow. A reasonable time later the first path (path B) breaks and his sub-flow has to stop pushing traffic through. At the end of this sequence, MPTCP should have performed traffic handover form path A to path B with no major effect over the application level transmission. We refer to these sequence

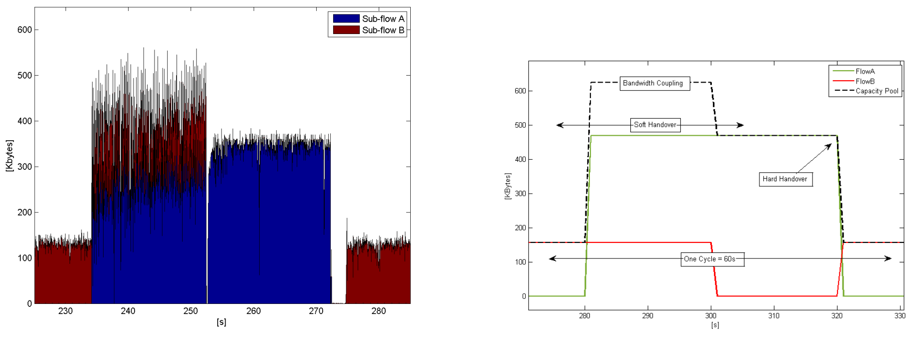
as a Soft-Handover.

3. **Hard-Handover (HH)** While the client is relaying in only one of the available paths for transmission (say path A), this path goes down. At exactly the same time the other path (path B) comes available, enabling another path for the communication to go on. In this case MPTCP should realize of the new state of the network and immediately flip all the traffic from one path to another. We have identified this Hard Handover as the most critical situation that may occur in mobility since in the failure moment MPTCP does not account for a "warmed-up" backup path to continue transmitting right away. It is also a similar situation to a single-path transmission being handed over from one AP to another in a roaming single-homed client fashion. The difference here is multi-homed devices can still save handover time from having the second interface already associated to the AP prior to the failure event.

Instead of looking for these three situations by moving around with our client, in this setup he will be kept static in between two APs providing Wi-Fi networks in different frequency bands (Figure 3.3) and we later force this critical situations by running a couple of scripts in the APs which simulate periodic link failure in the backbone network bringing the interfaces up and down. This periodic link failure went like this:

For the first 20 seconds of each cycle AP-A brings down the interface in the backbone network and only AP-B remains up. After that, AP-A recovers and both AP-A and AP-B are online for another 20 seconds. Then AP-B turns off the backbone interface, leaving AP-A as the only one online. Past 20 more seconds and AP-A and AP-B simultaneously switch their interfaces off and on (respectively) and the sequence starts over.

Figure 3.4b shows the available capacity as seen from the client point of view along one cycle. The black line represents the theoretical instantaneous throughput that the client could achieve by multi-path. Figure 3.4a shows one period from a 10 min test. If we look at it we can clearly see the three expected situations. For the first part of the capture all traffic is going through flow B since path A is down. As it becomes up the client starts transmitting in path A as well. It is in this moment that the maximum throughput is achieved due to bandwidth coupling. When path B breaks all the traffic switches to flow A and SH is completed. In the final part a HH is forced by flipping the availability of the paths, all the traffic need to be suddenly redirected from flow A to flow B and the test returns to the original state. During HH MPTCP took 2-3



(a) Throughput test (b) Available capacity

Figure 3.4: Emulated mobility in Static MAR

s to become aware of the new state of the path pool and adapt transmission.

For comparison, we needed a picture of single TCP performance in a similar environment. This is why the second part of this section covers a similar experiment using single TCP. The scenario in 3.1 for static MAR measurements was slightly modified. The two AP stopped simulating network failures and simply offered two Wi-Fi networks to the client, which again was placed in the middle of both and static. Every 60 seconds the client manually switched from one wireless network to another. This way we could measure how handover (comparable to HH in first setup) affected application layer speed in practice. Besides this few changes, everything remained the same as in the setup for MPTCP measurements.

Configuration

Network For this particular set-up we have used nodes 29 and 75 from the MRAT. We run a hostapd instance on each node to create two Wi-Fi networks in 2.4 and 5 GHz (2.4 GHz on node 29 and 5 GHz on node 75). A different hostapd configuration is tried this time, which improves Wi-Fi performance. It is also mandatory configuring the routing table on each node accordingly and enable the `'ip_forward'` flag in both of them so routing capabilities in the nodes are activated.

```
sysctl -w net.ipv4.ip_forward=1
```

Server Server in the backbone network will be again an Acer desktop (see B). IP configuration consists of one IP address for the Ethernet interface in the backbone network:

192.168.7.230. As for the MPTCP configuration, the path manager is set to `'full_mesh'` and the scheduler, just like every other network parameters, is the default one. We start an iperf service on port 5001.

Client The client is again the Fujitsu laptop in the same interfaces configuration as in Dynamic MAR set-up. IPs are again set manually to 10.0.0.225 and 10.0.1.225. IP rules and routes are configured accordingly. The MPTCP configuration is, like in the server, set to `'full_mesh'` (two sub-flows will be established) and no other major differences with the default configuration are introduced.

Design decisions

A few issues we wanted to address by designing the experiment in this way:

1. In the multi-path tests, by scheduling link failure in this manner we got the network to cyclically go through a set of states which forced the client into performing all three responses of interest (BC, HH and SH) in a predictable way. By doing this we obtained different realizations which allowed later statistical characterization of MPTCP behaviour.

2. In the multi-path tests, decision of placing the link failure in the network segment(APs) and not in the client or the server side was based on two reasons.

   a) Pushing away link failure events from Wi-Fi side: Previous experience showed messing with Wi-Fi may lead to unstable behaviour of the testbed. Bring interface up and down instructions are less stable for WLAN than they are for Ethernet, and flickering Wi-Fi networks would be more difficult to manage by the client's network manager. Observing pure MPTCP behaviour was one of the goals in this experiment and so we looked to isolate this behaviour from lower layers events interference. Moving failure simulation to a different network segment was the way to achieve this.

   b) Keeping the number of flows under control: Also Wi-Fi reconnections triggered from the client side TCP closures and reconnections, belonging to the same application run but making it much more difficult to track, record and process. The same effect would happened if failure was simulated by bringing interfaces up and down in the server side.

3. In the single-path tests, as the client's wireless card needs to switch from one frequency band to another the handover time is increased. This is not an unrealistic assumption because on a real Wi-Fi networks adjacent APs will be as well in different frequency channels and the wireless card will need to do a frequency switch too.

Measurements

For the multi-path experiment, we run a 10 minutes long throughput test from the client to the server (upload direction) to capture many SH, HH and BC events (10 appearances each). Both L4 and L5 throughput measurements are captured, but analysis in Section 4.2 is based on the L4 records, since these are the ones providing enough granularity to look for significant time gaps during handover. For these experiment we set the granularity to 50 milliseconds. Differences between L4 and L5 measurements are negligible in this case anyway.

For the single-path measurements, we run a 20 minutes L5 throughput test (20 handover events captured) with a granularity of one second. L5 measurements are enough in this case because transmission time gaps will last several seconds.

In Section 4.2 we analyse and present the results for this experiments.

3.2. Mobile Traffic Offloading

In figure 3.5, a multi-path capable phone is holding a transmission with a server. Traffic flowing through Path A is using the mobile network and the sub-flow going through Path B uses just Wi-Fi. From both the operator's and the user's point of view, it would be interesting that Path A is used as much as possible to send traffic which otherwise will have to be send through Path B, congesting the mobile network and consuming the users data plan. CCAs in MPTCP were implemented to do exactly this.

The MPTCP traffic balancing feature (see 2.2.3) offers the chance to move away traffic from congested paths to those where congestion is lighter. The scheduler uses information about the RTT and the congestion window to distribute outgoing traffic so load balancing can be achieved. This property makes the protocol an interesting choice for network operators to offload the expensive and usually congested mobile network to cost-efficient WLAN deployments. Another advantage for such application is that, since the MPTCP entity can handle the load balancing, not the application layer neither the user need to be aware of an undertaking traffic offload.

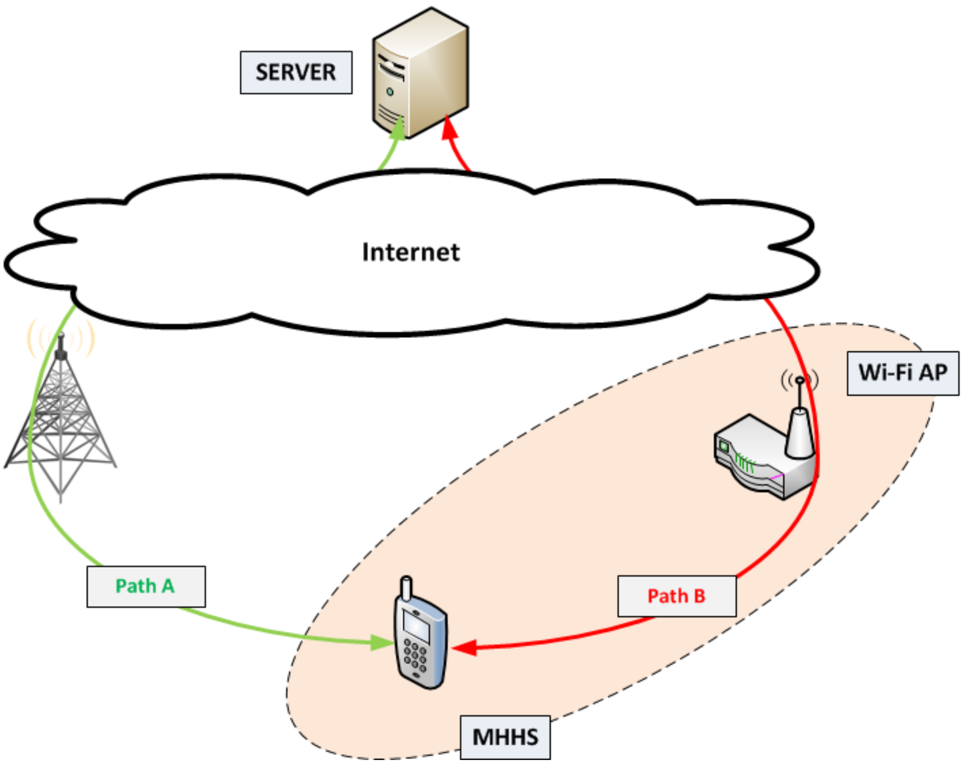


Figure 3.5: Offloading Spiderman Setup.

Note that, when the offloading is full we have a vertical handover from mobile to Wi-Fi.

To test this potential application of MPTCP, the Traffic Offloading set-up is proposed. Similar to the scenario in figure 3.5, an MPTCP capable phone was provided with an LTE and a Wi-Fi connection, and an MPTCP capable server was set-up in the Internet. We later run throughput tests between the client and the server, turning the Wi-Fi path up and down every 30 seconds.

Configuration

Network For this scenario we have tested MPTCP over the actual Internet, having previously checked that no middle boxes along Wi-Fi or 4G paths are interfering with MPTCP flags. Server link has a maximum speed of 100/10 Mbps, whilst mobile and Wi-Fi paths have been tested individually to have a roughly 25 Mbps symmetric average capacity.

Server Server was in this case a 64-bit machine running a 12.04 Ubuntu system properly patched to be MPTCP-capable. It sat in a different physical location in Darmstadt, where our colleagues set it up for us to access it an SSH service. No special configurations are needed in the server or in the local network besides basic NAT redirection at the gateway for those ports we will be using (SSH port for control, HTTP port for downloading tests and one more for video streaming tests.) and the installation of SSH, apache and iperf for testing.

Client The Nexus 5 Android phone listed in B acted as a client for this tests. A detailed guide on how to build, flash and configure an MPTCP kernel for Android OS can be found in the reference webpage for the IP Networking Lab [37]. Although routing configuration is usually mandatory for MPTCP devices using different access networks, this version of the MPTCP kernel already includes automatic routing configuration, and therefore no further adjustment need to be made in this regard.

Design Decisions

If we do upload tests now (like in previous experiments) the bottle-neck would be in the LTE and Wi-Fi links and as a result (having the server a downlink of 100 Mbps) MPTCP would simply aggregate their capacities. Since we are interested on observing how fast and how good MPTCP can move traffic away from mobile to Wi-Fi, we therefore need to place the bottle-neck at the server side by running download tests, i.e., all tests need to be performed with traffic flowing from the server side to the client. This way each path individually is sufficient to fill the pipe and, ideally, MPTCP only needs to use one of them to match the end-to-end capacity (10 Mbps).

Measurements

Download throughput measurement are collected in the server side with the L4 collection procedure [2.5.2] and the output was produced with a granularity of 200ms. Results can be consulted at the Traffic Offloading Results section [4.3]

3.3. Capacity Aggregation in Mesh Networks

Transmission speed provided by the classic DSL line is very sensitive to the distance to the central office. As in rural and suburban areas this distance can be quite high, it turns out that very often subscribers are limited to poor transmission rates in these cases. In the UK, the average broadband line over a DSL technology in the urban areas was 28 Mbps fast, whilst in rural areas the average was only 10 Mbps [46]. This is not a very impressive speed, and as multimedia services are demanding more and more capacity to operate with a minimum quality the figure is getting outdated. In addition to this, the usage rate of this lines is typically low, because of the gusty nature of user's surfing habits. This leads to both a poor Internet surfing experience and the waste of DSL transmission resources, since the line is idle for long time periods. This seems a little contradictory, and that is why some ideas have come out trying to build a more efficient and faster network in such contexts. One of these ideas is the DSL Community, an original idea from T-Labs with aims to use domestic APs cooperating in mesh networks to do it [47].

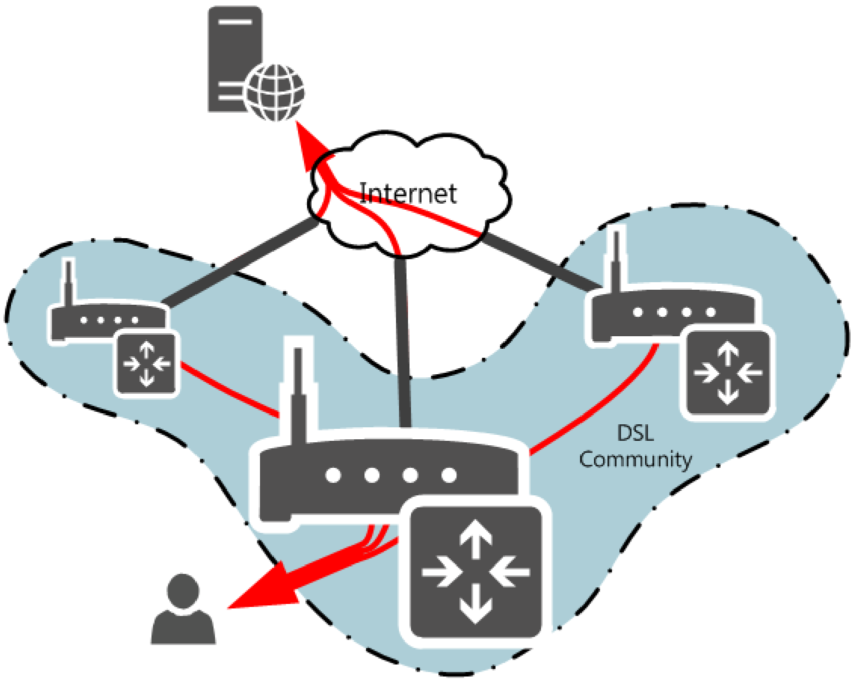


Figure 3.6: The DSL Community aims to provide a better user experience by using cooperating Wi-Fi mesh networks [47].

Commonly, private Wi-Fi networks attached to a DSL line extend further than the physical limits of the subscriber's residence, often reaching homes in adjacent floors, rooms and even more. Provided two of these APs are in range to each other, it should be possible to wirelessly connect both and offer its aggregated capacity by allowing a client sending one MPTCP sub-flow over each link. In a more general scenario, domestic Wi-Fi APs can create mesh networks to share their capacity pool among many users. This solution would be efficient on providing the

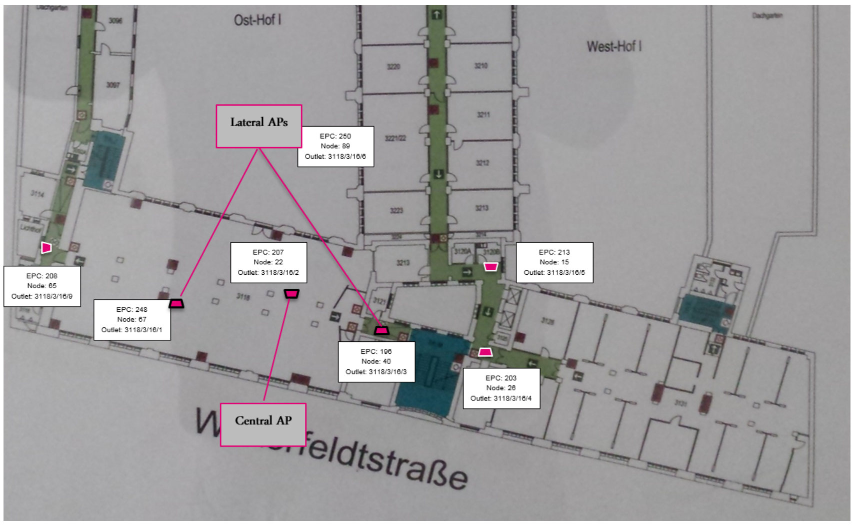


Figure 3.7: Nodes choice for the Capacity Aggregation in Mesh Networks test-bed.

best user experience at no cost for them or the ISP provider and at the same time make a more reasonable use of the set of DSL links available reducing their idle time. MPTCP is one of the choices we have to realize the required traffic bundling so such approach can be realized. In this section we suggest a reference scenario to test this it, and we explain its implementation in a MRAT test-bed.

For this test-bed we are going to make use again of the MRAT to create a network topology similar to Figure 3.6. In the third floor, we select three nodes close as possible to each other which we are going to connect in mesh forming a small DSL Community to do our performance test. Figure 3.7 shows the nodes choice.

Nodes are close enough to each other to be reached wirelessly and form a mesh network. Then, a multi-path capable client connected to the central node triggers a three-flowed throughput test to the server. Each one of the sub-flows is routed through a different router on their way to the server (the central node does the routing). Under these conditions, an MPTCP connection should be able to aggregate the capacity of the three backlinks and the client should see an average speed noticeably higher than that of a single TCP connection. We test the scenario with one, two and three paths available and with different Wi-Fi mesh configurations. On the first sub-scenario, the client will be connected via Ethernet to the central node, which is generating two Wi-Fi networks in different frequency bands so the lateral nodes can associate. This sub-scenario is represented in Figure 3.8. On the second sub-scenario, all connections are done via Wi-Fi, with the client associating to a Wi-Fi network in 5 GHz and the lateral nodes

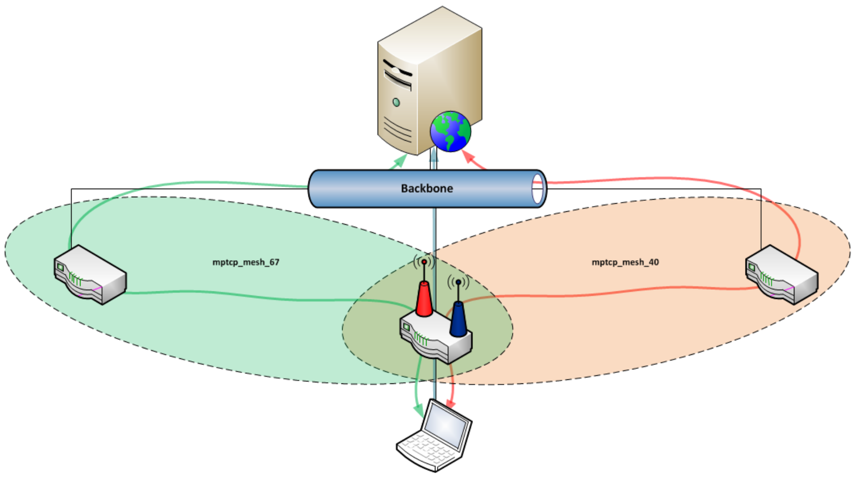


Figure 3.8: The Capacity Aggregation in Mesh Networks sub-scenario one.

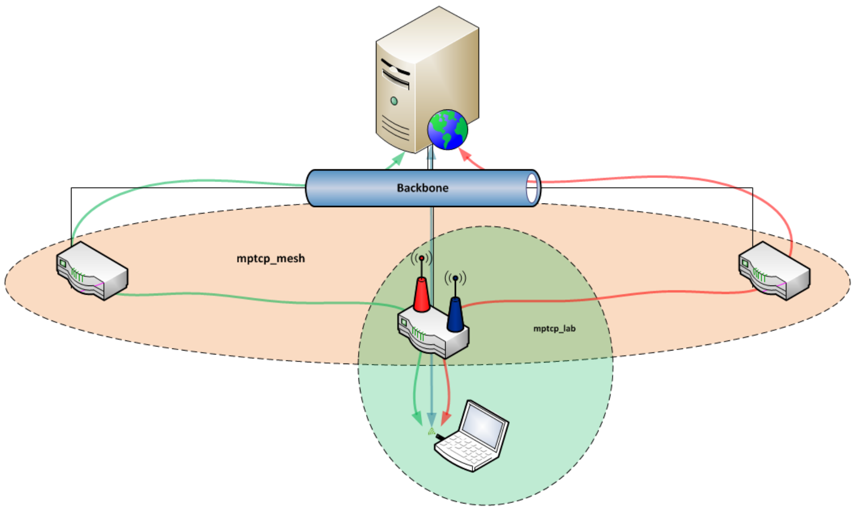


Figure 3.9: The Capacity Aggregation in Mesh Networks sub-scenario two.

doing the same in a 2.4 GHz Wi-Fi network as in figure 3.9.

Configuration

Network To test this application, again we used the MRAT network. In the central node we generate two Wi-Fi networks in different frequency bands. For the first sub-scenario these are "mptcp_mesh_67" (because node 67 will associate to this network) and "mptcp_mesh_40" (for the node 22 to node 40 wireless link) and for the second sub-scenario we create "mptcp_mesh" for both nodes 40 and 67 to connect and "mptcp_lab" for the client wireless connection. Each Wi-Fi network is a different IP subnet, and thus interfaces connected to them are addressed accordingly. Namely, for the first sub-setup interfaces in "mptcp_mesh_67" will have an IP of the form 10.0.0.X and interfaces in "mptcp_mesh_40" will have IP addresses of the form 10.0.1.X. In the second sub-setup the IP addresses allocation is done similarly. Besides this, the most relevant change introduced in the network settings have to do with the routing tables on each node, which had now to be a little bit more complex to properly forward packets in each flow to the corresponding link.

Server The server configuration remains the same as in 3.1.1, besides properly adjusting the routing information. We have assigned three different IPs to the Ethernet interface and an iperf service was also set up.

Client Client configuration was extra simple in this setup. It only had to make sure MPTCP is enabled and the path manager is set to default full-mesh and that it wireless card is associated to "mptcp_lab" in the second sub-setup. No special routing is needed on the client side.

Design decisions

- Nodes choice in figure 3.7 was the best course in this occasion so as to provide the best wireless link quality between nodes. By picking this nodes we made sure the relative RSS[1] to all the nodes was sufficient to hold stable connections through the Wi-Fi paths.

- Whilst in 3.1.1 the network bottle-neck was on the client's access links, for this experiment we moved the bottle-neck one hop further to the APs backlink by manually throttling the their speed with an ethtools instruction.

- Bundling is generated by setting up three IP addresses in the server. We did this because it was required the central node could identify and properly forward the sub-flows through the mesh network. IP routing based on the server's IP was the solution.

Measurements

Tests were recorded from the client side with the L5 method in 2.5.2 the same as in Section 3.1.1.

[1]Received Signal Strength

Chapter 4

MPTCP Performance Evaluation

In this chapter we present the obtained results from the tests performed in the test-bed implementations explained in Chapter 3 and therefore we analyse MPTCP ability to outperform single path approaches in such scenarios.

4.1. Dynamic MAR

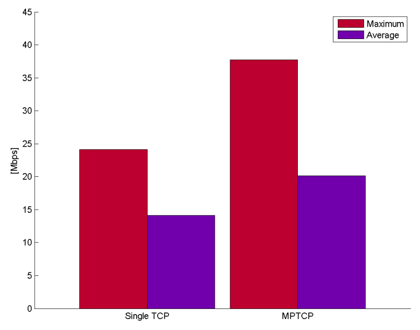
4.1.1. Results

We run two tests along the corridor following the path in figure 3.2. both with MPTCP enabled and disabled (single TCP). To make the tests were as comparable as possible the pace and the total time of the iperf session was similar with both MPTCP and TCP. We are interested here in two kinds of parameters:

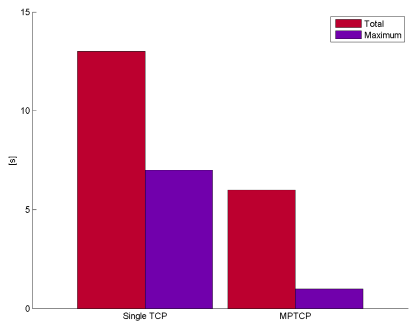
Throughput We take instantaneous throughput measurements along the path to compute maximum and average bandwidth as observed from the client side for both MPTCP and Single-TCP (Figure 4.1a).

Off-line time We account for the maximum consecutive and total time which the throughput was zero along the test to give a view of the delay gain due to multipath (Figure 4.1b).

For this set-up a multipath video streaming demonstration was realized as well. We set up an HTTP video stream in the server with VLC and reproduced it in the client with the mplayer tool, configured to use a streaming buffer of 500ms. In the video we prove MPTCP can perform



(a) Throughput



(b) Off-line time

Figure 4.1: Dynamic MAR results

a smooth handover of the traffic from one AP to another without any cut or quality loss in the video playback [48].

Conclusions

From the results we can see how the client achieved perfect bandwidth coupling during a few intervals in the test, and beat single TCP in the average. The total and maximum consecutive off-line time are also significantly better due to multi-path. However, we can see connection was not seamless, since the maximum off-line time in multipath was 3 seconds. The Static MAR results give more details about why this is happening.

This test-bed showed a high instability, which made it hard to obtain reliable measurements. This was mainly due to two reasons:

- The realization of the test-bed was complex. With so many devices involved, just too many things could go wrong. Often it was one AP unexpectedly going down during a test, the network becoming unavailable or other Wi-Fi networks in the building causing interference which ruined an ongoing test.

- We also had trouble with the L2 manager ruling the decision on when to re-scan for new stronger APs and which ones to connect to. This is because these managers are not aimed for multipath transmission and therefore do not try to exploit the available wireless links form this point of view. Many times the L2 manager connected the two interfaces to the same physical access point (which basically makes multipath useless) or was not active enough on looking for new APs in range.

The results presented in Figure 4.1 correspond to two realizations of the experiment where we were able to maintain the stability of the test-bed for the whole duration of the test. In such case, the multi-path gain pops up and the improvement compared to single TCP is clear in both throughput and off-line time. However, this set-up was not suitable for accurate characterization of such gain, since it was difficult to get many comparable realizations of the tests for both single path and multipath TCP and the results must only be taken as a demonstration of what MPTCP is able to do. Since we are only showing here one realization of the test, it is not possible to provide any error interval for the mean values. The absolute values for the off-line time (Figure 4.1b) are conditioned by the granularity we used in the measurement procedure, which for this set-up was 1 second. The instantaneous values for the throughput have a zero error.

The instability of the Dynamic MAR set-up was the main reason to develop the Static MAR experiments in the first place. These showed similar performance metrics as in Static MAR but with more detail and in a more controlled environment.

4.2. Static MAR

4.2.1. Results

We explained in Section 3.1 how the Static MAR set-up emulates BC, SH and HH situations. Just like in Dynamic MAR we provide here a set of bandwidth and handover performance parameters showing MPTCP behaviour in these situations.

Analysing the tests results, we observed MPTCP showed convergence variation whenever the two paths were available for transmission (BC) with the throughput showing multipath gain for some of the periods and for other showing none or little gain. This is why the "BC sel" figure is included as well in the results summary where only those periods which showed good BC behaviour were included.

Behaviour of the throughput for the Joining and Leaving flow events may as well be divided into "good" and "bad" performing events. Figure 4.3 shows these two situations. Due to this dual behaviour of the throughput during the tests, where a sub-flow damaged its mate sub-flow in some occasions, we have also divided the results for the off-line time during the SH sequence into off-line time during the Joining Flow event and off-line time during Leaving Flow event. The total off-line time during SH is later computed as the summation of both two. This enabled us to provide a more detailed description of the protocol's behaviour during this events.

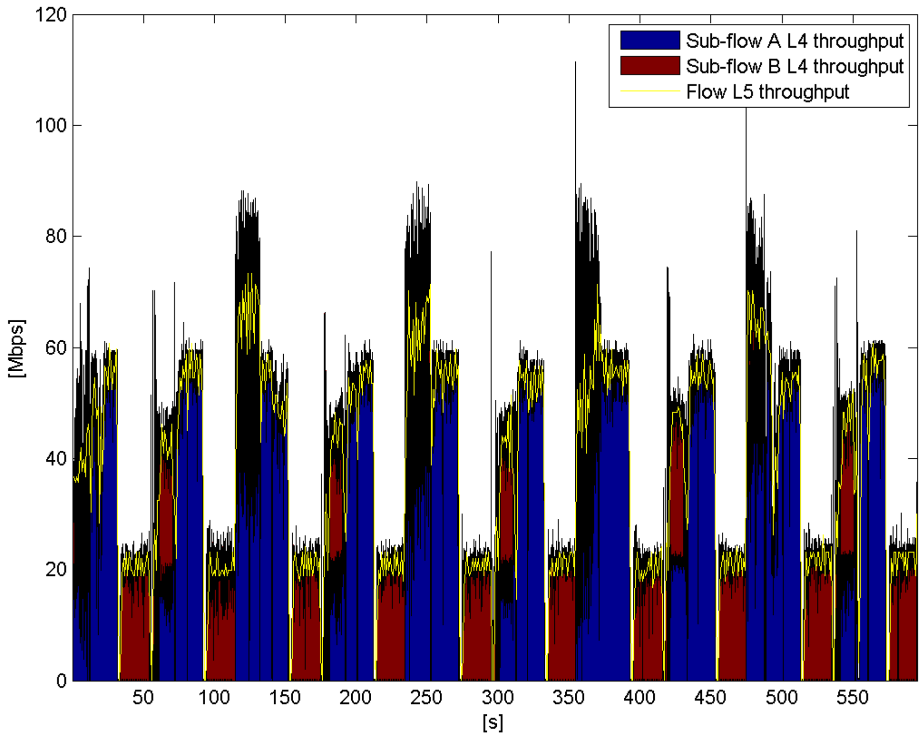


Figure 4.2: Throughput test in Static MAR set-up with MPTCP.

With this in mind, we provide maximum, average and minimum values for a set of events along the test, measured in both throughput and off-line time. These events are:

Throughput Figure 4.4a summarizes this measurements. The bars in the figure correspond to:

- BC sel: Values for Bandwidth Coupling events, computed with a selection of those periods which performed as expected, like it was explained before.
- BC: All periods are taken into account for computation.
- MPTCP-A: Transmission going through Path A only in the MPTCP set-up.
- MPTCP-B: Transmission going through Path B only in the MPTCP set-up.
- TCP-A: Event of transmission going through Path A with single-TCP.
- TCP-B: Event of transmission going through Path B with single-TCP.

Off-line time Figure 4.4b summarizes this measurements. The bars in the figure correspond to:

- Single-TCP: Off-line time during handover events in the single-TCP set-up.
- Hard Handover: Off-line time during an HH event.
- Soft Handover: Off-line time during an SH event.
- Joining Flow: Off-line time during a Joining Flow event.
- Leaving Flow event: Off-line time during a Leaving Flow event.

Similarly to the video demonstration in MAR we explained before, we have also recorded a demonstration of a video streaming in this set-up. Our server on the internet offers a public video stream service which we opened with the VLC android application in the smartphone. Along the demonstration in [49] we successively turn the mobile and WLAN interfaces on an off, forcing a stream handover from Wi-Fi to LTE back and forth. It can be noted how the video never suffers from freezing during the handover.

4.2.2. Conclusions

- Even when it might take a few seconds, transmission always recovered from an HH. We will not require here the protocol to provide seamless traffic handover since this is a very

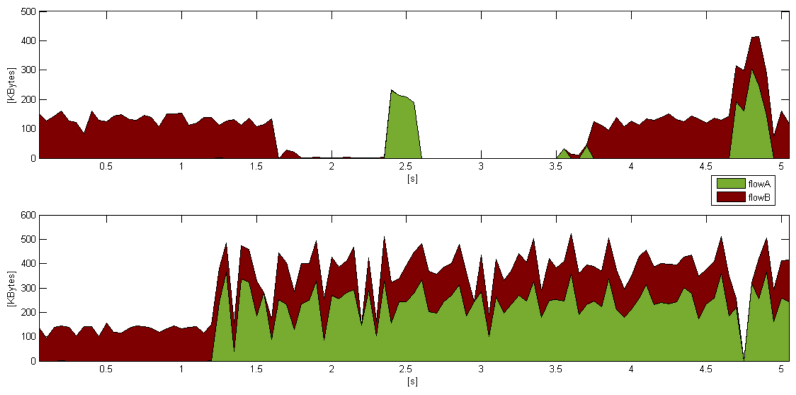
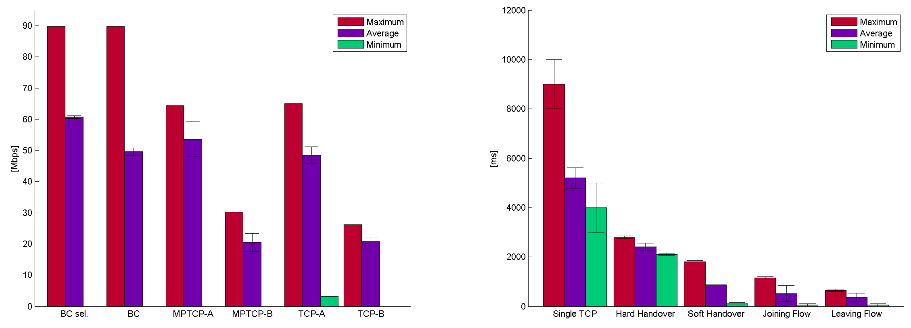


Figure 4.3: Comparison of two different Recovering Flow events.



(a) Bandwidth (b) Off-line time

Figure 4.4: Static Spiderman results

unlikely case which will only happen when one interface loses connection and the other finds one at the same time. In any case, we can see how the HH gap was much shorter than that of the handover time in single-TCP.

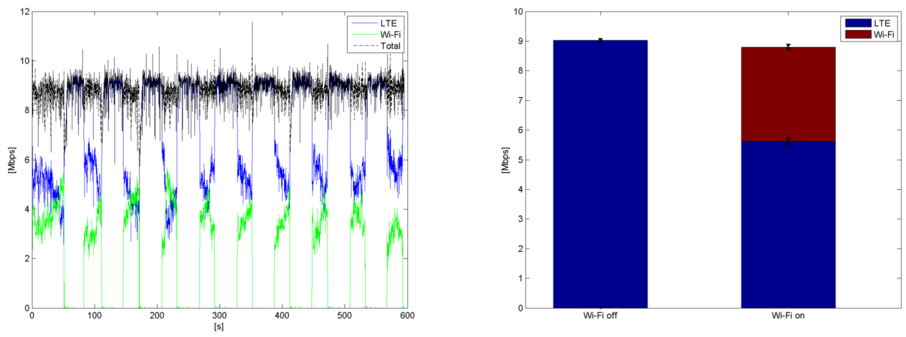
- Many times a Joining/Leaving Flow event hardly affected transmission, degrading it to zero for up to a several seconds. The expected bandwidth gain is not always achieved by the protocol which sometimes even delivers lower speed than a single TCP transmission would. This behaviour is problematic for MPTCP to successfully perform a soft handover and to even be consider as a advantageous alternative to single TCP. It is required MPTCP holders to solve it if the protocol is to be a reliable option for seamless connection in roaming.

- Beyond the described problems, MPTCP did succeed on providing bandwidth aggregation and soft seamless handover in many cases. It is noticeable from the results how the use of MPTCP brought a boost to the application level speed, which sometimes reached up to 90 Mbps (Perfect BC). Also perfect seamless traffic handover was observed several times in the sequence for SH.

- In single-TCP, Layer 5 handover time was about 7 seconds. In the most optimistic scenario, considering L2 hand-off time is in practice usually higher tan 250ms, time handover time will never fall under that figure. When compared to previous results for MPTCP, we can clearly see the multipath gain in handover time. Gaps in transmission caused by MPTCP events, such as a joining flow event or a leaving flow event, never exceed gaps caused by a reconnection event in single TCP transmission.

4.3. Traffic Offloading

4.3.1. Results



(a) Complete test (b) Average throughput

Figure 4.5: Traffic Offloading results

The measurements for the set-up in Section 3.2 are covered here. In figure 4.5a we show one of the recorded tests consisting of a big file download which starts in LTE (blue line) and is occasionally assisted by Wi-Fi (green line). We force this behaviour by periodically triggering a gateway mismatch in the Wi-Fi network with a failure simulator script. Note that the combination of both (black line) roughly matches the servers uplink capacity (10 Mbps), as it was expected in this set-up. Figure 4.5b shows the contribution (in average) to the connection speed both when only LTE is available and when Wi-Fi is assisting the connection.

4.3.2. Conclusions

It can be noticed how the LTE sub-flow is offloaded when MPTCP finds a different path available to share the load. The performance of traffic offloading in this case is not very impressive though. It can be seen in figure 4.5b how Wi-Fi could only offload the LTE link a short 38% when present. The reason for this poor performance is MPTCP congestion control mechanism.

In the introduction to MPTCP earlier in this document we have already explained the principles of LIA, its congestion control mechanism. Shortly, this mechanism uses information about RTT combined with packet loss events to adapt per sub-flow congestion window size and thus transmission rate along available paths. Although RTT for the Wi-Fi path is several times smaller than that of an LTE link, the loss rate is however higher for the former. On the long term, this effect dominates over an the RTT difference leading to the LTE path accounting for most of the traffic share even when Wi-Fi path is ready to be used [50].

4.4. Mesh Capacity Aggregation

For the Mesh Capacity Aggregation set-up we were mainly interest on observe MPTCP ability to combine the back-links of the nodes, which were simulating home DSL routers, into a single one which would be the summation of all. In Figure 4.6, which correspond to the obtained results for the first sub-scenario with Ethernet connection between the client and the central node, the throughput gain due to multipath can be perfectly observed. The first set of columns correspond to the averaged throughput measurements as observed from the client side for a one minute long throughput test when we enable only one sub-flow for the connection (going though the central node). Similarly, sets of columns two and three shows our measurements for the same test, but enabling the lateral sub-flows (going through the lateral flows only) and finally with all possible paths enabled. The expectation was that, enabling all flows the capacity pool of the nodes, i. e. the summation of all backlink, could be used for the client to maximized speed. In this case, the maximum theoretical throughput we could achieve was 30 Mbps, since the backlink of the nodes had a link layer capacity of 10 Mbps. As we can see from the results, the maximum in roughly achieved with the Ethernet set-up, since the client enjoyed a 26 Mbps connection when all paths were available.

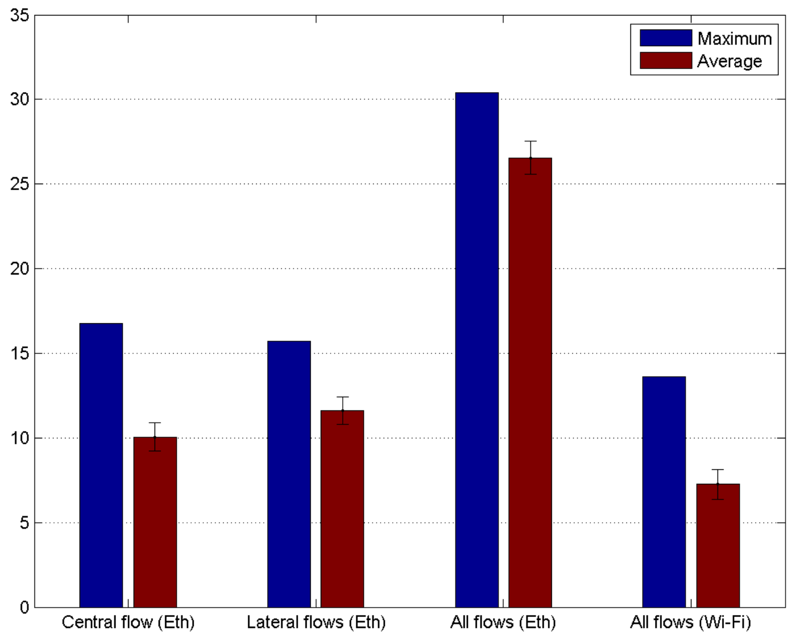


Figure 4.6: Throughput offered by the DSL Community for the central node only, the lateral nodes only and all nodes available.

The results obtained for the pure Wi-Fi sub-scenario, where a wireless link is used as well in the client to central node link, were however very different. When we did exactly the same experiment with the lateral nodes connecting to the central nodes in the same frequency band (with the central node operating in AP mode for the other two) we realized the throughput observations were by no means as close from perfect capacity aggregation as those for the firs set-up. Moreover, the throughput measurements showed an extremely high variance and the average observed throughput after 60 seconds tests was never even reached the expected average throughput for one sub-flow (10 Mbps). The last bar group in Figure 4.6 shows one of the result for the pure Wi-Fi setup. After further investigation, we found the reason for this poor behaviour was situation of Hidden Terminal Problem.

The Hidden Terminal Problem happens in a wireless network whenever a terminal in the network is visible (in range) from the AP but not from other terminal in the network. In such case, the collision avoidance mechanism of the Wi-Fi MAC layer will fail, and transmitting stations will interfere each other, because they will try to use the media at the same time. Recalling set-up in Figure 3.9, what we had in practice was a situation where a quite good

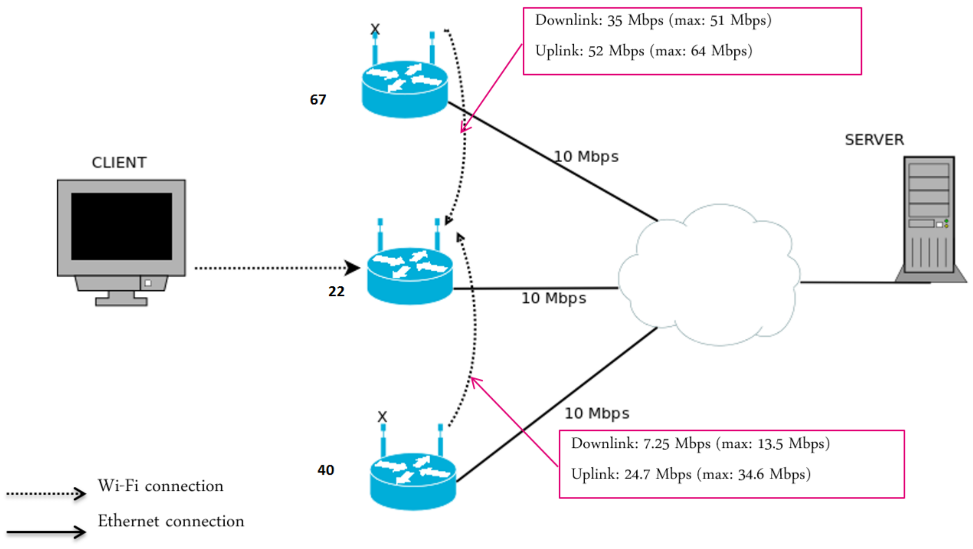

Downlink: 35 Mbps (max: 51 Mbps)
Uplink: 52 Mbps (max: 64 Mbps)
X
67
CLIENT
SERVER
10 Mbps
22
10 Mbps
X
40
10 Mbps
Downlink: 7.25 Mbps (max: 13.5 Mbps)
Uplink: 24.7 Mbps (max: 34.6 Mbps)
Wi-Fi connection
Ethernet connection

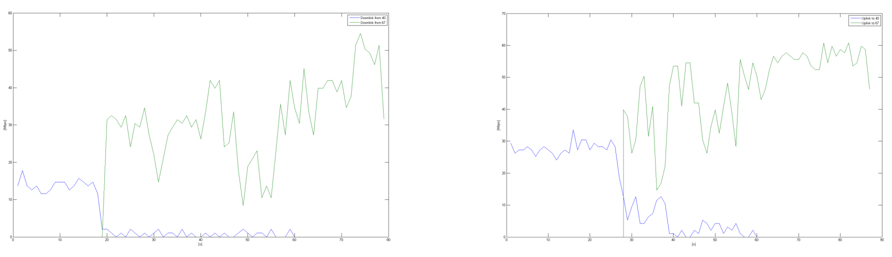
Figure 4.7: Sub-scenario two was potentially a Hidden Terminal situation.

wireless link was connecting node 67 to node 22 (the central node) and less healthy one was connecting node 22 to node 40. This was because there was a door sitting in the middle of this wireless link.

Although each link separately was fast enough to maintain the lateral sub-flows (individual link capacity of the 67 to 22 and the 22 to 40 links can be found in Figure 4.7), when used at the same time their aggregated capacity dropped to nearly zero due to interference caused by one node not recognizing the other was transmitting as well. This specially a problem in the downlink (from the server to the client), since packets would collide in the middle, as much as in the uplink due to colliding of TCP acknowledgements. Figure 4.8a shows the impact in the throughput performance of the node 40 to node 22 link when of a hidden node 67 starts an iperf test to node 22 as well in the other side of the corridor (downlink situation). Similarly, Figure 4.8b shows the uplink situation.

4.4.1. Conclusions

- Results for the first sub-setup, with wired connection between client and server, proves MPTCP is a suitable traffic bundling technology to achieve backlink pooling in the DSL Community vision, provided the mesh network is able to route the traffic accordingly. Designing and implementing such mesh routing algorithms is indeed an issue, but any realization of the DSL Community will have to deal with that problem at any rate. More-



(a) Downlink (b) Uplink

Figure 4.8: Hidden node transmission impact in performance.

over, MPTCP allows for the dynamic generation of new sub-flows from either the client or the server side, and can cope with the network shutting down sub-flows unilaterally without affecting the overall connection. This MPTCP mechanisms can be useful for dynamically adaptation of DSL Community user's connection to the backlink pool. In the Future Works section [.3] a mechanism is proposed to trigger the generation of new sub-flows at the initiative of the network as well.

- From the point of view of the realization of the DSL Community, the Hidden Terminal Problem we had to deal with here can be an issue. This is actually a tricky problem to undergo in wireless networks such as those mesh wireless networks indispensable so the DSL Community can be useful.

We have showed in this section can MPTCP can be a viable option to provide better throughput to the users in a situations of high diversity of access, achieving bandwidth coupling in many occasions during our tests. More over it outperformed single-TCP in both average throughput and transmission delay in every test, which, together with the inherent extra resilience provided by a multi-path connection makes MPTCP a very interesting choice to give high connection quality. It also showed promising results as a candidate technology to address backlink pooling in the DSL Community approach. The results for the traffic offloading application were less positive, with only a 38% of saving in the mobile network traffic in our tests. However, more congestion control algorithms need to be tested before discarding MPTCP as a candidate here.

Chapter 5

Personal conclusions and Future Works

.1. Summary

In the previous pages we have shown top level the tendency of the access network to become more and more multipath and explain the benefits a technology capable of exploiting this multipath would bring. In this sense, we have identified here the traffic bundling as a very interesting approach to provide bandwidth aggregation, vertical handover and seamless mobile traffic offloading in today's access context.

We have further explored the traffic bundling approach and in particular the traffic bundling technology which we wanted to test in this work, the Multipath TCP protocol. Basing on our hands-on experience with MPTCP all along the related experiments and research, we provided a detailed explanation of its goals and technical operation which would be useful for interested readers to understand it, configuring it and getting to install it on their own devices. We also showed how the use of such traffic bundling technology can bring big improvements to a connection's speed, response time and reliability, and presented a set of tools which can be of use for network performance analysis in this context, talking about the MRAT multi-purpose network which was the core of our test-beds implementation. It is interesting the compendium of software tools we related here to realize performance analysis of the protocol looking at different network performance metrics. We believe our work putting together these tools and our experiences with them, which we have also included here can be useful for similar experiments in the future.

An evolution from the top level traffic bundling advantages we presented in the Chapter 1 to a testing scenario and finally to a test-bed set-up for real performance tests was related as well. This could of use for the reader to go ahead and replicate our experiments when interested or to find design flaws which can be used to rebate our results further in the evaluation chapter. These descriptions can be helpful not only to understand the results in Chapter 4, but also to find out whether this experiments and therefore the results and conclusions of this work are relevant for the reader purposes.

Finally, an evaluation of the performance tests run over the test-beds described in the previous chapters as well as our in-field experiences for the duration of the project was done. Here, we not only show different demonstrations performed over the test-beds which the reader can look at to observe functional MPTCP-based applications, but also accurate measurements for throughput and delay were possible sometimes. When such accurate characterization was not possible, approximate figures are provided so as to give an outlook of MPTCP performance in those cases. We include as well our conclusions about these results, offering possible explanations for those which were unexpected and about the feasibility of some scenarios. To sum up, we showed how MPTCP beat or equalled single TCP in almost any scenarios in terms of achieved bandwidth and vertical and horizontal handover, and how it offers new possibilities for mobile traffic offloading. In the DSL Community the protocol proved to be a natural choice for the implementation.

.2. Personal Conclusions

From our point of view the protocol has sufficiently proved potentials in every analysed scenario, being relevant the protocol's ability to adapt transmission to the network context (transferring traffic from Wi-Fi to mobile or vice-versa, relying on WLAN to offload part of the traffic from the mobile network and even aggregating the capacity of many gateways) without intervention of the application layer. It allows all those things to happen in the underlying network transparently to the user, that does not need to be aware of transmission nicely adapting itself to suit the possibilities offered by the network, this making communications more efficient. In the video streaming demonstration for the Dynamic MAR set-up, MPTCP is keeping the user away from any disruption in the playback quality due to roaming by relaying on the multipath, but, with a minimum off-line time of 50ms in soft handover, the protocol can also be a candidate to host real time traffic such a phone call or a video call. This is it as far as the found advantages are concern.

Regarding the disadvantages, I believe the most worrying thing we get form the test is the random protocol's behaviour in the Static MAR tests. We believe this behaviour is mainly caused by the congestion window coupling in the CCA used by MPTCP. This algorithm fulfils a set of requirements regarding protocol fairness and adaptation to the network's congestion, but it has not proved to be optimized yet to hold real time transmission. It would be interesting to analyse the performance of the new implemented CCAs as referred in Section 2.2.3 in these scenarios and even new CCAs can be proposed in this sense to overcome this instability problem. MPTCP CCA was also an issue in the traffic offloading scenario in Section 3.2. LIA CCA (the default CCA in the reference implementation) has shown to be inefficient in this tests, with low offloading performance.

A question raises as well on whether the protocol offers such improvement with respect to single TCP that would pay for hosting services to upgrade their safe, fairly well performing TCP stack to MPTCP. It is true that MPTCP can bring a boost to transmission performance and therefore improve the user experience, but the present single path solution is already satisfactory for most of user needs and upgrading from TCP to MPTCP would requires an extra effort form the servers marketplace's players. It is questionable that they will assume the cost of this upgrading when not doing it would most likely come at little or none impact for their business. I personally believe that, in order for MPTCP to become an attractive technology for them its benefits should be clearer, form their point of view, than they are in our experience.

Apart from this, the multipath transmission (or more precisely multi-homed transmission) has the inherent drawback with respect to single-homed transmission of an increased energy needs. No matter how effective the implemented energy control policies are in a multi-homed device, multi-interface transmission it will always consume more energy than if it only used one interface. This is because, regardless of how much the interfaces are actively transmitting and/or receiving, keeping many interfaces connected is more costly than keeping up just one.

.3. Future Work

Here we include a relation of future steps which are believed would complete the research undertaken in this work. We propose a set of further tasks of interest to have a better outlook of the protocols performance in some of the scenarios as well as some suggestions to extend the protocol's integration in the system:

- The first proposal has to do with the observed under-performance in the Traffic Offloading scenario in Section 3.2. The conclusion we analysing the results for this test-bed was that the achieved traffic offloading was quite low, the reason probably being a bad choice for the CCA in use. Since the reference implementation of MPTCP for Linux already offers other CCAs to handle congestion in multipath, one first step to look into improving traffic offloading performance would be to test this other algorithms in the same scenario and find out whether a better result can be obtained with those. The problem with LIA, the CCA used in this set-up, is that, as we already explained tends to set the convergence throughput for each path as a function of both the RTT and loss rate, causing the algorithm to slightly prefer those paths with a lower loss rate even when they may have a higher RTT. However, wVegas (also available as CCA) mainly relies on RTT measurements to allocate the traffic and it could be that this algorithm shows a better performance in this scenario, maybe as better as achieving full offload from the mobile network to the WLAN path when available. A broader exploration of other CCAs in this scenario would be interesting and would probably have as an outcome that other choices are preferable for a better traffic offloading performance. To do this, we propose the realization of a simulation based study of all available protocols in any of the network simulation platforms with MPTCP support today. The ns-3 network simulation can be an option for this, since it already includes support for MPTCP simulation.

- Another thing the MPTCP project lacks of is communication with the Network Manager, which can enable the MPTCP engine to make better choices on how and when to use a multi-homed device's available access links. We talked in the Introduction about the Network Manager and how this pieces of software is a key player on the realization of efficient multipath communications. The MPTCP engine could use information about which network interfaces are more expensive to use, more energy consuming or how important is for the user the connection robustness, speed and delay to adapt the multipath transmission to this preferences. For instance, a user may configure the "Costly Efficient" mode in the Multipath Network Manager (MNM), which therefore will prohibit MPTCP to transmit over the mobile interface meanwhile a WLAN connection is available. The "High Robustness" mode can, for instance, tell MPTCP to flood all available links with redundant copies of one stream so communication can go on at the least time cost whenever one of them fails. Configuring one "Full Speed" mode in the MNM will tell MPTCP to use extensively used all links to transmit data as fast as possible aggregating the capacity of all the available links. All this requires the Network Manager to become smarter and to be able to translate information about the user preferences, the link type, the cost... into MPTCP configuration. If we remember the problems we had in the Dynamic MAR experiments in Section 3.1.1 with a L2 controller which was not mature enough to deal with multipath a sense, we also see how such a MNM could be beneficial in this situation as well. This is why we propose here a project to develop a Multipath Network Manager which can centralize all this intelligence and to move forward into a better integration of the MPTCP paradigm inside the present system.

- Finally, another idea has come out our experiences trying out MPTCP as a technology to support the DSL Community approach. The bare MPTCP at the endpoints approach we used here, setting up the MPTCP entities at both the client and the server has the important drawbacks of (1) not being compatible with non MPTCP-capable web servers and (2) not supporting the dynamic joining of new gateways in the DSL Community to an ongoing connection, since triggering of new sub-flows can only be done form either the client or the server side, which are not aware of the DSL Community status. To solve this issues two approaches are proposed. The first one how be to use an MPTCP-in-the-middle scheme such as the one described in Section 2.2.1, with a proxy on each home station in the community. Assuming the mesh network forming the DSL Community has all the information regarding the network topology and the gateways availability, this would enable the home station to establish the corresponding sub-flows and to join and cut new gateways to the connection as they become available. In case (1) is not a problem, a semi MPTCP-in-the-middle scheme can be used instead the proxy standing only in the client side. Another approach here, requiring no proxy, could be virtualizing both the home station and the client wireless interface, so the sub-flows could be triggered at the client side by generating new virtual access points at the home station.

Appendix A

Budget

In this appendix, a detailed view of the project phases is presented together with an estimation of the budget for the execution of this project. The total cost of this project is computed taking into account both material and human resources expenses based on the project phases discussed in Section 1.2 and detailed below. The amounts are expressed in euros.

A.1. Project planning

As explained in Chapter 1, the realization of this project followed a set of tasks and sub-tasks. These are:

1. Hands-on phase: Getting in contact with MPTCP protocol.
   a) Study of the reference MPTCP implementation from UCL Louvain.
   b) Installation and configuration of MPTCP on a Laptop and Desktop for tests.
   c) Hands-on test of the protocol in a basic ad-hoc network.
   d) Research on suitable tools for network monitoring and testing.
   e) Hands-on tests with the MRAT network.
2. Test-bed design and implementation: Move from an scenario of interest into a test-bed design and implementation.
   a) MAR test-bed design and implementation in the MRAT.
   b) Experiments run and measurements collection.

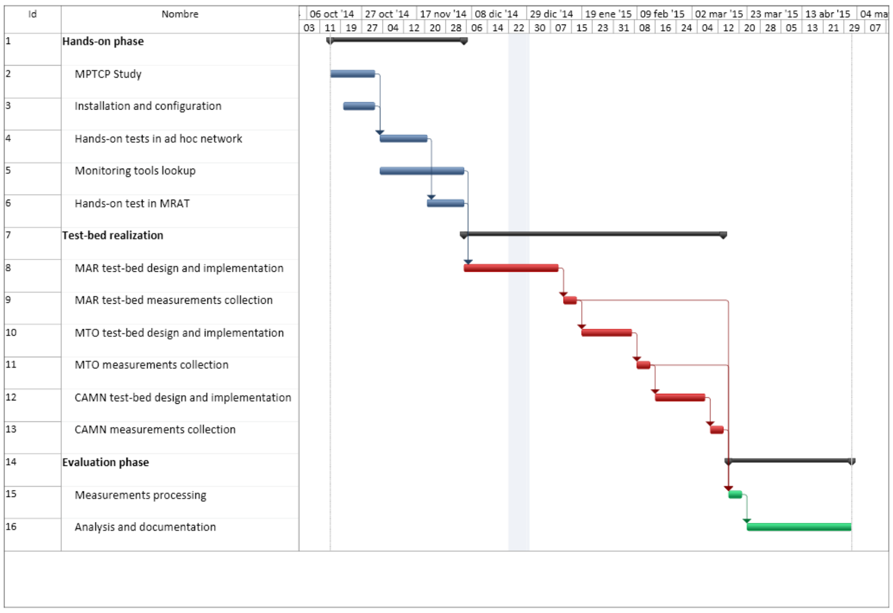


Figure A.1: Gantt diagram of the project.

c) TMO test-bed design and implementation.

d) Experiments run and measurements collection.

e) CAMN test-bed design and implementation in the MRAT.

f) Experiments run and measurements collection.

3. Evaluation phase: Going through the collected measurements and analyse them.

a) Analysing the data in Matlab.

b) Documentation and conclusions.

The Gantt diagram in Figure A.1 refers every task and sub-task time span inside the project's development.

A.2. Equipment expenses

Here follows a relation of the amortization cost of all the equipment used in the project (Section B) to compute the whole expense in materials. The amortization is calculated for a six months period for every device but the MRAT nodes, which were roughly used during the

Concept	Units	Cost (€)	Lifetime (months)	Amortization Cost (€)
Fujitsu Laptop (B.1)	1	500	60	50
Acer Desktop (B.2)	1	300	60	30
Nexus 5 (B.4)	1	250	48	31.25
Wireless Adapters (B.5)	2	30	60	10
Nodes (B.3)	10	500	60	250
Total				371.25

Table A.1: Amortization cost.

Level	Working time (hours)	Gross Salary (€/hour)	Total cost (€)
Junior	1104	30	33120
Senior	48	40	3840
Total human resources expenses			36960
Total project cost			37331.25

Table A.2: Human resources cost.

test-bed implementation phase only (three months). The total lifetime for the equipment was assumed to be five years for the computers, the nodes and the externals wireless adapters and four years for the smartphone.

A.3. Human resources expenses

To computes the total expenses in human means we will be assuming the whole project was executed by two junior engineer along six months, with the assistance of a third junior engineer and the supervision of a senior engineer for two and two hours per week approximately. The total work time per day is 8 hours, and we assume 22 working days per month in average. Per hour salary is assumed the average salary in Germany, for a graduate in the case of junior engineers and for a high level engineer in the case of senior.

Appendix B

Equipment

Here follows a relation of the whole physical equipment used in this project. When possible a technical specification of such devices is also provided. It is as well included a list of all the key drivers and software pieces we had to use in the set-ups.

B.1. Hardware

Here follow an specification of all hardware equipment used in the different testbeds.

B.1.1. Network Nodes

- Fujitsu Laptop: RAM 8GiB, Processor 64-bits Intel Core Duo @ 2.80 GHz (two cores). Used as the client. B.1

Figure B.1: Fujitsu Laptop

- Acer Desktop: RAM 4GiB,Processor 64-bits Intel Atom @ 1.80 GHz (four cores). It hosted the server in most of the occasions. B.2

Figure B.2: Acer Desktop

- Black Node: RAM 4GiB, Processor 64-bits Intel @ 1.80 GHz (four cores). Forming the testbed network mostly. B.3

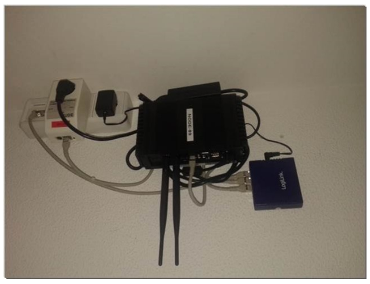


Figure B.3: Network Node

- Google Nexus 5: RAM 2GiB, Processor 32-bits Qualcomm Snapdragon 800 @ 2.26 GHz (four cores). Client in some tests. B.4

Figure B.4: Google Nexus 5

B.1.2. Wireless Adapters

- USB External Wireless Cards: Buffalo Atheros AR7010+AR9280. Used in the client laptop as the main wireless adapters. B.5

Figure B.5: External wireless card.

- Integrated Wireless Cards:
    - Fujitsu Laptop: Intel Corporation Ultimate N Wi-Fi Link 5300.
    - Network Nodes: Qualcomm Atheros AR9462, Qualcomm Atheros QCA988x 802.11ac and Qualcomm Atheros AR5418.

B.2. Software

Here follows a relation of the most relevant software pieces installed along the testbed:

B.2.1. OS Platforms

- Fujitsu: Debian Wheezy 7.7. Press enter on boot up to select booting from CD/DVD drive. This laptop does not support booting from a removable media.
- Acer Laptop: Burn an ISO image to a USB stick and press F12 on boot up to select booting form removable media.
- Server: Debian Wheezy 7.7. ISO images from the Debian official.
- Development kits: Xubuntu Trusty Tuhr 14.04. ISO images from Xubuntu official.
- Black Nodes: Ubuntu Server 14.04.1 LTS from the Ubuntu official. DEL will get you to booting selection.

B.2.2. Drivers

- firmware-atheros for Atheros adapters: Install it by typing

```
apt-get install firmware-atheros
```

- firmware-iwlwifi for Intel adapter: Install it by typing

  ```
  apt-get install firmware-iwlwifi
  ```

  You can also find de source here

- You can also install the drivers for a Ralink chipset by typing

  ```
  apt-get install firmware-ralink
  ```

  In Ubuntu distributions you can install them straight, if not installed already. In Debian it is possible apt-get does not find the repository. If so you just need to add the following line to the file /etc/apt/sources.list

  ```
  deb <nowiki>http://ftp.de.debian.org/debian</nowiki> stable main contri
  ```

Bibliography

[1] Ericsson, "Ericsson mobility report," tech. rep., 2015. http://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf.

[2] "iPass Wi-Fi Growth Map." https://www.ipass.com/wifi-growth-map/. Accessed: 15-10-2015.

[3] E. Perahia, C. Cordeiro, M. Park, and L. L. Yang, "IEEE 802.11 ad: Defining the next generation multi-Gbps Wi-Fi," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pp. 1–5, IEEE, 2010.

[4] R. Braden, "Requirements for internet hosts - communication layers," STD 3, RFC Editor, October 1989. http://www.rfc-editor.org/rfc/rfc1122.txt.

[5] "Approved IEEE Draft Standard for Information Technology-Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, (Superseded by IEEE 802.3-2005)," *IEEE Std P802.3REVamD2.2 Part 4 Apr*, 2005.

[6] "Cisco's EtherChannel solution." http://www.cisco.com/c/en/us/tech/lan-switching/etherchannel/index.html. Cisco's website. Accessed: 15-10-2015.

[7] V. Hunter, J. Regan, A. Nothaft, and A. Duggal, "Automatic load sharing-trunking," May 4 2004. US Patent 6,731,599.

[8] R. Lapuh, Y. Zhao, W. Tawbi, J. M. Regan, and D. Head, "System, device, and method for improving communication network reliability using trunk splitting," Feb. 6 2007. US Patent 7,173,934.

[9] R. Lapuh and T. Homma, "Routed split multilink trunking," Dec. 9 2008. US Patent 7,463,579.

[10] W. Simpson, "The point-to-point protocol (ppp)," STD 51, RFC Editor, July 1994.

[11] K. Sklower, B. Lloyd, G. McGregor, and D. Carr, "The ppp multilink protocol (mp)," RFC 1717, RFC Editor, November 1994.

[12] K. Evensen, D. Kaspar, P. Engelstad, A. F. Hansen, C. Griwodz, and P. Halvorsen, "A network-layer proxy for bandwidth aggregation and reduction of IP packet reordering," in *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pp. 585–592, IEEE, 2009.

[13] D. Phatak and T. Goff, "A novel mechanism for data streaming across multiple ip links for improving throughput and reliability in mobile environments," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 773–781 vol.2, 2002.

[14] R. Stewart, "Stream control transmission protocol," RFC 4960, RFC Editor, September 2007. http://www.rfc-editor.org/rfc/rfc4960.txt.

[15] I. van Beijnum, "One-ended multipath tcp (darft)." http://tools.ietf.org/html/draft-van-beijnum-1e-mp-tcp-00, 2009.

[16] C. Raiciu and C. Paasch, "Multipath TCP, Presentation at Google Talks." https://docs.google.com/presentation/d/1KR2UuoODrkGuzeAYiTmR6j4NKD7vmOnWLhAdNX1Y1D4/, 2012.

[17] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," RFC 6824, RFC Editor, January 2013. http://www.rfc-editor.org/rfc/rfc6824.txt.

[18] M. Allman, V. Paxson, and W. Stevens, "Tcp congestion control," RFC 2581, RFC Editor, April 1999.

[19] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," RFC 6356, RFC Editor, October 2011. http://www.rfc-editor.org/rfc/rfc6356.txt.

[20] N. Gast, "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP, Nicolas Gast 2014 (T-Labs and TU-Berlin)," T-Labs and TU-Berlin, IETF.

[21] J. H. S. L. A. Walid, Q. Peng, "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP," Bell Labs, Caltech and Samsung Electronics, IETF.

[22] Y. C. Mingwei Xu and E. Dong, "Delay-based Congestion Control for MPTCP," Tsinghua University and NDSC, China, IETF.

[23] L. Boccassi, M. M. Fayed, and M. K. Marina, "Binder: a system to aggregate multiple internet gateways in community networks," in *Proceedings of the 2013 ACM MobiCom workshop on Lowest cost denominator networking for universal access*, pp. 3–8, ACM, 2013.

[24] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 5–12, 2005.

[25] P. Key, L. Massoulié, and D. Towsley, "Combining multipath routing and congestion control for robustness," in *Information Sciences and Systems, 2006 40th Annual Conference on*, pp. 345–350, IEEE, 2006.

[26] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path tcp: A joint congestion control and routing scheme to exploit path diversity in the internet," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 1260–1271, Dec. 2006.

[27] S. Barré, *Implementation and assessment of modern host-based multipath solutions.* PhD thesis, University of Applied Sciences, 2011.

[28] S. Barré, "Linshim6-implementation of the shim6 protocol," *Université catholique de Louvain, Tech. Rep*, 2008.

[29] IETF, "MPTCP contributions tracker." https://datatracker.ietf.org/wg/mptcp/documents/. Compilation of all the official contributions to MPTCP up to date. Accessed: 15-10-2015.

[30] U. of Louvain, "MPTCP contributions tracker." http://nrg.cs.ucl.ac.uk/mptcp/. Complitation of UCL contributions to MPTCP up to date. Accessed: 15-10-2015.

[31] S. U. of Technology, "Multipath TCP for FreeBSD v0.1." http://lists.freebsd.org/pipermail/freebsd-net/2013-March/034882.html, 2013.

[32] F. Technologies, "Release Note: BIG-IP LTM and TMOS 11.5.0." https://support.f5.com/kb/en-us/products/big-ip_ltm/releasenotes/product/relnote-ltm-11-5-0.html#rn_new, 2014.

[33] J. Gudmundson, "Maximize mobile user experience with NetScaler Multipath TCP." https://www.citrix.com/blogs/2013/05/28/maximize-mobile-user-experience-with-netscaler-multipath-tcp/, 2013.

[34] "iOS: Multipath TCP Support in iOS 7." https://support.apple.com/en-us/HT201373, 2014.

[35] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/WiFi handover with multipath TCP," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, pp. 31–36, ACM, 2012.

[36] C. Raiciu, C. Pluntke, S. Barre, A. Greenhalgh, D. Wischik, and M. Handley, "Data center networking with multipath TCP," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 10, ACM, 2010.

[37] "The Linux kernel MultiPath TCP project." Official website, http://multipath-tcp.org/.

[38] I. S. Association *et al.*, "Ieee std. 802.11-2012," *Piscataway, USA*, 2012.

[39] I. Corp., "Wireless Networking." http://www.intel.com/support/wireless/wlan/sb/CS-025325.htm, 2014.

[40] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, "Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs," in *Proceedings of the second international workshop on Mobility management & wireless access protocols*, pp. 19–26, ACM, 2004.

[41] J. Geier, "How wi-fi roaming really works." http://www.wireless-nets.com/resources/tutorials/how_roaming_works.html.

[42] M. D. David Murray and T. Koziniec, "Scanning delays in 802.11 networks," 2007.

[43] "Mobile technologies: GSM." ETSI. http://www.etsi.org/technologies-clusters/technologies/mobile/gsm.

[44] M. comms technology, "Global Telecom Insight." http://www.mobilecomms-technology.com/projects/imt2000/.

[45] ITU-T, Rec and Recommend, I, "G. 114," *One-way transmission time*, vol. 18, 2000.

[46] O. UK, "Infrastructure Report," tech. rep., 2014. http://stakeholders.ofcom.org.uk/binaries/research/infrastructure/2014/infrastructure-14.pdf.

[47] T-Labs, "The DSL Community from T-Labs." http://www.laboratories.telekom.com/public/English/Innovation/Exponate/Pages/DSL-Community.aspx.

[48] D. Cereijo and I. Romero, "Mptcp demo - handover performance." Video streaming over MPTCP in MAR demonstration. https://www.youtube.com/watch?v=JlXJCTGVDA4&feature=youtu.be.

[49] D. Cereijo and I. Romero, "Mptcp demo - seamless handover on a video streaming session.." Throughput as seen from a server which is streaming a video to a MPTCP enabled smartphone in a multipath transmission. https://www.youtube.com/watch?v=nESvFiC28t8.

[50] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of multipath tcp performance over wireless networks," in *Proceedings of the 2013 conference on Internet measurement conference*, pp. 455–468, ACM, 2013.